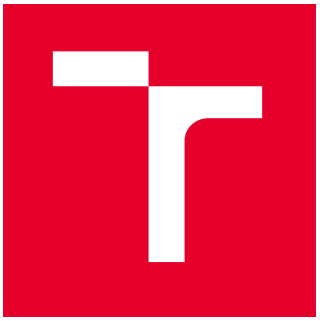


BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING

FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

DESIGN AND PRODUCTION OF FDM PRINTER FOR ADVANCED PRINTING TECHNIQUES TESTING

NÁVRH A VÝROBA FDM TISKÁRNY PRO TESTOVÁNÍ POKROČILÝCH TISKOVÝCH FUNKCÍ

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR — **Kristián Haris**

AUTOR PRÁCE

SUPERVISOR — **Ing. Filip Kšica, Ph.D.**

VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Assignment Bachelor's Thesis

Institut:	Institute of Solid Mechanics, Mechatronics and Biomechanics
Student:	**Kristián Haris**
Degree programm:	Mechatronics
Branch:	no specialisation
Supervisor:	**Ing. Filip Kšica, Ph.D.**
Academic year:	2023/24

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Bachelor's Thesis:

Design and production of FDM printer for advanced printing techniques testing

Brief Description:

The current state of Fused Deposition Modeling (FDM) technology in the field of 3D printing presents a challenge in optimizing printer control to achieve advanced printing capabilities. Currently, a variety of different approaches are used for controlling FDM 3D printers, but there is room for innovation and improvement. Typically, readily available devices in the semi–professional price range do not offer an easy implementation of advanced printing functions. Therefore, it is often necessary to either retrofit existing printers or entirely build a custom printer from scratch. The goal of this work is to analyze the current practices in printer control and use them to design and assemble an FDM 3D printer based on the CoreXY principle, which will be dedicated to testing advanced printing functions.

Bachelor's Thesis goals:

1. Summary of currently utilized techniques for FDM 3D printer control.
2. Design and assembly of an FDM 3D printer based on the CoreXY principle suitable for testing of advanced printing functions.
3. Implementation of selected firmware into a suitable control unit and calibration of the printer.
4. Testing of printing functions and comparison of speed, print quality, and other key parameters with a chosen commercial printer.

Recommended bibliography:

CANO-VICENT, Alba, Murtaza M. TAMBUWALA, Sk Sarif HASSAN, Debmalya BARH, Alaa A.A. ALJABALI, Martin BIRKETT, Arun ARJUNAN a Ángel SERRANO-AROCA, 2021. Fused deposition modelling: Current status, methodology, applications and future prospects. Additive Manufacturing [online]. 47(September). ISSN 22148604. Dostupné z: doi:10.1016/j.addma.2021.102378

JANDYAL, Anketa, Ikshita CHATURVEDI, Ishika WAZIR, Ankush RAINA a Mir Irfan UL HAQ, 2022. 3D printing – A review of processes, materials and applications in industry 4.0. Sustainable Operations and Computers [online]. B.m.: Elsevier B.V., 3(May 2021), 33–42. ISSN 26664127. Dostupné z: doi:10.1016/j.susoc.2021.09.004

KUN, Krisztián, 2016. Reconstruction and development of a 3D printer using FDM technology. Procedia Engineering [online]. B.m.: The Author(s), 149(June), 203–211. ISSN 18777058. Dostupné z: doi:10.1016/j.proeng.2016.06.657

Deadline for submission Bachelor's Thesis is given by the Schedule of the Academic year 2023/24

In Brno,

L. S.

prof. Ing. Jindřich Petruška, CSc. Director of the Institute	doc. Ing. Jiří Hlinka, Ph.D. FME dean

Abstrakt

Táto bakalárska práca sa zaoberá procesom navrhovania a zostavovania FDM tlačiarne, na ktorej budú otestované pokročilé tlačové funkcie. Prvá časť pokrýva teoretické podklady pre mechaniku a software, ktorý bude použitý. Druhá časť prechádza samotným návrhom a zostavením tejto tlačiarne, vrátane ladenia a sprevádzkovania. Práca je zakončená porovnaním s komerčnou tlačiarnou.

Summary

This bachelor thesis describes the process of designing and building an FDM printer to test advanced printing functions. The first part covers the theoretical background for the mechanics and software involved. The second part explains the design and build process, including calibration and tuning. Lastly, the work is concluded by a comparison with a commercial printer.

Klíčová slova

FDM, 3D tlač, CoreXY, Input shaping

Keywords

FDM, 3D printing, CoreXY, Input shaping

Bibliographic citation

HARIS, Kristián. Design and production of FDM printer for advanced printing techniques testing. Brno, 2024. Available from: https://www.vut.cz/studenti/zav-prace/detail/157438. Bachelor thesis. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Solid Mechanics, Mechatronics and Biomechanics. Thesis supervisor Filip Kšica.

I declare that this bachelor thesis titled Design and production of FDM printer for advanced printing techniques testing, is my own work and the result of my own original doing and research, with the help of cited references.

Kristián Haris

Brno . . . .

I would like to thank my supervisor Ing. Filip Kšica, Ph.D. for his assistance and constructive criticism during consultations. And I would also like to thank my close ones, who provided support during my studies.

Kristián Haris

Contents

1 Introduction

FDM printers are very useful machines for various applications. As of writing, they are becoming popular not only for industrial use, but also for casual users. Their main advantage is the ability to quickly produce a designed object without the need of specialized equipment or environment. With the development of higher performing mechanical parts, and increasingly advanced printing functions, they can even be used for a small-scale commercial application.

The point of this thesis is to design, assemble and tune a custom printer, which will be suitable to test advanced functions. These should allow to print at higher quality, while shortening the print time. In a rapid prototyping process, time can be of great value, so the development of such functions is explainable.

This work will also show, how much can be achieved with a custom design that will be built without special tools and expensive parts.

2 Analysis of motion and fuctions

This chapter covers the basic building blocks of any FDM printer. The kinematic motion system, which is one of the most distinguishing features when choosing between printers. A description of motors and drivers, their respective parameters and functions. And advanced printing functions, such as input shaping and pressure advance, that are now used in modern printers.

2.1 Kinematic platforms in FDM 3D printers

There are several options to consider when designing an FDM (fused deposition modeling) printer from scratch. One of the most important decisions to be made is the type of kinematic system to be used. This choice will have a significant impact not only on the size and price of the machine, but also on the printing characteristics and other capabilities. Popular choices include Cartesian, CoreXY, H-Bot and Delta.

Cartesian
The simplest and most widely used kinematic platform is Cartesian. In this system, all three axes of the printer are separate, each having its own motor or set of motors that move their respective masses in one direction. This system can be seen in the left section of 2.1. There are several reasons why this is the platform of choice for the majority of 3D printers in the world.

One of the important factors that is favoring, is packaging compactness. Cartesian printers are often *bed-slinger*, which means that the printing surface moves on the Y axis and the print head moves on the X and Z axes. This configuration has a footprint of at least twice the area of the print surface, but otherwise the Z-axis consists of only two slim towers. Because each axis is driven by its own motor or set of motors, motion calculation is also simpler than with other types of platforms. This is because movement in one direction is performed by commanding a specific angular position that is linearly related to the actual movement of the component. When using a belt-driven system, another upside is the relatively short belt length. Since belts affect the dynamics of the motion system, it must be considered that the use of longer belts may cause more unwanted oscillations during rapid speed changes.

But such a system also has it's disadvantages. For example, the mass of the Y-axis, which consists of the print surface, often with a heating solution. Not only is this component relatively heavy on its own, but it's mass increases as the printer produces a part. This means that the maximum acceleration and oscillations of this axis are not constant during printing.

CoreXY

Another popular choice, especially in recent years, is the CoreXY. This platform uses two motors to move the print head in the X and Y axes. Some designs also move the entire XY assembly on the Z-axis and keep the print surface static. However, a more common choice is to attach the print surface to a moving Z-axis. Because the Z movements during printing are small and require relatively low acceleration, they are not a significant limitation. The motion of the print head is a combination of two motor rotations. In this platform, they are often referred to as motors A and B, instead of X and Y. The reason for this comes from the analysis of the motion equation.

$$\Delta X = \frac{1}{2}(\Delta A + \Delta B) \quad \text{and} \quad \Delta Y = \frac{1}{2}(\Delta A - \Delta B) \tag{2.1}$$

To move the print head in the X-axis, motors A and B must have the same direction of rotation. To move in the Y-axis, opposite directions of rotation are needed. If only one motor moves, diagonal motion is performed. Note that this is the limiting factor when considering speed and acceleration.

CoreXY machines do not suffer from the problem of moving the mass of the print surface and the printed part on a fast axis. On the other hand, they require longer belts, which can cause more oscillations. Another difference is the routing of these belts. Cartesian systems require only one idler and one drive pulley for an axis. So for a functional motion platform that moves in the X and Y axes, the CoreXY uses 8 idlers while the Cartesian uses only two. These pulleys do not have a significant moment of inertia, but having more moving parts is rarely considered an advantage.

These machines are often box-shaped, which can result in a less compact printer. However, this shape can be useful when considering a heated chamber. Even just enclosing the outer faces can help when printing various warp-prone materials.

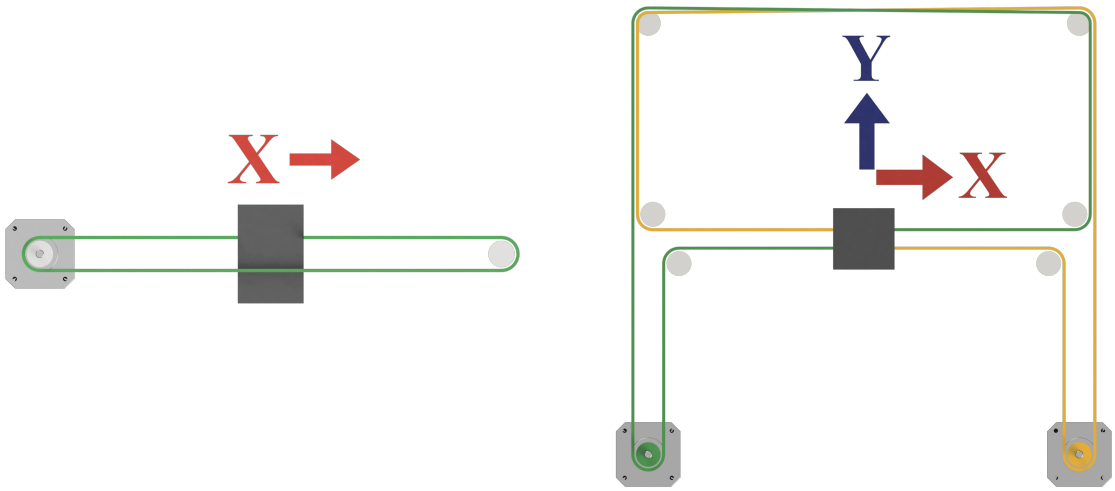


Figure 2.1: Cartesian and CoreXY kinematics

2.2 Motion systems: motors and drivers

The core of a motion system in a 3D printer must meet certain requirements. First and foremost, precise motion without positional deviation. In addition, maximal speed and a acceleration must be specified. Most common choices would lead to servo or stepper motors. Servo drives being superior to the latter. They offer closed-loop control that ensures positional accuracy. Stepper motors lack an encoder on their rotor, most of the available drive circuits have no way of detecting whether a motor has skipped a step. This is a common issue in 3D printing. When the print-head collides with extruded plastic, the motor can be overloaded. This results in a loss of position and failure of the print. Servo motors are the more capable option, but in reality, almost all printers use steppers. This is mainly due to the price difference. For this reason, most of the commercially available controllers only have drivers for steppers [1].

Stepper motors

The most common stepper motor choices for this application are bipolar motors in the NEMA 14, 17 or even 23 packages. NEMA is a size standard [2], the number defines the length of one side of the square-shaped face of the motor. It is provided in inches, multiplied by 10. From each package, there are multiple options for the length of the body. Other important parameters include step angle, holding torque, phase current, resistance and inductance.

Step angle defines the angle the rotor will turn on one full-step. There is some debate as to whether 1.8 or 0.9 degree step angle motors are better. The choice depends on several factors. It is true, that by reducing the step angle, better motion resolution can be achieved. However, with two times the steps for one full rotation, calculations must be made to ensure that the motor will be able to reach the desired maximum angular velocity. Since stepper motors are driven by pulses, the 0.9 degree motor will have double the frequency of pulses on it's coils. This means that the phase inductance will show it's effects sooner. The inductive back EMF (electromotive force) generated, will approach the supply voltage level sooner. Torque output drops, with lower angular velocity, than in an equivalent 1.8 degree motor. These facts conclude, that selecting a stepper motor based only on the torque output alone can be a mistake. All parameters must be evaluated in conjunction with the desired maximum speed and acceleration. Another important factor, especially in high-speed 3D printing, is rotor inertia. Larger steppers provide higher torque and can have low inductance, but their rotor inertia will be significantly higher than smaller steppers. The effects of this parameter are discussed in more detail in 3.3.

Stepper drivers

Since the advent of 3D printers, the development of drivers has followed suit. At the time of the first RepRap machines, the A4988 chip from Allegro microsystems was the standard [3]. This driver operates in a constant current drive mode, with a specific chopper profile that switches between different decay modes. The output current is set by a voltage reference. In most cases, these drivers were placed on a small PCB, which was then plugged into the controller board of the printer. This format can be seen on 2.2. The voltage reference could be set with a tiny potentiometer. The finest micro-stepping for the A4988 was 1/16 of a step. This also had to be set physically, with a set of jumper pins, and could not be changed while the printer was running.

The next widely used driver was the DRV8825 from Texas Instruments [4]. It also used a voltage reference for the current setting, and pins for micro-stepping. But it featured 1/32 micro-stepping and was running a little quieter chopper profile than that of the A4988.

The big change came with the announcement of Trinamic's TMC series of drivers [5]. They are available in several configurations, but the most used in 3D printing are the TMC2208, TMC2209 and TMC2130. Using these drivers eliminates the need to set a voltage reference for the current setting and pin changing for micro-stepping. The driver can run in UART mode, where certain registers can be used to set these parameters. This allows different drive profiles to be created directly in the firmware of a printer. Another register defines the chopper mode. These drivers provide an option to use one specifically designed for quiet operation. Another advantage is a feature that detects the stalling of a motor. It is still not enough to catch up with a closed-loop driven servo motor, but it can be useful to create a virtual endstop for an axis. The motor will simply drive the print-head to a mechanical stop at low speed and use the stall detection feature to stop. This works by monitoring a few key parameters, including the back EMF generated by the motor as it rotates. The UART communication comes handy because now different run modes can be created and issued right from the firmware without having to physically touch the electronics.

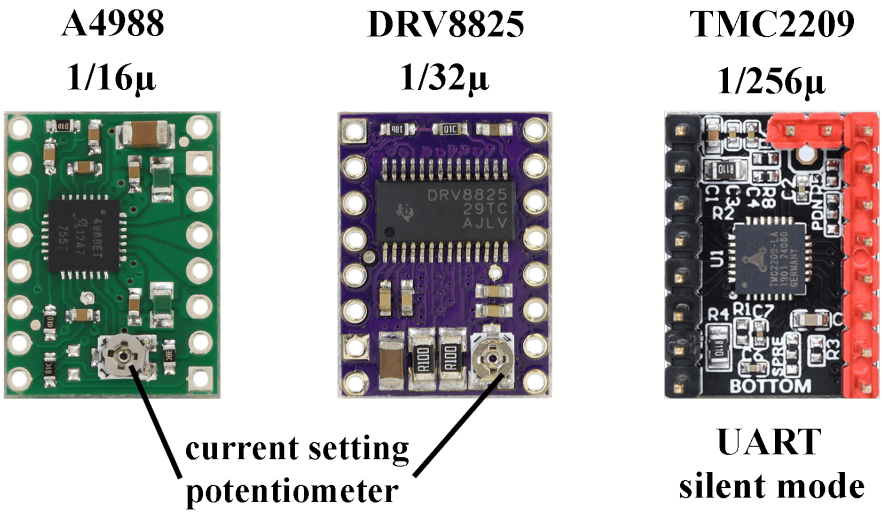


Figure 2.2: Stepper drivers

It is important to note the relationship between torque output and different driver parameters. While finer micro-stepping provides better resolution, it significantly reduces torque. The TMC drivers can switch between the SpreadCycle and StealthChop chopper modes. Spreadcycle is already quieter than an A4988 or DRV8825, while providing the same torque. StealthChop is specifically designed for very silent operation, but the torque output is reduced, especially at higher motor speeds. Therefore, it should be used with caution. If a printer profile with lower speeds and accelerations is used, the silent chopper mode can be used and will make the printer more comfortable to be around. However, if noise level is not a priority, SpreadCycle is a more reliable and powerful option. A diagram of the chopper mode can be seen on 2.3. Where *on*, *sd* and *fd* are the different decay phases used to maintain target current. These phases are shown on a simplified diagram in 2.4. These diagrams come from the pages of the manufacturer [6].

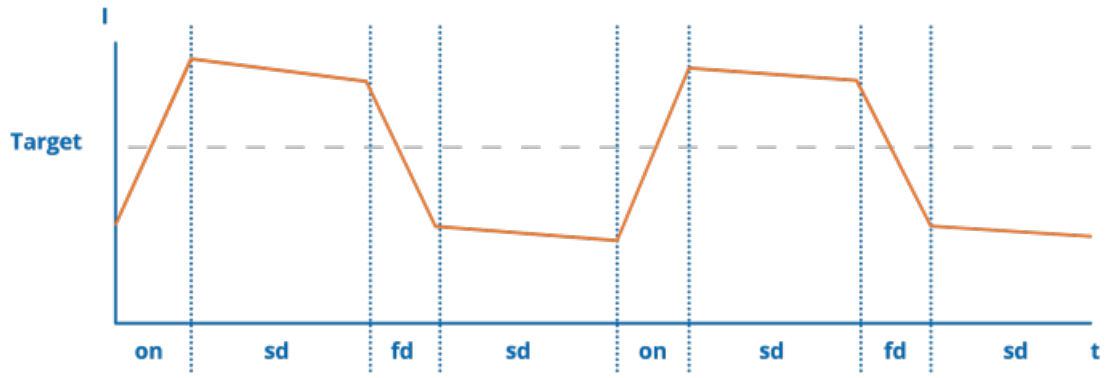


Figure 2.3: Trinamic SpreadCycle chopper mode. Source: [7]

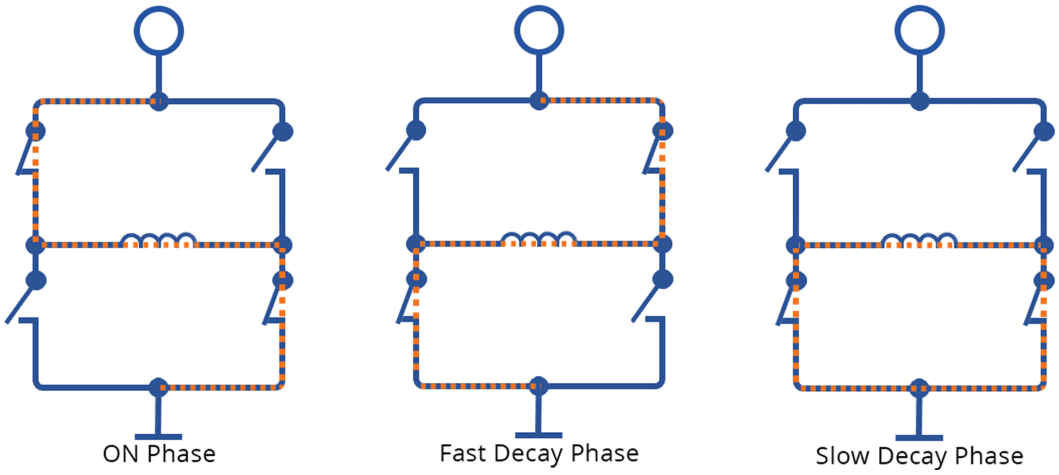


Figure 2.4: Stepper driver phases. Source: [7]

2.3 Firmware functions and options

Firmware is an essential part of a 3D printer's control system. It defines how all the functional parts should operate. During a print, the firmware processes commands that were generated for the print job and translates them into steps for the motors, heating of elements, and others. Today, there are more than a dozen firmwares available, most of them as a free open-source project.

Open-source firmware
In the early days of 3D printing, an open-source firmware called Marlin was the common choice [8]. It was written in C, primarily to be compiled for Arduino boards. It was popular to use an Arduino Mega with a breakout board to control a custom printer. Even Prusa Research's Prusa Firmware is a fork of the original Marlin firmware[9]. Another one that will be mentioned from personal experience is SmoothieWare, mainly developed for the Smoothieboard controller [10]. It is much less popular than other options, but it was one of the first 32-bit boards at the time. It had a lot of configuration options and an Ethernet port that allowed the user to use a web interface.

Klipper
In this work, the chosen firmware for the build is Klipper. It is one of the newest and most advanced firmwares, created in 2018 by Kevin O'Connor. It is available for anyone, and has more than 400 contributors on GitHub [11]. This firmware is unique because it utilizes a more powerful computer that runs Linux, as the host, and then the controller as the client. This way, all the complex calculations are done on the more powerful standalone computer, instead of the computationally weaker controller. Here are just a few of the key features, taken the webpage of this project [12]:

- High precision movement calculations (without kinematic estimations),
- Very high stepping rates, even for older micro-controllers,
- Support for a wide range of hardware,
- Support for most kinematic systems,
- Separate printer configuration file that requires no re-flashing,
- All code written in Python for easy development,
- Input shaping and pressure advance with built in tools for tuning,
- Built-in API server and several web interfaces.

Because of these benefits, many older printers running outdated firmware are turning to Klipper for an upgrade. With just the addition of a standalone computer, even an entry-level machine with old hardware can become more accurate, capable, and fast. And an old segmented display with an SD Card slot can be replaced with a refreshing web UI (user interface) with programmable macros. All for free, thanks to the open-source community.

2.4 Advanced printing functions

Advanced printing functions are new additions to modern firmware. They aim to increase print quality and speed. One of the biggest changes was the implementation of input shaping. This section provides an introduction to some of these features.

2.4.1 Input shaping

When analyzing the fast moving axes of a common 3D printer, one can observe a classical example of a dynamic system. The components having their respective masses and moments of inertia. And the rubber belts having considerable stiffness and dampening. It can be expected, that when such an assembly is forced to accelerate, the actual motion may differ from the requested due to vibrations. The print-head experiences damped oscillations around the target position. This effect can be seen on 2.5.

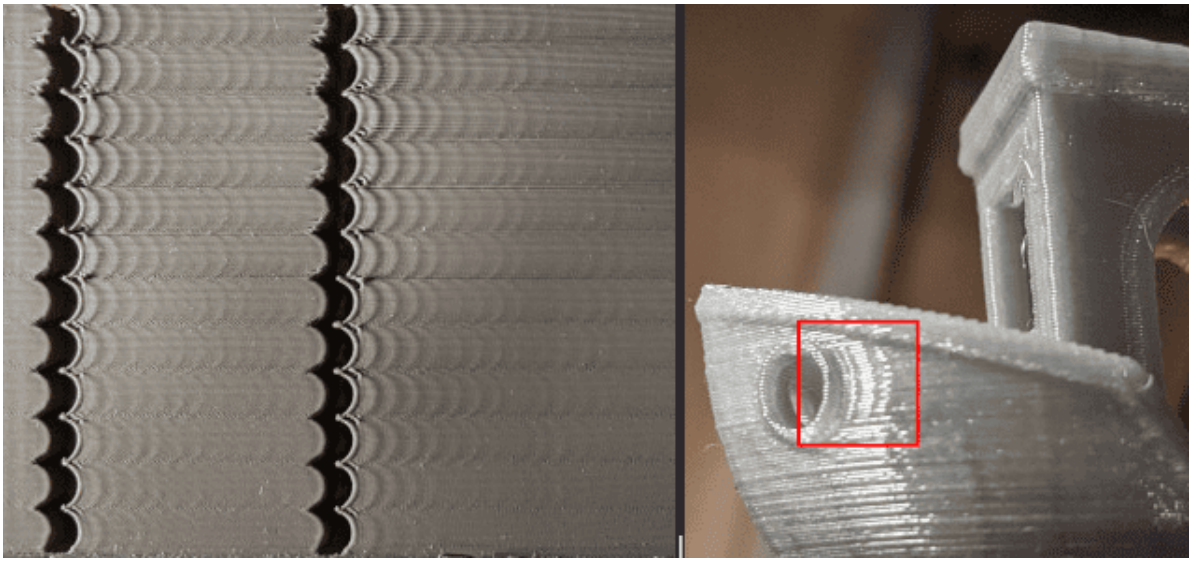


Figure 2.5: Ghosting effect on surfaces. Source: [12]

To mitigate this, input shaping algorithms have been implemented in 3D printing firmware. These algorithms alter the motion at the input to try to achieve a real output that is closer to the requested. The key parameter to achieve this is to determine the resonant frequencies of the axes. Since the Z-axis is slow or static in most cases, no input shaping is necessary. The X and Y axes are of interest. Once a resonant frequency is found, a shaper algorithm can be applied, which then has a certain progression of damping, that eliminates oscillations as a final result. In reality, this is a bit more complex, as will be discussed in 4.2.

When properly tuned, this feature not only eliminates ghosting effects on the printed object, but also reduces vibrations which put strain on components. This means that significantly higher accelerations can be achieved during printing without the need to face the mechanical consequences. Higher accelerations cut significant amounts of printing time, which can be a great benefit, since FDM 3D printing is a *rapid* prototyping process.

2.4.2 Pressure advance

Another problem that occurs, especially at high print speeds, happens in the extrusion system. During a rapid change of direction, the extruder gears pull or push the filament according to exact theoretical calculations of motion. The problem is that molten filament often behaves differently than the ideal motion equations predict. Pressure is created as the solid filament is pushed against the melt zone of the hot-end, and this pressure causes a delay effect on the amount of plastic actually extruded [13]. As illustrated in 2.6.

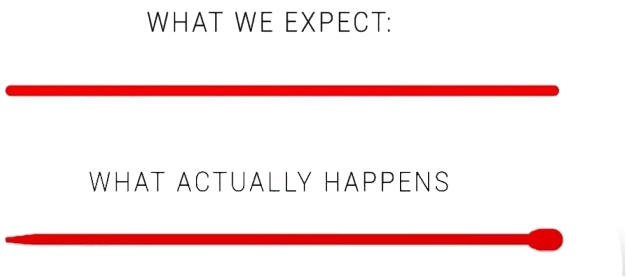


Figure 2.6: Extrusion line

Pressure advance is a relatively new printing function that attempts to eliminate these unwanted effects. By tuning two parameters, time and smooth time, the extruder gears create a jerking motion to quickly reduce or increase pressure in the melt zone to negate the unwanted effects 2.7. These parameters are generally more aggressive in bowden setups than in direct-drive extruders. Because the length between the gears and the melt zone is much greater, the filament itself acts as a spring, amplifying this unwanted behavior. But at very high print speeds and flow rates, even direct-drive systems benefit greatly from a well tuned pressure advance function.

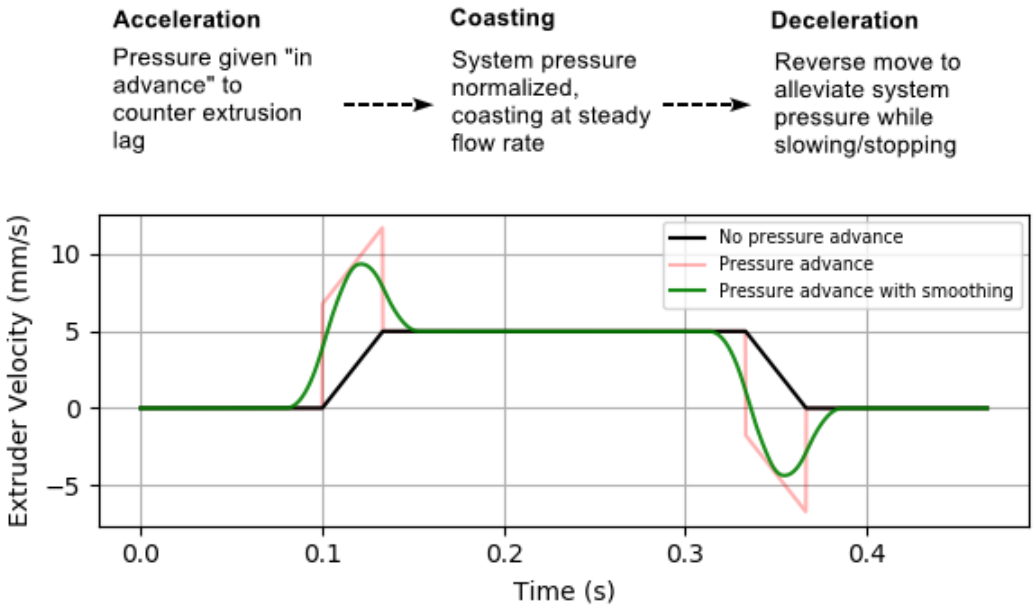


Figure 2.7: Pressure advance graph. Source: [14]

2.4.3 Experimental functions

The two functions mentioned in the previous sections are implemented in the firmware of a printer. They only modify the behavior of the motion system in a way that improves the printing process of any part. The next function is of different nature. It creates unconventional print code that can be run on any printer. It does not modify how the printer works.

Full control

This technique generates special gcode, a set of commands that the printer processes while printing a part to create objects in a different way than usual. Most of the time, the 3D printing workflow consists of importing a CAD model into a slicer software, which generates gcode from the imported geometry based on general rules. These always include layer height, wall thickness, infill density and pattern, and more. The full control technique shows something new here, it is basically a set of tools that generate special gcode for certain objects that would be conventionally very difficult or impossible to print. For some objects it even chooses not to use consistent layering to build them. Such a process is visualized in 2.8.

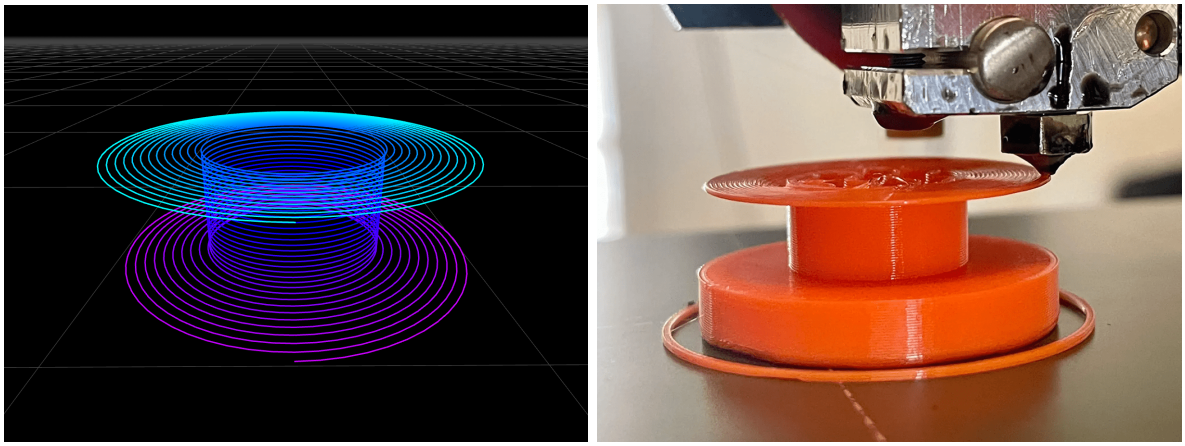


Figure 2.8: Full control gcode and print. Source: [15]

When trying this printing technique, it is fascinating to see how this unconventional method can generate gcode that actually creates very challenging geometries. At the time of writing, there are several tools that generate such code, some of them available at [15]. These methods use creative ways to overcome problems that basic slicing software cannot, or must use support structure or other tricks to overcome. Since it is a fundamental property of 3D printing to be able to print almost anything that can be designed in CAD, slicer algorithms try their best to be general purpose. But there is only so much they can do. Writing gcode manually would allow smarter printing, but gcodes can easily be over a million lines long, so this is unthinkable. Full control is a step towards more creative gcode generation but unfortunately, it has a very limited use case. This technique cannot be used on any self-designed model. As of writing, it is just a set of tools that create certain geometries, which is a very interesting process, but hardly usable in the real world for prototyping.

3 Design and build of CoreXY printer

This chapter describes the process of designing and building the printer in question. In reality, it took more iterations to get to the final version of the design. It was a back and forth process of designing, printing, and testing.

3.1 Specifications and existing solutions

For the machine that will be the topic of this thesis, following parameters were specified :

- 300x300x300 mm build volume
- CoreXY kinematic platform
- Stiff aluminum frame
- High printing speed
- Direct-drive extruder
- Open-source software
- Low cost components

When designing a 3D printer from scratch, the resources available from open-source communities can be a great help. When printers started to become popular with basic users and hobbyists, the RepRap project was the foundation. It is an idea that a custom machine should have most of its parts designed so that the finished printer would be able to reproduce most of its own mechanical parts. This idea is still alive today in many modern designs. Expensive professional CNC machining is tried to be avoided during this build. Universal parts that are easily obtainable for a relatively low price will be used as they are. But for the mechanical parts, a custom design will be created and then 3D printed. This allow the parts to be carefully designed with low weight and high stiffness in mind.

Lately, a company called BambuLab, which makes very advanced and capable machines at competitive prices, decided to base its main product line on the CoreXY motion system [16]. These machines have similar characteristics to those mentioned above. They are a great option for a wide range of users, but they implement some proprietary closed-source solutions, that couldn't be used as inspiration in this building process. Instead, a few open-source alternatives were analyzed and used as inspiration.

First, the Voron 2.4 by VoronDesign [17]. A render of a certain configuration is shown in 3.1. This model exists as an online guide that serves as a configuration tool to create a list of parts that are needed to construct such a printer. The Voron 2.4 is a very clever solution that implements a lot of high quality and high performance parts. It utilizes linear rails and a special high flowing print-head. While this printer has a CoreXY motion system, it also uses rails and belts to move the whole XY assembly in the Z-axis, which then results in a static print surface. The only deviation from the mentioned specifications is the use of more expensive components and the static build surface.

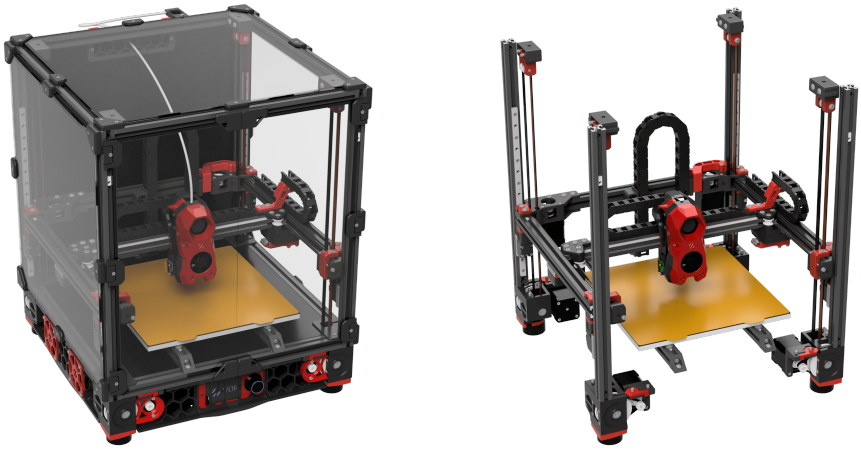


Figure 3.1: Voron 2.4 render

The main source of inspiration is a design called HEVO [18]. One of the possible configurations is rendered in 3.2. It is a redesign of a project called HyperCube which is listed as a RepRap printer. The main differences are in the Z-axis and the extruder type. HEVO implements the usual dual leadscrew design for the Z-axis, which moves the print surface. For X and Y motion, the CoreXY platform uses linear rails, similar to the Voron 2.4. While that uses a direct drive extruder on top of the print-head assembly with a lightweight stepper motor, HEVO opted for a bowden extruder with a more common full-sized motor. This design complies closely with the specifications.

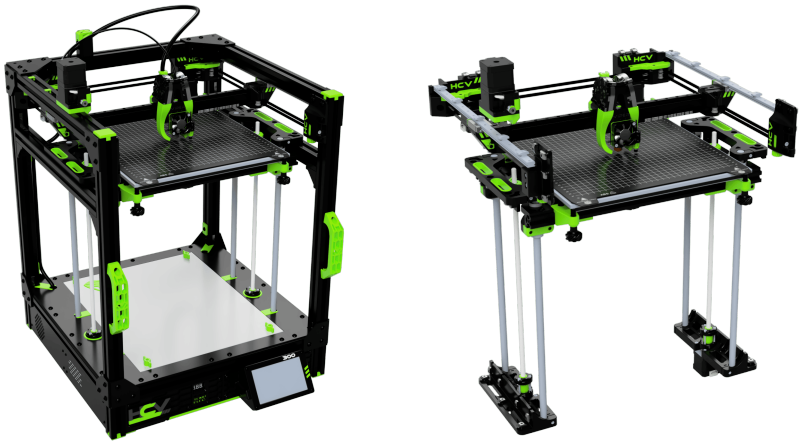


Figure 3.2: HEVO render

3.2 Design and 3D model

The design process begun with assessing some of the components that were already available. The print surface with an integrated heater and a used extruder assembly from E3D were selected to be included in the build.

Frame and Z-axis

For increased stiffness, the frame will consist of 30x30 mm aluminum profiles with machined corner pieces ensuring low skew and rigidity. Only the support structure for the heated print surface uses thinner 20x20 mm profiles, as this part isn't expected to withstand higher loads.

The Z-axis was a relatively straightforward task. Neither rapid accelerations nor high speeds are required here, so weight savings were not a priority. The linear motion is supported by four 12 mm smooth steel rods with recirculating ball bearings on the moving frame. Two stepper motors working in parallel move two leadscrews on each side. Only 5 printed parts were modeled, then mirrored and rotated on the appropriate axes to create an assembly of 16 parts in total. Another small addition are corner brackets that hold the heated print surface to the aluminum frame underneath. They also allow for leveling with screws located on each corner.

To keep all the electronics all organized in one space, a compartment was created on the bottom of the frame. A honeycomb-patterned plate consisting of four pieces that screw together creates an undertray. This pattern not only saves filament when printing, but also allows the use of zip ties for cable management. The cables have not been modeled or rendered. The same goes for the belts. Since this model was intended as a functional design to aid in the development of parts, there was no benefit in modeling those.

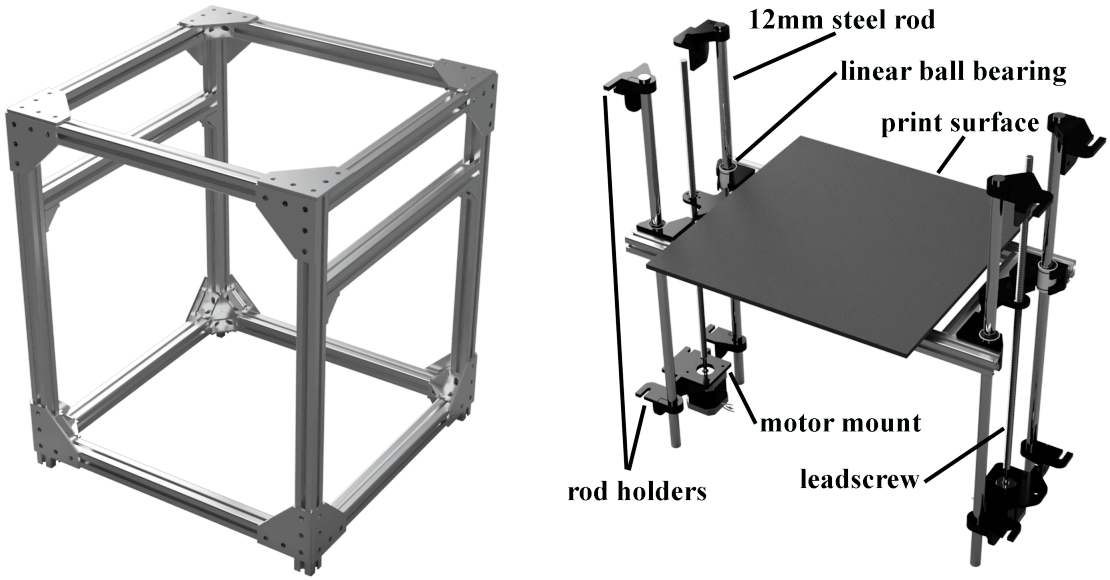


Figure 3.3: Render of frame and Z-axis assembly

XY assembly

This process of development involved a lot of trial and error. The design went through several iterations and a number of components were tested along the way. First, it was decided that the motor size would be NEMA17. The motor mounting brackets are designed for this size, but the motor length can be changed if needed. The second decision was to use 6 mm wide GT2 profile belts. This is the standard for many printers and was readily available from local vendors. Third, 10 mm and 8 mm steel rods were selected for the Y and X axis respectively. For minimizing weight, the XY connecting parts and the print-head carriage were designed around plastic dry-running linear bushings. They are considerably lighter than equivalent recirculating ball bearings.

A lot of time was invested into the design of the X and Y assembly and the print-head carriage. These parts are subjected to high speeds and accelerations. Weight savings had to be maximized while still keeping the parts printable on a regular 3D printer. This entire assembly consists of several parts. There are two motor brackets at the top front corners that attach to the frame. Then, at the opposite corners, there are two smaller parts that serve as fixtures for the Y-axis rods and idler pulleys. On these rods, there are the XY connecting parts. Another set of idlers, bushings, and the X-rods are attached to these. Finally, there is the print head carriage which holds the extruder motor and hot-end assembly. The back side of the carriage has slots for bushings and a belt tensioning mechanism. The protruding parts are the part cooling fan ducts.

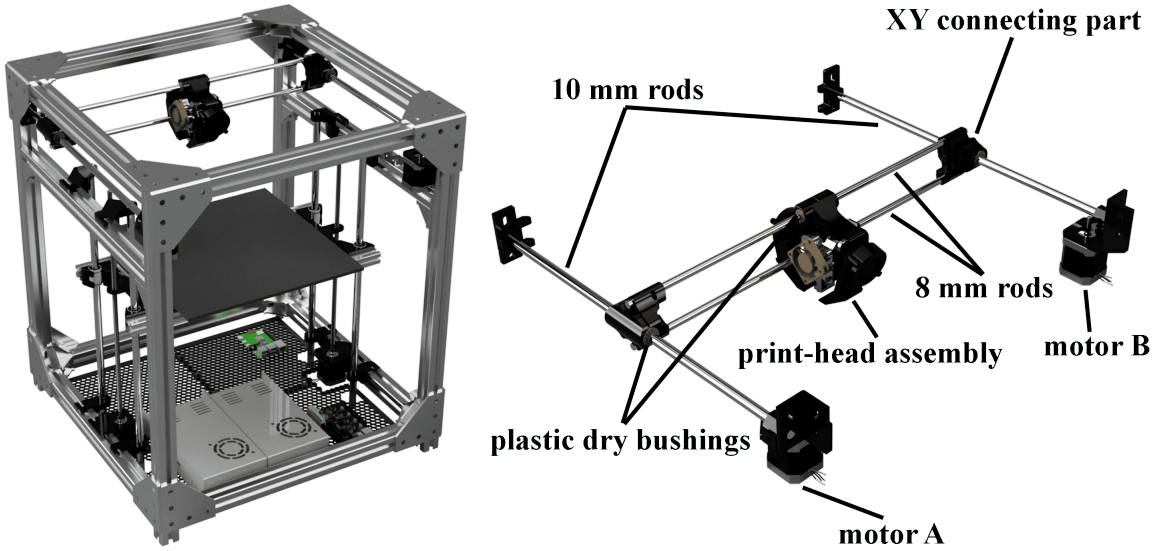


Figure 3.4: Render of complete printer and XY assembly

Another design using lightweight hollow aluminum shafts for the X-axis, was also created, but this configuration wasn't built or tested. While it would make the assembly significantly lighter, it proved difficult to find such precision shafts at a price that would be acceptable for this build.

3.3 Dynamic analysis

It was of interest and value to delve deeper into a dynamics analysis of the designed motion platform. The software of choice for this process was Matlab. First, a basic review of the motion in a CoreXY system was performed. When the print-head is moved in the X-axis, the moved mass consists of the print-head assembly, 10 pulleys, and two rotors of the motors. Both steppers contribute to this movement. A move in the Y-axis is the same, but the mass is increased by the X rods and the XY connecting parts. Most interesting is the diagonal move. The CoreXY system, with it's belt routing, achieves motion in this direction by rotating only one motor and holding the other steady. The implication is that the actual maximum acceleration is likely to be limited by a move in this axis.

The calculations in the script are based on the reduction of masses and inertias to a single point of interest that will be doing a translational movement. This point will be the print-head for all cases. The reduction method was applied three times, separately for the X, Y, and diagonal axes. For the X-axis:

$$m_{redx} = m_{ph} + 2\frac{I_{dp} + I_{rotor}}{r_{dp}} + 8\frac{I_p}{r_p^2} \tag{3.1}$$

where

- m_{redx} is the reduced mass on the print-head,
- m_{ph} is the mass of the print head assembly,
- I_{dp} is the inertia of the drive pulley,
- I_{rotor} is the rotor inertia of the stepper motor,
- I_p is the inertia of a single belt pulley,
- r_{dp} and r_p are the respective pulley radii.

These equations neglect the masses of the belts and the friction in the bushings and pulley bearings. After calculating the respective reduced masses for all directions, maximum acceleration can be calculated with the equation:

$$a_{maxX} = 2\frac{M_m}{m_{redX} \cdot r_p} \tag{3.2}$$

where

- a_{maxX} is the max achievable acceleration,
- M_m and the torque output of the stepper motor.

After calculating the maximum achievable accelerations for all directions, variation of the input parameters could be implemented in the script. Since the inertia of the pulleys was found to have little effect on the dynamics, all other parameters were selected for analysis. The output of the script are surface plots. Each of them shows the maximum acceleration on the Z-axis, in m/s^2. On the X and Y axis, there are be the weights of the print-head and XY assembly. Each input parameter has been increased and decreased by 50% from the estimated weight of the actual parts in the design. The red dot in the middle of the surface plots shows the point which corresponds to the weight of the actual design. First trio of surface plots can be seen on 3.5.

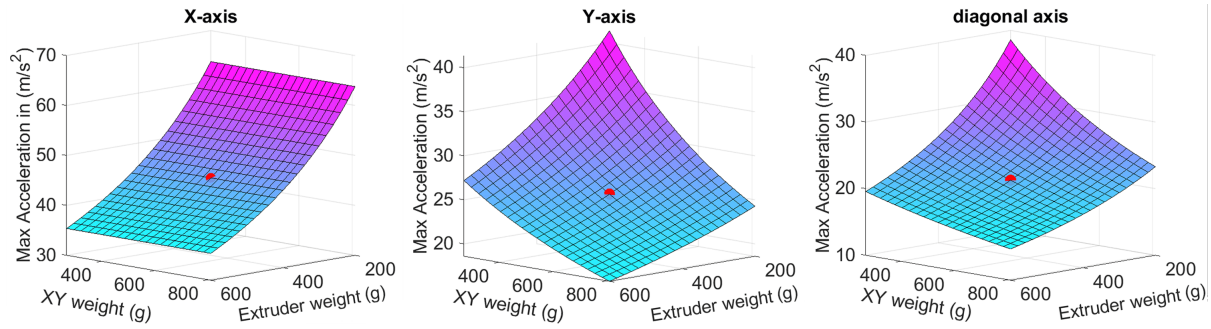


Figure 3.5: Maximum acceleration for weights

The next three surface plots were created with a 50% variation of the drive pulley diameter and the rotor inertia of the stepper motor.

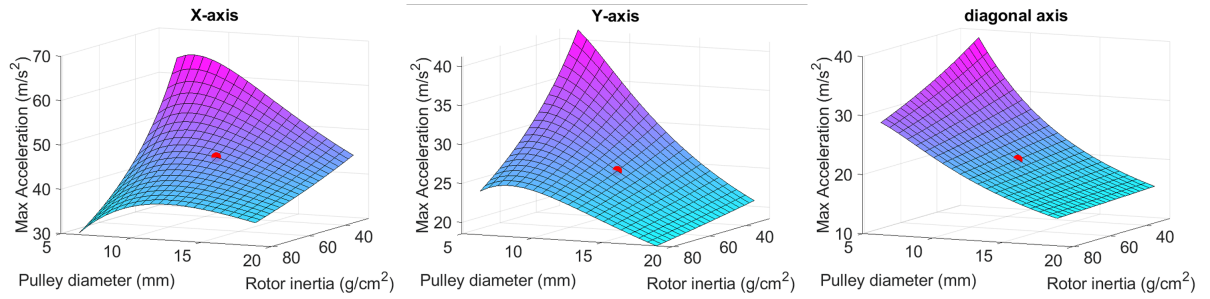


Figure 3.6: Maximum acceleration for diameter and inertia

Several conclusions can be drawn from analyzing these plots. First, the acceleration on the X-axis in independent on the XY assembly weight because it stays static. Second, variation of the drive pulley diameter shows a concave plot. This is caused by the changing rotational speed for the rotor inertia, which becomes the dominant factor in an X-axis move compared to the moving masses. But only up to a point. When the derivative of this curve is zero, the optimal pulley diameter can be found where the rotor inertia and the moving mass are in equilibrium. This effect is dampened in a Y-axis move because the moving mass is greater. It fades out even more in a diagonal move where only one of the motors is rotating. It can be observed that in both sets of plots, the diagonal moves show the lowest maximum acceleration, as expected according to 2.1.

3.4 Hardware assembly

3.4.1 Frame and motion platform

This section discusses the physical build with an emphasis on some of the processes that are of interest. The progression of this chapter will not be chronological. After the printer got to the point where it was able to print parts for itself, the external printer was no longer used, and more than a dozen versions for the parts were designed, printed, and tested.

Frame and Z-axis
First, the frame had to be assembled. The aluminum extrusions were cut to size, whilst trying to keep dimensional deviation to a minimum without precision machining. The main frame was then held together with inner and outer corner pieces and T-nuts that rotate inside the extrusions when tightened. After assembly, the dimensions were checked again.

With the frame complete, printed parts were next. These required another 3D printer, primarily a Prusa MK4, which was used for the first iteration of the designed parts. The first set of printed parts were mounted to the frame using mostly the same T-nuts, but with more care due to the brittle nature of some of the parts. One of the first challenges encountered was the positioning of the Z rods and leadscrews. For the first iteration, the alignment wasn't perfect, which was later corrected by adding oval holes for the motor mounts to allow for position adjustments.

XY assembly
By far the most time consuming part of the build and development was the X and Y assembly including the print-head. In the early stages, recirculating ball bearings were used on the X and Y axes of the motion system. They had almost no issues after assembly, but were considerably heavier and noisier than their plastic variants. Naturally, the second iteration included the implementation of dry-running plastic bushings. This involved a lot of trial and error. Because these bushing are all-plastic, the clamping forces exerted on them by the printed parts caused them to deform. This resulted in either binding or increased friction. Adjustments were made and even press-fitting was tested, which was very tricky to get right with printed parts. The conclusion from this experience is that it is wiser to either stick with the metal ball bearings or use plastic bushings that have a metal sleeve. Manufacturing a perfect press-fit with an FDM printer proved to be very difficult and inconsistent. On the print-head carriage a half-cut cylindrical slot was used to put pressure on the smaller plastic bushings. This design worked in the end, but took considerable time to get right.

By manually exerting force on the print head, it was checked for any binding or excessive friction. Once everything moved smoothly, the next task was to install the motors, pulleys, and belts. In total, two drive pulleys were installed on the motors, 4 idler pulleys in the XY connecting parts and another 4 idlers on the back corner of the frame. The motors could then be mounted on the printed mounts in the front corners of the frame. At this point the belts could be routed as shown in 2.1. For this, a genuine Gates GT2 belt was used, as it has a better surface finish, which reduces audible noise and friction during operation. In this design of the printed parts, the open ends of the belts are attached to the print-head assembly, where a tensioning mechanism is located. It works by forming a loop and then tightening a screw that increases the length of the entire loop. To calculate the required tension, the formula 3.3 was used. In practice, the belts were plucked and their frequency was measured with a smartphone. The force used in the equation came from analyzing the datasheets of the motors. From there, a limit was found on the radial force that can be applied to the shaft of the rotor.

$$f = \frac{1}{L} \cdot \sqrt{\frac{F}{\rho}} \tag{3.3}$$

where

- f is the resonant frequency of the plucked belt,
- L is the length of the belt between two points,
- F is the chosen tension force,
- ρ is the density of the belt.

In a CoreXY printer, it is important, that the two belts have the same tension. Since they work in conjunction with motors A and B, this results in motion in the X and Y space. For a first setup, it is not critical to achieve perfectly equal tension forces. However, later, when the input shaping function is be tested, a method for correcting any difference in belt tensions will be discussed in 4.2.

Vibrations
It was later discovered that when the printer ran at higher accelerations, the vibrations were conducted by the frame and the ground where the printer was placed. This caused unpleasant noises that were amplified throughout the room. Foam pads were mounted on standoffs placed below the protruding corner pieces of the frame that touch the ground. This reduced the noise by a significant amount.

3.4.2 Electronics integration

To connect all the electrical parts of the printer, custom wiring had to to be made. The connections were made according to 3.7. This schematic is derived from respective datasheets that were available [19].

Power and heating

Unfortunately, two power supplies had to be used. The heated bed available for this build has a measured resistance of only 0.8 Ω. The lower voltage power supply has a nominal voltage of 12 V, but can be adjusted with a potentiometer. Using this, it was set to 14 V to increase the heating power. Applying Ohm's law, we get:

$$P = U \cdot I = \frac{U^2}{R} = \frac{14^2}{0.8} = 245W \tag{3.4}$$

The low resistance of the heated bed would result in a power draw of over 700W with a 24 V power supply. However in order to use different voltages for the bed and the controller, an external switching board had to be implemented. The wiring for this module is straightforward, and presented no problems.

Except for the bed heater, everything else runs on 24 V. This is mainly beneficial, for the stepper motors. During operation, the back EMF generated by the steppers must be kept below the supply voltage, so a higher voltage allows higher rotational speeds.

Other electronics

Other components, such as the endstops, fans, and thermistors plug directly into the controller and require no further modifications. One fan is needed to cool the heat-break in the print-head, while the other one is used as a part cooling fan during printing. The X-axis endstop was replaced with a virtual one, discussed in 2.2. It was beneficial to use this feature here, because a hardware endstop would move relative to the frame, requiring additional wiring that would move with the XY assembly.

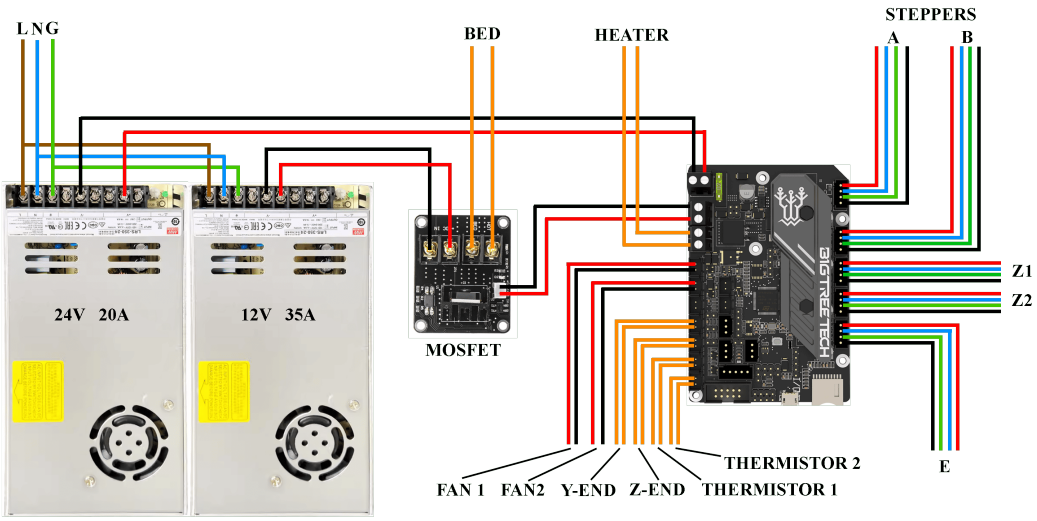


Figure 3.7: Electronics schematic

The chosen controller for this printer is the SKR mini E3 V3 from BigTreeTech. It may be a little unconventional to use such a small board with limited options, but the cost was significantly lower, and the capabilities were just satisfactory enough.

List of key parameters :

- 32-bit architecture CPU STM32G0B1
- 4x integrated TMC2209 stepper drivers
- 12-24 V input voltage
- 2x heater outputs
- 3x fan output
- BL-touch leveling sensor support

While it was beneficial to have integrated stepper drivers, pre-configured to run in UART mode, there is only four of them available. As can be seen on 3.7, five stepper motors are connected. This is is possible with a parallel connection for the two Z motors. They are controlled by a single driver. This means that the current is split between them, and no individual control can be realized. While it is not a necessity, with five drivers, better bed-leveling methods could be applied. Another limitation is the max current output. The two stepper motors on the Z axis have a higher combined rated current, than the driver can provide, so the rated current cannot be achieved. This was overcome by setting a lower acceleration and max rotational speed, which proved to be sufficient in testing.

This board was mainly developed to be a drop-in replacement for the Creality Ender printers. All the connectors and peripherals, even the shape of the PCB, was designed in such a way, that it can be easily replaced in those printers. This, however, isn't a limiting factor when using this board in a custom built machine. The only downside that comes to mind, are the limited options for an LCD screen, it has only a few supported variants. But in this build, no screen was planned to be used, because of the capable web UI.

There was a plan to use a BL-touch sensor for automatic bed leveling, since it is supported by the board, but after manually calibrating it using screws on the corners, the leveling seemed good-enough for the time being. Only after further testing was it discovered, that the bed deforms significantly when heated to higher temperatures. The late discovery was caused by the low priority of testing different printing materials. After that, a BL-touch sensor was tested, but seemed to be faulty, and an alternative has not been acquired.

3.5 Firmware and initialization

The official guide available at [12] was a complete and very useful resource in this process. However, there are multiple ways to go about implementing this firmware on a standalone computer.

3.5.1 Firmware setup and implementation

Note that there is only official support for the Raspberry Pi 2 and newer. It is possible to make this firmware package work on other single-board computers, such as the BeagleBone, but it may require more Linux knowledge. In this case, a Raspberry Pi 4 was used. The official guide instructs to first create an SD card that has a bootable instance of OctoPi. Which is a software suite developed to control 3D printers that run other firmware. It is a very handy tool that connects to a working printer via USB and provides the user with a lot of extended functionality with a web interface. However, it is only a software to control a functioning printer with its own firmware, it is not a substitute for it. Klipper utilizes these functions, but is not dependent on them.

OctoPi is built on Raspbian, which is a lightweight fork of Debian Linux specifically modified for the Raspberry Pi. Creating a micro SD card with a bootable instance of this software solution is as simple as clicking a few buttons in a tool called Raspberry Pi Imager. Note that at this point, the user either has to specify a network connection and enable SSH communication, or has to have a monitor and keyboard ready. After inserting the created microSD card in the Raspberry Pi and providing power, the system will perform a first-time bootup. At this point, the OS should be running, including the OctoPi services.

Now, the Klipper firmware services can be installed. Internet connection is mandatory for this step. The process starts with cloning into a GIT repository that contains all the necessary files. After that, an installation BASH script can be executed. At this point, the Klipper host service should be up and running.

Klipper firmware consists of two parts, the second being the client that runs on the controller of the printer itself. At this point, a client firmware binary needs to be created. For this, the host part has a script ready to do just that. After executing it on the Raspberry Pi, a simple command-line menu will show up:

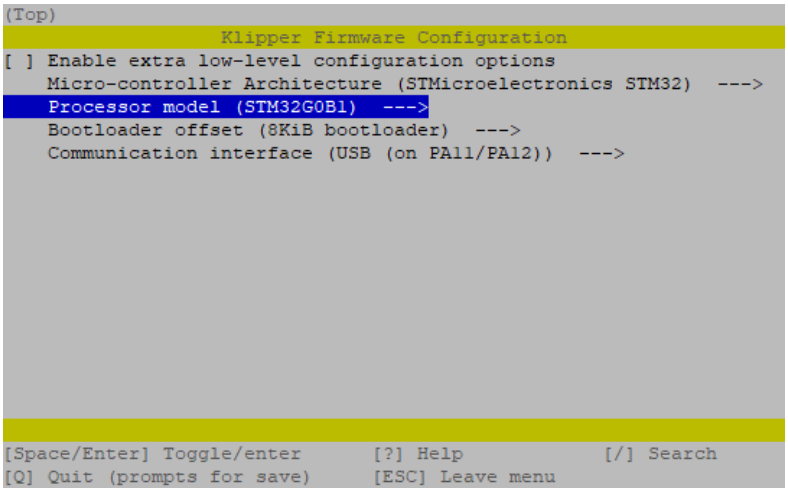


Figure 3.8: Firmware binary creation menu

Important specifications about the micro-controller need to be specified in this menu. These can be either obtained from configuration references on [12], or in datasheets provided by the manufacturer of the controller. After doing so, there are two possibilities. Some controllers can be flashed directly via the USB connection between the Raspberry Pi and the controller. If this is the case, communication needs to be established first. With the controller powered up and connected, its device ID should show up. With that, a serial communication can be used to flash the firmware binary directly. Otherwise, the binary file is created using another script, and then manually copied to another micro SD card that will be used as a bootable media for the controller.

This is where certain struggles occurred. A quality shielded USB cable, preferably of short length, is recommended here. And even with that, several problems can occur. In this case, the controller did not show up in the list of devices at the host. After lots of experiments, it was found that the controller failed to boot the client firmware binary. When dealing with low-level software, there are certain situations that can be confusing and hard to diagnose. When programming something in an IDE, the software usually shows where the problem may be and what could be the cause. Not in this case, the only feedback that was available was a little red light that indicates whether the controller has power. It was later found, that not only was the micro SD formatted using an unsupported file-system, but even the binary file had the wrong name. After correcting these two mistakes, the controlled did show up in the device list. But it's ID wasn't looking as it should have. After some more struggling, it was discovered that there was a baud-rate mismatch with this specific board, what resulted in the communication failing. But in the end, after changing this, connection was finally established.

Alternative method

There is another way of installation, available from [20], named KIAUH. An acronym for *Klipper Installation And Update Helper*. It is an all-in-one script, that offers multiple software solutions, as showcased on the left side of 3.9. The prerequisites only differ in the main OS. While the official Klipper installation is recommended to be done on OctoPi, KIAUH recommends just Raspbian as the base. After cloning into the GIT repository and executing a script, an intuitive menu shows up.

Figure 3.9: KIAUH script software and menu. Source: [20]

This tool comes helpful when installing more software, or updating the already installed. While Octoprint, the web service of OctoPi, is a very helpful UI, it doesn't maximize the capabilities of Klipper. For that, two popular options are present, Mainsail and Fluidd. Both of these solutions rely on the Moonraker service, that handles the back-end of the web applications.

Fluidd

In this build, both were tried, but Fluidd was chosen to be showcased here 3.10. This UI provides the user with a lot of useful information and tools such as, current print status, manual control, file management, command window, macro buttons, webcam view and more. Fluidd has more tabs for even more functionality. It can manage system updates, edit printer configuration and other files, show systems statistics, and more. Such a setup drastically changes the interaction with a 3D printer. This is especially refreshing if the user is used to a classic printer that only has an LCD display with an SD card or USB slot. With those, each gcode must be manually copied to a portable media at a computer, then transferred to the printer physically. With Fluidd and Klipper, sliced gcode can automatically be sent over to the printer and started. The web interface can even be port-forwarded, with little networking skills. This then allows the user to control and oversee the printing process from anywhere.

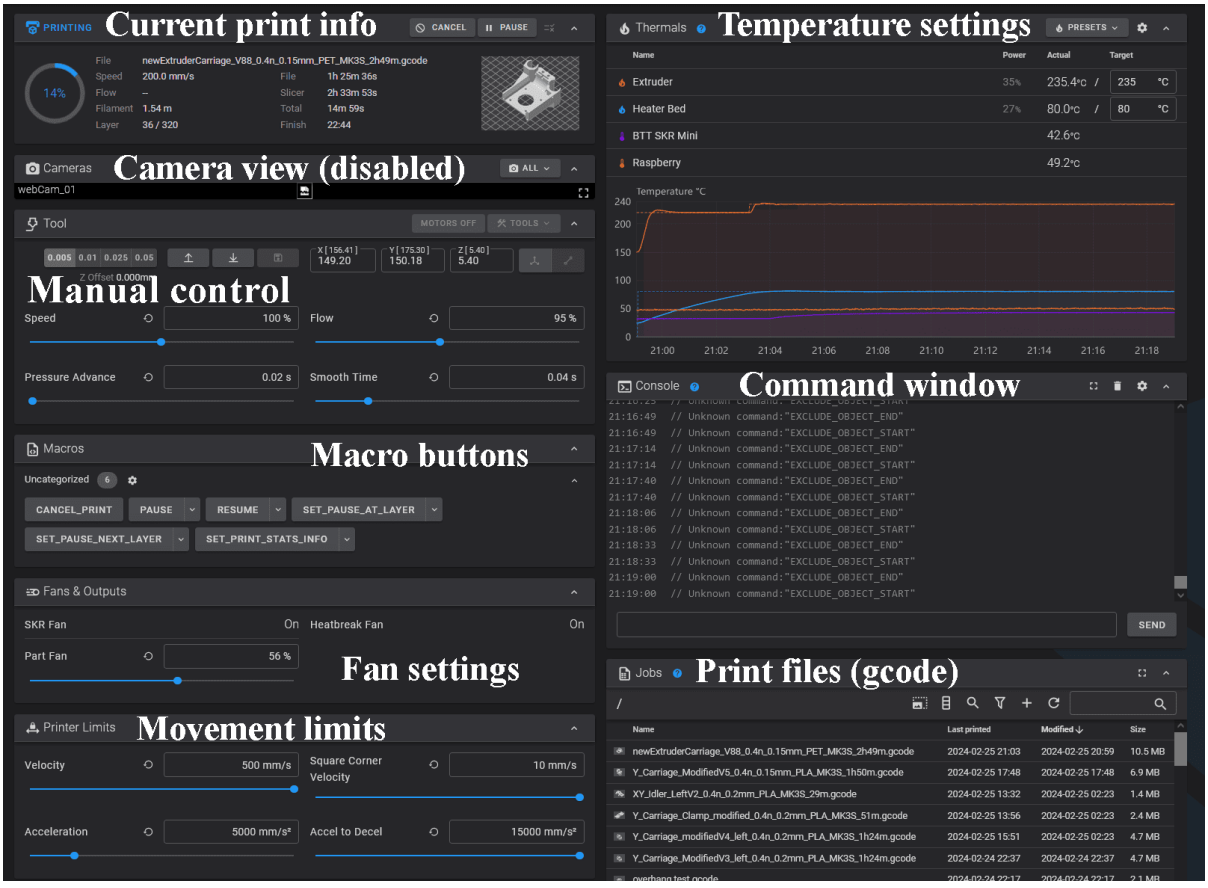


Figure 3.10: Fluidd user interface

3.5.2 Initial printer parameters configuration

Another advantage of using Klipper firmware is in the ease of editing a printer configuration file. In some other firmwares, this process involves re-compiling the binary and re-flashing the controller. Here, only a text file needs to be edited, what can be done right in the web UI. After that, with a click of a button, the controller is reset and the new configuration is applied.

When creating this file for the first time, a general reference configuration is used, available from Klipper's Github repository [11]. It contains all the necessary pin mappings for this particular controller. As of writing, the configuration file for this machine extends to more than 200 lines, only a few key parts will be discussed next.

First, the kinematic platform and motion limits had to be defined. It was decided that velocity and acceleration control will be managed mainly from the slicer profiles. Therefore, intentionally high values were used in the configuration file. Next, the motors themselves, needed parameters to be set. Respective rotation distances for the axes were calculated using the equation in 3.5. These were identical for the X and Y axes, but a significantly lower value was used for the Z-axis, because of low-pitch leadscrew in use. The extruder rotation distance was initially only guessed, since it cannot be measured easily because of the meshing of the gear on the filament.

$$rotation\_distance = \pi \cdot d = \pi \cdot \frac{belt\_pitch \cdot teeth}{\pi} = belt\_pitch \cdot teeth \tag{3.5}$$

Motor driving parameters

Next were driver-specific parameters. The run currents were set according to the rated currents in the stepper's datasheets. These values are provided in RMS(root-mean square). As mentioned before, this wasn't the case for the Z-axis. Here, the max recommended value of the driver was used. This was still an undershoot for the combined rated currents of these motors. After that, interpolation and micro-stepping needed to be specified. Interpolation being a TMC specific parameter which enables internal stepping, in combination with the chosen micro-stepping value, to decrease load on the controller's CPU. But since Klipper is efficient with stepping-rates and the CPU of this controller is a capable 32-bit chip, it was decided to disable this function. Klipper's documentation explains, how interpolation introduces a small positional deviation, which was rather to be bypassed. Micro-stepping was more of a dilemma. But for initial testing, a value of 32 was selected for both the A and B motors This value is a middle-ground between smooth and silent operation, and reasonable torque output. The micro-stepping was set to 1/16 for the Z-axis and extruder. These axes have lower rotational distances and torque output was more important. For the X-axis exclusively, a diagnostic pin needed to be set as a prerequisite to enabling virtual endstop. When used, the driver sends signals to the controller, from which it calculates whether the motor came to a stall. As discussed in 2.2. For this, a threshold value needs to be set that specifies how aggressively the motor will bump into the mechanical limit of the axis. It was initially set low, because the recommended method is to tune it in an iterative way. But to make the axes home at all, the homing directions and speeds had to be set. For this, it was easier to just power on the printer and do an experiment, instead of analytically determining these directions.

Heating, cooling and others

Other crucial initial parameters were more straightforward to determine. Both of the heaters needed their thermistor sensor types to be set, according to their model numbers. As well as their respective minimum and maximum temperatures. Since the hot-end in use, is an all-metal type, temperatures can go as high as 300° C. As a bonus, the controller and host CPU temperatures could also be included in the configuration, making them visible in the web UI. This could potentially come useful when the printer would be upgraded with an enclosure. Since its point is to trap heat inside of the printer, testing would need to be conducted to make sure that none of the electronics are overheating.

Fans only need very few parameters, such as their respective pin, max duty cycle, and triggering condition. The part cooling fan is operated manually with gcode commands, this is determined by the sliced gcode. But the heat-break and controller fan use a trigger to turn on. Since most of the heat on the controller board is generated at the stepper drivers, the triggering was set so that the fan starts, whenever a stepper is active. For the heat-break, a minimal hot-end temperature was defined.

Other parts of the configuration file include non-critical settings, such as the use of gcode arcs, macros for starting or ending a print job, heater timeouts and others. Some examples of different parts of this file can be seen on 3.11

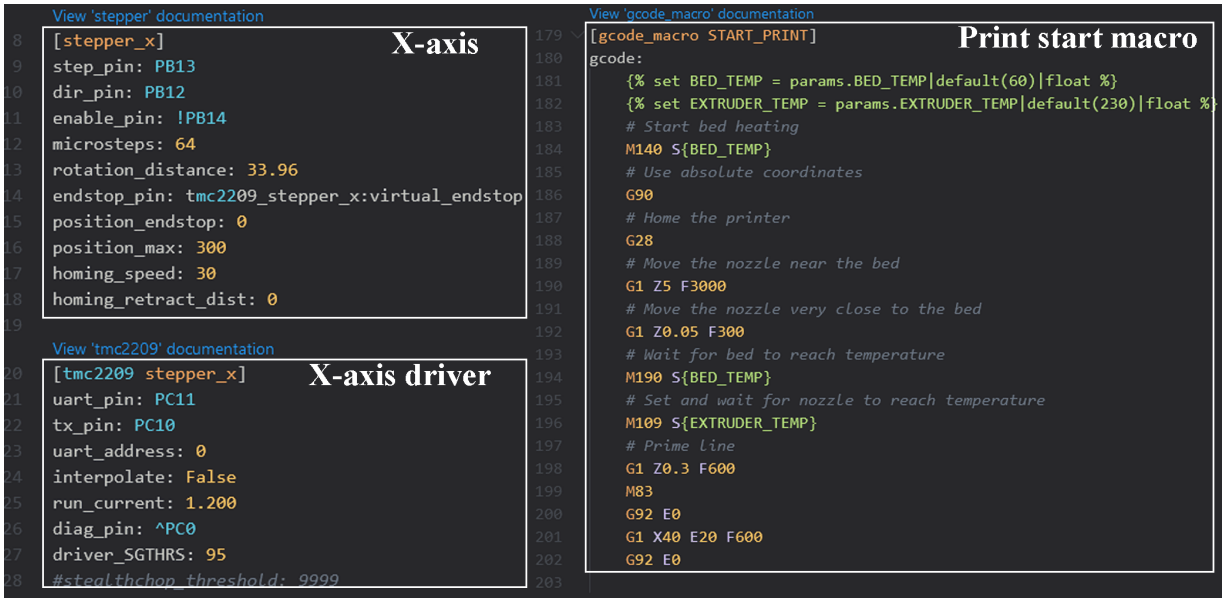


Figure 3.11: Printer configuration file snippet

4 Tuning and functions testing

Proper calibration and testing of defined settings and parameters is one of the most important elements that has to be done right to get great printing results. It is a very different process for a printer that comes as a kit, with predefined settings from the manufacturer, than for one built from scratch. Some calibration procedures are simple, but others, such as creating printing profiles can be a very tedious process to get right. Before even attempting any printing, there are a few procedures that had to be done, those will be discussed next.

4.1 Calibration and testing

Both of the heaters, will be controlled by a PID algorithm. In this printer, the heated bed is connected to an external MOSFET, that has no problems with high frequency switching. The PID tuning is a very simple and straightforward process in this case. There is a dedicated command that automatically tunes the necessary parameters for a specified target value. This target temperature is the only thing that the user needs to define. For the hot-end and bed respectively, 230° C and 80° C were chosen. These values are somewhere in the middle of the range of the most used temperatures. After executing the command from the web UI, the corresponding heater starts heating and the auto-tuning algorithm measures the rise-time and overshoot. Based on these measurements, it adjusts the P, I, and D parameters and then tries heating again. After 6 cycles, it responds with the final parameters, which can be then defined in the printer configuration file.

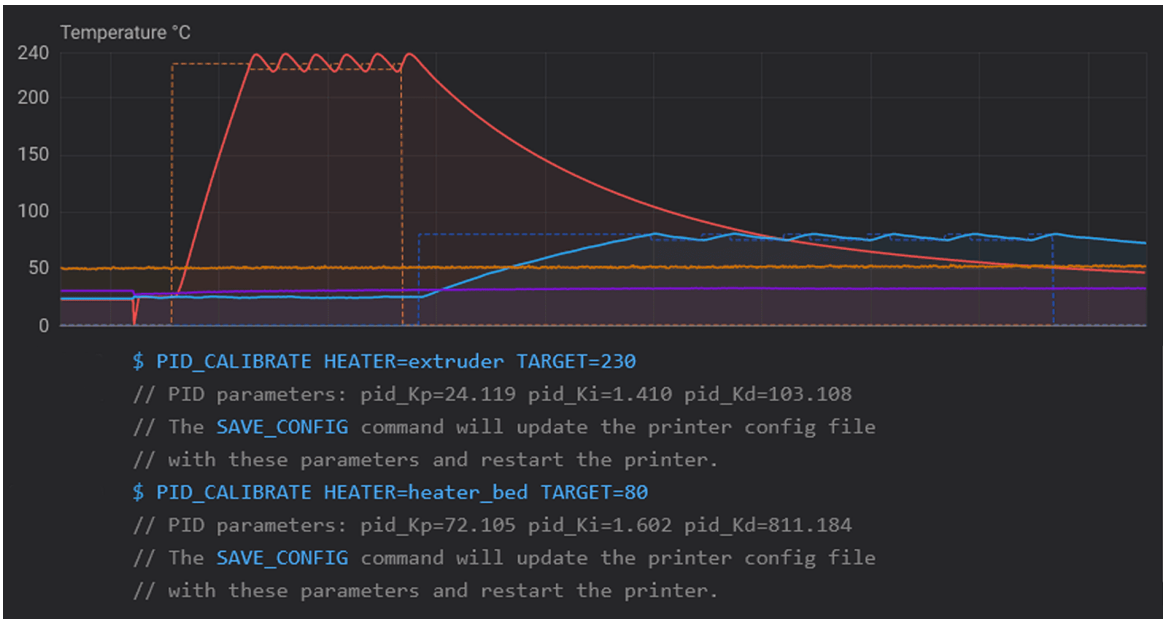


Figure 4.1: PID auto-tuning temperature graph and commands

Extruder calibration
As mentioned in 3.5.2, the rotation distance of the extruder needed tuning. It is recommended to use a manual method for adjusting the rotation distance. Since the extruder uses a gear and a bearing to squish and push the filament, it is hard to analytically calculate the effective driven radius. It is a better approach to insert the filament and command an extrusion move to measure the actual extruded length. Note, that after doing such tuning, the pressure on the bearing that is set by a screw, needs to stay the same. Otherwise, re-tuning might be needed. It is also important for the commanded movement to be relatively slow, otherwise the filament may not be able to melt quickly enough, and cause back-pressure that would make the measurement unusable. In reality, this test is easily conducted by using a light colored filament and a black marker. The filament gets inserted into the gears and a mark is created at a chosen length from a reference point. After that, the extrusion move is completed, the new distance between the mark and the reference point is measured. The ratio between the actual and requested extrusion length is a linear scaling factor for the rotation distance.

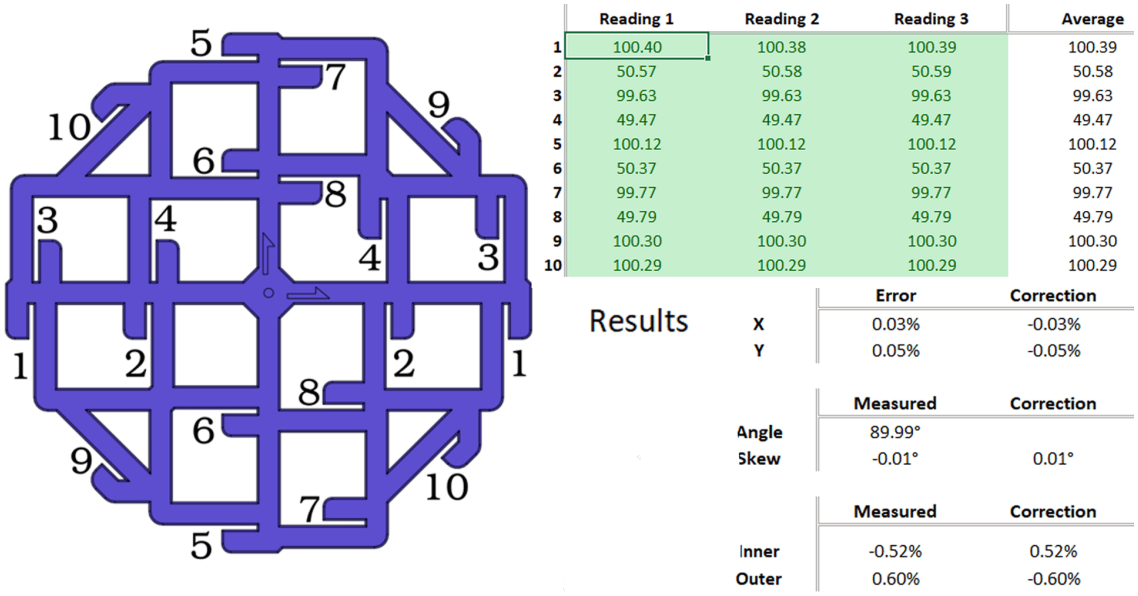
Motion abilities
The acceleration and speed limits of the XY motion system could be tested. Only the absolute limits were in question at this initial calibration phase. That is, the acceleration and speed, at which the motors start skipping steps. At regular printing operation, running at such speeds and accelerations may not be possible, but it was important to first know the capabilities. For this, a manually written gcode was created, which moves the print-head in the X, Y, and diagonal axes. Such a test essentially corresponds with the plots showcased in 3.3. The speeds and accelerations were increased in an iterative manner. Since a general torque characteristic of a stepper motor is expected, the gcode first increases speed and then tests all accelerations defined in a range. At some point, failure is expected by skipping of steps.

This gcode was run with different ranges of values for both the the regular and silent chopper mode set on the stepper drivers. As expected, the silent driving mode started skipping steps at a lower speed than the regular mode. After closely observing the process, maximal speed and acceleration values were obtained, which could be used as a starting point for creating printer profiles in the slicer software. Of course, it is wise to not use these values for real printing. While the printer was capable of achieving them, during a printing process, the print-head can bump into molten plastic or encounter situations where more load is applied. That would lead to skipped steps, so it is advised to stay a considerable amount below these values.

At this point, the printer was ready for a first print test. For this, a initial slicer profile was created with very conservative values. Slow printing speeds, low acceleration, common layer height and so on. After more testing, multiple slicer software were tried, but the final choice was OrcaSlicer. It is a fork of PrusaSlicer and Bambuslicer. Their names are derived from their respective company names, which are very popular in the 3D printing community. After making sure the bed is leveled, using screws on it's corners, a 3D model needed to be sliced using the conservative settings.

Dimensional accuracy

A common choice for a first print is a so called *Calibration cube*. It is a simple cube with 10 mm long sides that has axis names engraved on its faces. While it is not a bad model per se, measuring the sides of the cube is problematic. One of the problems is the squishing of the first layer, the other are the sharp corners. When the print-head prints these corners, it needs to decelerate, change direction, and accelerate. Unfortunately, the filament that is being pushed does not extrude with surgical precision because of the pressure that is created in the melt zone. This effect will be more discussed in 4.3. Because of this, the model hasn't been used. Instead, a very clever model from a creator named Vector3D, the CaliFlower was chosen [21]. It is a carefully designed model, specifically for calibrating dimensional accuracy and skew of a printer. It comes with an Excel document that serves as a tool to make the process even more user friendly. This can be seen in 4.2.



	Reading 1	Reading 2	Reading 3	Average
1	100.40	100.38	100.39	100.39
2	50.57	50.58	50.59	50.58
3	99.63	99.63	99.63	99.63
4	49.47	49.47	49.47	49.47
5	100.12	100.12	100.12	100.12
6	50.37	50.37	50.37	50.37
7	99.77	99.77	99.77	99.77
8	49.79	49.79	49.79	49.79
9	100.30	100.30	100.30	100.30
10	100.29	100.29	100.29	100.29

Results

	Error	Correction
X	0.03%	-0.03%
Y	0.05%	-0.05%

	Measured	Correction
Angle	89.99°	
Skew	-0.01°	0.01°

	Measured	Correction
Inner	-0.52%	0.52%
Outer	0.60%	-0.60%

Figure 4.2: CaliFlower model and calculator sheet

What cannot be seen from this top view of the model, is that the upper and lower edges of the model are all chamfered. This eliminates the problem of squishing that occurs at the first layer that contacts the printing surface. A well designed feature are the outer and inner dimensional measurements, which not only eliminate the impact of over or under extrusion, but show by how much it is offset. And the edges at the corners, which form a diagonal measuring point that serves for calculating the skew of the axes. With the measurement seen on 4.2, it can be observed that there was a negligible amount of skew, almost no error in the averaged X and Y dimensions, but considerable over-extrusion took place. This is also seen plainly from the measurements, all the inner ones are smaller and all the outer ones are bigger than the reference dimensions. Over-extruded lines tend to be pushed outwards from the printed object, what makes this explainable. But what may be less intuitive, is how the extrusion rate can be inaccurate if it was calibrated with the method described earlier. Well, measuring a long piece of curved filament and marking it with a sharpie can only go so far with precision. Also with that method, the extrusion rate is steady, which makes the pressure in the melt zone consistent. This is never the case in actual printing.

Material flow rate

When tuning a printer for fast printing, it is important to know, how fast the filament can be melted for extrusion. This depends on several factors such as the extruder design, nozzle size, and most importantly the length of the melt zone. This printer uses a *Volcano* style hotend from E3D. It is a variation of their V6 hotend, but with a longer melt zone. This generally allows for higher flow rates because the extruded filament has more time to soak up heat.

Two different methods have been tested. One prints a thin-walled object at increasing speeds until failure. While this method is closer to a real printing scenario, it proved to be difficult to cool the layers fast enough. This caused problems when trying to evaluate the point of failure, when the maximum flow rate would be reached. Instead, the second method was used to create the graph 4.3.

An online tool was used to create custom gcode, that prints blobs of plastic at increasing speeds. Three materials were tested this way, PLA, PETG, and ASA. Different gcodes were created for each of them, because their recommended print temperatures weren't identical. After the gcode has finished, each blob of plastic has been measured with a micro-scale. The code took into account the different densities of these materials, so that each blob would be exactly one gram in weight. The point of this test was to see, what the actual weight would be, at higher flow rates.

Test prints showed that it is acceptable to use a flow rate that had a 10% drop in actual flow in this test. With such a setup it was about $20 - 25\ mm^3/s$.

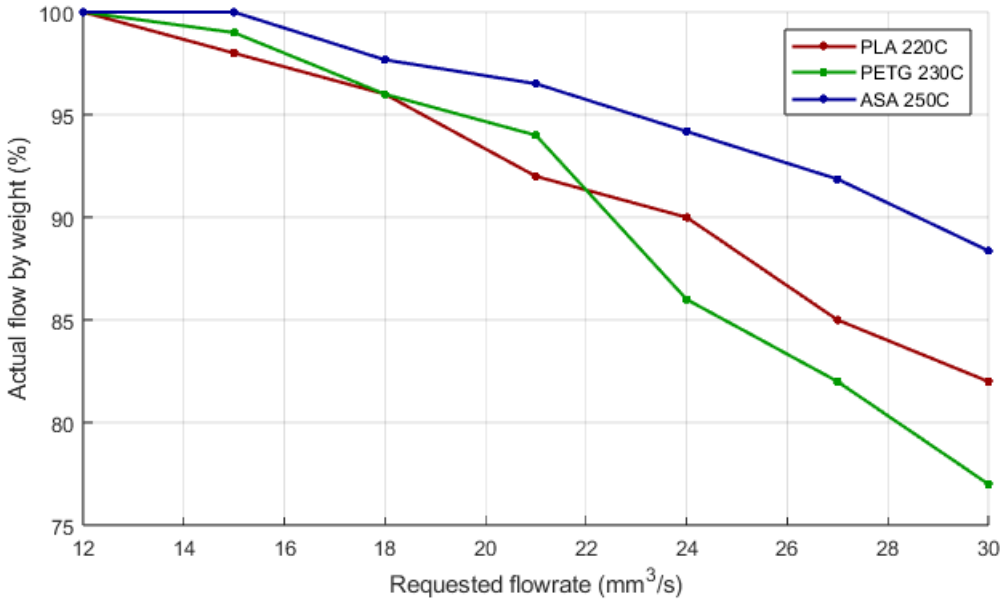


Figure 4.3: Flow rate test

4.2 Input shaping

To take advantage of a robust frame and capable motion system, input shaping had to be set up. When using Klipper firmware, there are two ways to find the necessary parameter, the resonant frequency. Klipper has built-in support for 5 shaper algorithms. But first, measurements must be taken to enable them in the printer configuration file.

One approach is based on visual inspection of a particular print. A special thin-walled model that has sharp corners and notches, is first sliced in the slicing software of choice. Then, a command is sent to the printer that gradually increases acceleration during this print. It can be inspected after finishing. The thing to look for is the ghosting effect, as mentioned in 2.4.1. This method relies on measuring the distance between the artifacts that are created after passing a sharp corner, or a notch in the model. The drawback of using this method to determine the resonant frequency is the inaccuracy of manual measurement.

Resonance measuring setup

If an accelerometer is available to use, a better method can be used with a little more setup. In this build, an ADXL345 sensor was first connected to the Raspberry Pi. During this process, the Klipper documentation proved to be a very useful guideline. After soldering and connecting the wires according to the schematic, a few lines had to be edited in the printer configuration file. The sensor type and other parameters were set. After double-checking the connections, the sensor was queried with the help of a command. A report of acceleration values showed up in the command window, but it was obvious that the values were unrealistic. This issue needed a considerable amount of troubleshooting to resolve. The problem was the use of simple, small cross section wiring. The length between the Raspberry Pi and the accelerometer was only about one meter, yet the noise caused by this wiring had a significant impact on the reported values. A shielded Ethernet cable with the connectors removed was used. They have 4 twisted pairs, each with its own shielding. After replacement, the sensor reported expected values and the noise levels were in the normal range. At this point, the sensor could be mounted on the print head, an illustrative render can be seen in 4.4.

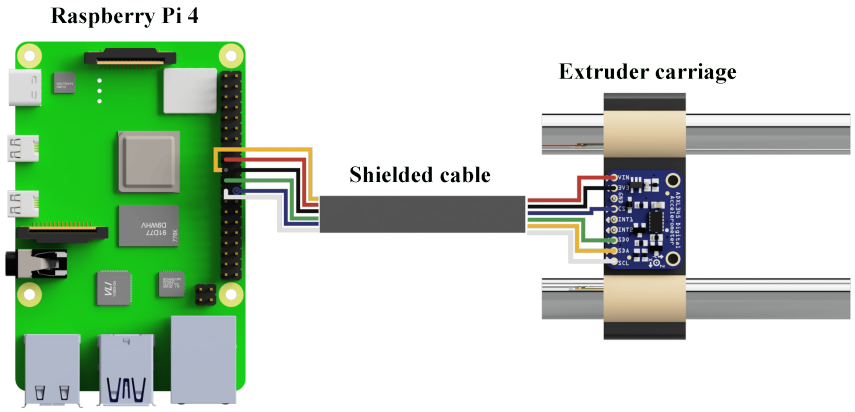


Figure 4.4: Render of resonance measuring setup

Frequency response and shapers

After making sure the sensor was reading correctly, the resonance measuring command could be issued. Since this is a CoreXY printer, the X and Y axes are tied together. This way, the accelerometer is attached to the print head for all the measurements. The command takes two input arguments. First is the axis, it can be X, Y, or even a custom axis by entering a vector. Second is the position where the measurement will be taken. The latter must to be specified in the printer configuration file, the default is the center of the printing area. First, the X and Y axes are measured. When the command is executed, the printer first homes the axes and then moves to the defined position for the measurement. Once there, it vibrates the print head in the defined axis with an increasing frequency. The default sweep range is 0-133 Hz.

When the frequency sweep is finished, a CSV (comma separated values) file is created with raw data. Then a Python script is used, which is included in the Klipper scripts folder on the Raspberry Pi. This script processes the raw data and creates a figure 4.5.

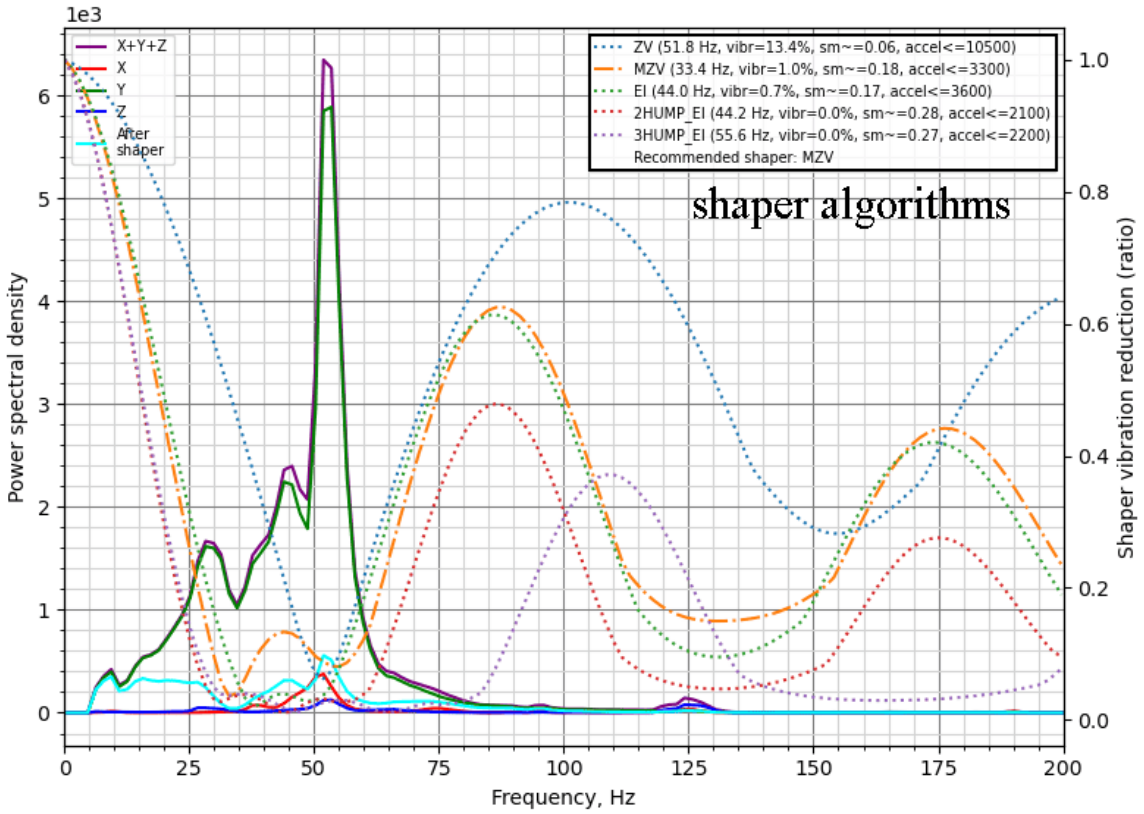


Figure 4.5: Frequency response from first measurement

In this figure, the X-axis is the frequency, and on the Y-axis a calculated PSD (power spectral density) is shown. In the upper left corner is a legend that describes the axes of the accelerometer. It is not necessary to align the axes of the sensor with the printer's. In the upper right corner, the legend shows information about how the 5 different shaper algorithms would dampen the measured vibrations.

Frequency response as a diagnostic tool

Such a graph can be useful for determining more than the just the resonant frequency. Otherwise, the script could just show the value of the frequency at which the most dominant peak was measured. Progression is also useful. In the best case scenario, a single clean and dominant peak on the PSD graph would be expected. But most of the time this is not the case. Especially with custom built printers that use untested parts that have imperfections. This can be seen to the left of the dominant peak in 4.5. The rising edge contains several smaller peaks, that could indicate multiple issues. In this case, after some troubleshooting, it was found that the linear bearings were mounted with excessive pressure. This caused increased friction of an uneven nature, which showed up when the print head was vibrated at lower frequencies. After correction, a nicer response was measured, as seen in 4.6.

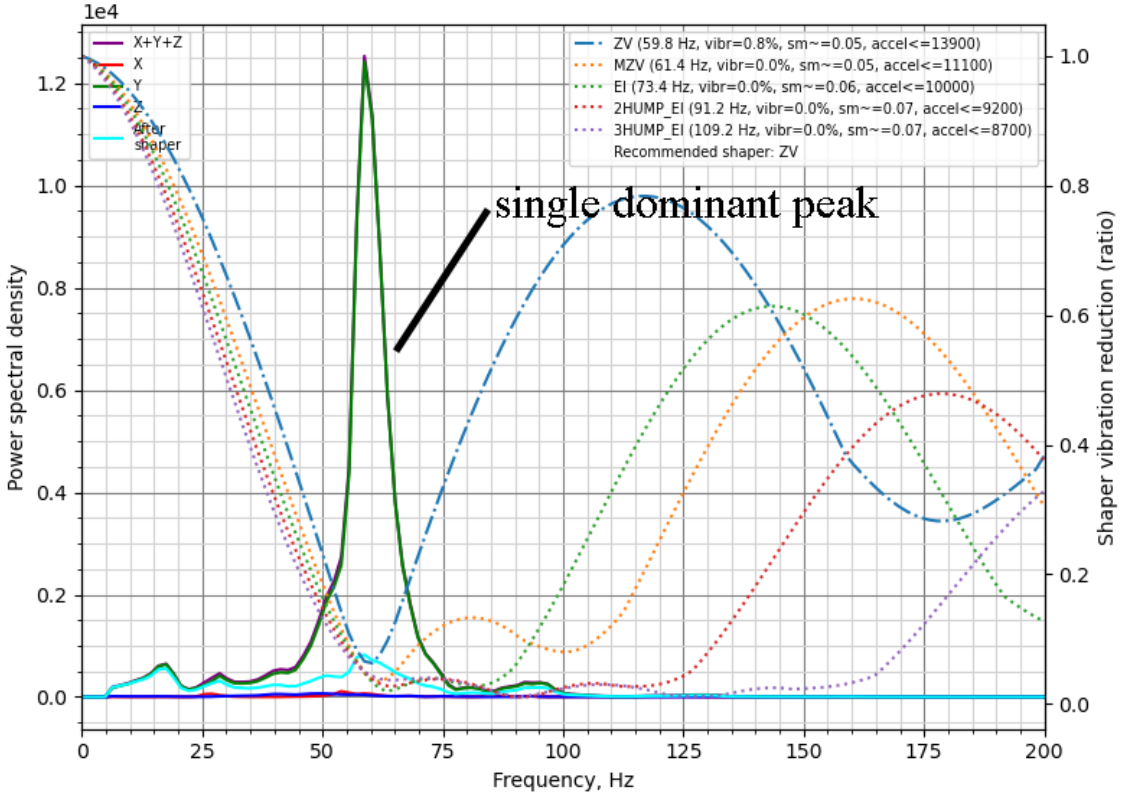


Figure 4.6: Frequency response after corrections

This process was a clear demonstration of how measuring the resonances with an accelerometer setup can be useful. There is a reason for correcting mechanical problems this way. The next paragraph will discuss in more detail, why it is important to have a single dominant peak.

Shaper algorithms

The legend in the upper right corner of the generated graphs show several estimated values. Each shaper algorithm, visualized by a dotted or dash-dotted line, operate in different ways. Their purpose is to try and dampen prominent vibrations. Their respective lines show a frequency dependent damping ratio. It's scale is shown on the right border of the graph. This ratio has a range of 0-1. In general, prominent vibrations need to be dampened by shaping the motion input by this algorithm. But by doing that, a side effect called *smoothing* is introduced. It is shown in the brackets for each shaper algorithm as 'sm'. It is a dimensionless value that is calculated as an estimation for a default acceleration value in the Python script. Next to it, an estimation of the peak residual vibration is shown as a percentage. This estimation is important, it tells how much of the un-shaped vibrations will reside after applying the shaper algorithm with the respective resonant frequency. A comparison of shaper parameters from the first and second measurement can be seen in 4.7.

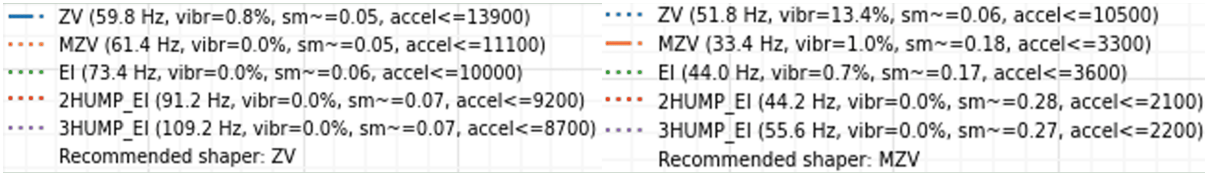


Figure 4.7: Comparison of shaper algorithms with parameters

On 4.7, the right side legend is for the uncorrected mechanics, the left is for the corrected. The dash-dotted line marks the algorithm that is recommended by the Python script. Comparing the two, it can be seen that the corrected one has significantly less estimated residual vibrations, with less smoothing. These vibrations will not only have an impact on the print quality, but also affect the mechanical parts. When high accelerations were tested without input shaping, it was found that some of these vibrations could cause the whole frame of the printer to shake. At one point, even the print head carriage broke in the middle of the test. These experiences underscore why it is important to fix any mechanical issues that are revealed by these measurements, before attempting to use high accelerations.

When a dominant peak is measured in a well assembled and designed system, the ZV algorithm produces the best results. It has the narrowest damping ratio curve, minimizing residual vibration and smoothing at the same time. However, if the system has unresolved problems, or is designed in a sub-optimal way, other algorithms might be appropriate. In the legends shown in 4.7, they are ordered by their damping spectrum. The one with the widest spectrum is 3HUMP EI. Looking closely at the graphs, it can be seen that it dampens vibrations over a very wide range. While this ensures low residual vibration, it also introduces more significant smoothing. And smoothing increases with higher accelerations.

Equal belt tension

In CoreXY machines, it is important that the belt tensions for the two loops running around motor A and B are the same. The tensioning method described in 3.4.1 is not the most accurate. It relies on measuring the frequency of the sound waves that are created by plucking the belts. A smartphone was used for this method, and it's measurements have had a deviation of a few Hz. To correct this, the resonance measurement command can be issued again, but this time with an input of a vector that defines the diagonal axis. In this direction of motion, the two loops of the belts are strained in a back and forth manner. The output data is then processed by another Python script, but this time, it only shows the PSD graph, without any shaper algorithms.

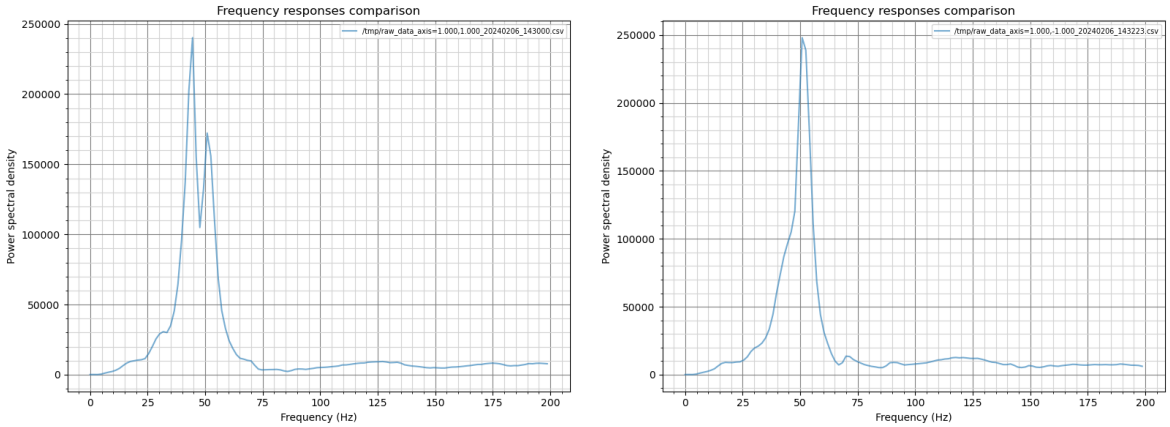


Figure 4.8: Frequency response for diagonal axis

On the left side of 4.8 is the first measurement after tensioning the belts with the method described in 3.4.1. It can be clearly seen that there are two peaks in the PSD graph, indicating two distinct frequencies that the system is responding to. After adjusting the respective belt tensions carefully, another measurement was made. This can be seen on the right side of 4.8. A single peak here confirms that the belts are now equal in tensioning force.

Measuring position

Later on, measurements for the X and Y axes were repeated, but this time in different positions. A slight deviation in resonant frequency was observed. It is difficult to explain this behavior with certainty. But only one frequency can be used to define the input shaping in the printer configuration. Therefore it is probably for the best to use the values obtained from a measurement that was made in the center of the build volume.

4.3 Pressure advance

As discussed in 2.4.2, pressure advance is a very useful printing function that ensures more precise filament deposition. This is especially important when printing at high speeds. When high flow rate is used, the pressure in the melt zone becomes more significant.

In Klipper's pressure advance function, two parameters need to be set. The first one being the time. This essentially serves as a time window to quickly increase or relieve pressure. The second is the smoothing time, this defines how quickly this action can be performed. To determine the first parameter, a visual tuning method is used. A hollow object with various sharp corners is printed at high speeds, with an increasing time parameter for the pressure advance. The time is increased by a defined step at each new layer. After the print is finished, the object is examined. Then from the height of the layer that has the best-looking corners, the time parameter can be found.

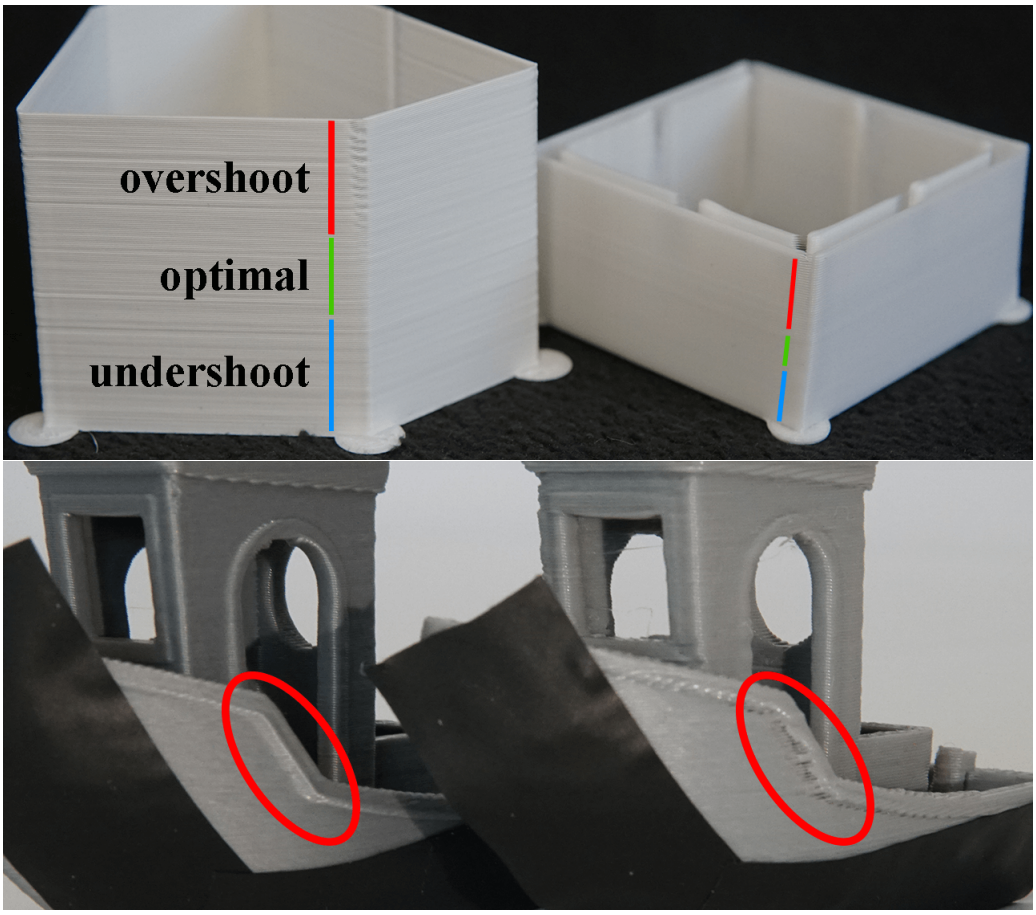


Figure 4.9: Pressure advance, a) tower model, b) effects on sharp corners

Note that this parameter is dependent on the nozzle size and even material choice. Most of the times it is suitable to leave it unchanged when using similar filaments. However, a nozzle change can have a bigger impact and re-tuning might be needed.

5 Summary of build and comparison

After successful calibration of the printed was achieved, several slicing profiles were created to test real-world capabilities of the printer. Note that appearance and ease of use were not a priority. This printer was built only for me, with function and performance being the most important.

5.1 Summary of build

All the specifications listed in 3.1 were fulfilled. A cost-effective, fast CoreXY printer, with a relatively large build volume was built. It is a useful and capable tool for rapid prototyping. However, further improvements have to be mentioned.

First, the implementation of automatic bed leveling would be beneficial. While printing with materials that only require low bed temperatures, the absence of this feature is negligible. However when ASA was tested, which requires 110° C on the bed, it was discovered that it deforms significantly. Probing the surface with a leveling sensor could easily solve this issue.

Another feature that would be useful when printing warp-prone materials would be an enclosure. An experiment was conducted with a few sheets of plastic on the sides of the printer. Even such a basic setup improved layer adhesion and reduced warping significantly. However, in one of those experiments, the motors started overheating. This could be fixed by the addition of coolers, or by lowering currents while using slower printing profiles.

To make the printer even faster, more weight reduction would be needed. One option is to change the direct-drive extruder for a bowden type. This would make the print-head assembly significantly lighter and that would allow for higher accelerations. However, if the extruder would be kept as is, the XY assembly could be made lighter. By redesigning the XY connecting parts and print-head carriage, 12 mm hollow aluminium shafts could be used. With such a change, it would also be interesting to test wider belts. They should change the dynamic properties of the motion platform and may contribute to a better frequency response.

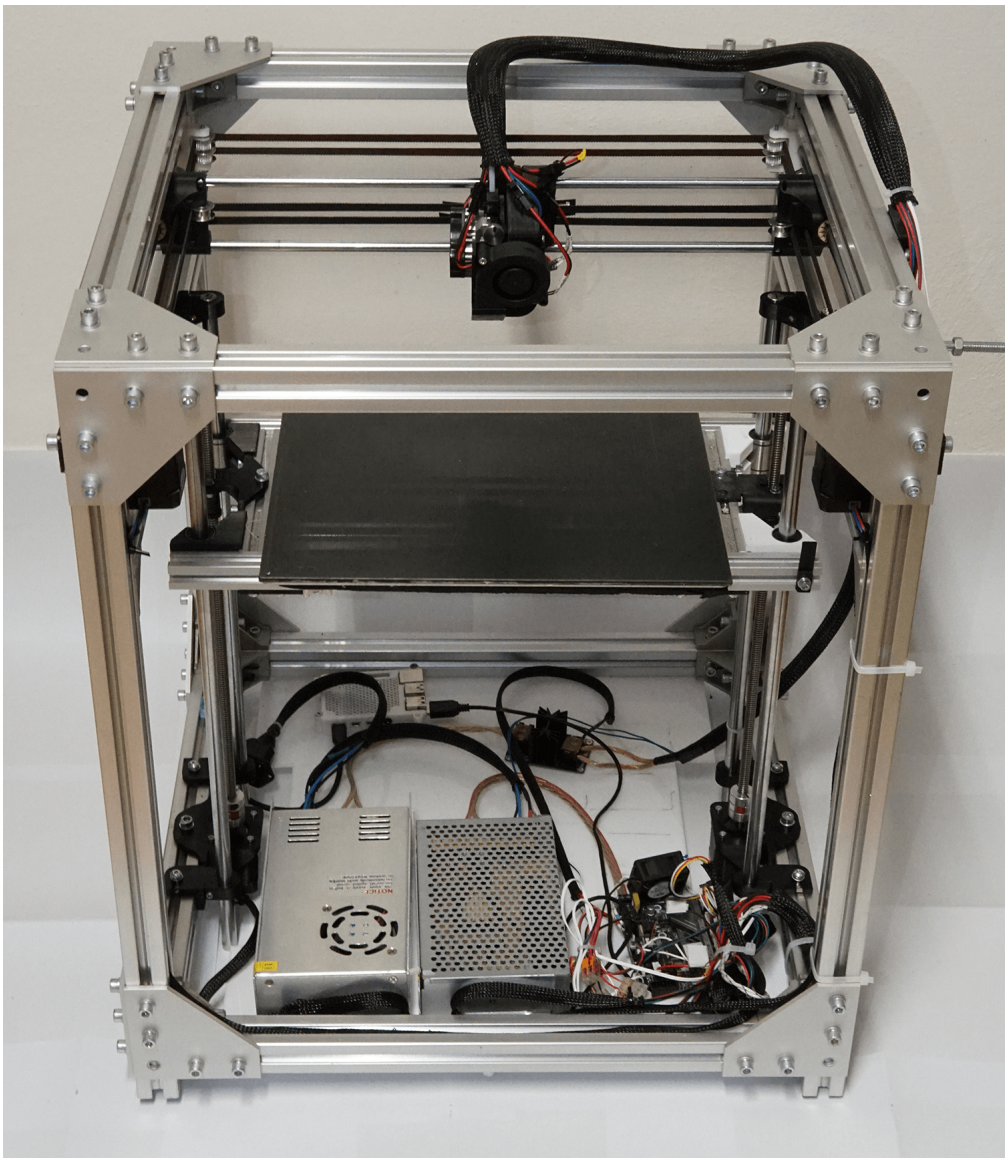


Figure 5.1: Photo of the completed printer

It has to be mentioned that the honeycomb-shaped undertray for the electronics was not finished in time. The first version proved to have insufficient stiffness to hold the heavy power supplies. And as of writing, the second version wasn't printed or mounted. Until this is fixed, a white acrylic plate is used to hold the electronics in place.

5.2 Comparison

For the comparison, an unmodified Prusa MK4 with input shaping was used. It is true that it uses a different motion platform, but because of its popularity, it was the easiest to obtain. The more important property is that these printers are very well tuned and therefore can be used as a reference for a comparison.

Both machines were using fast printing profiles with the same settings. Since this is a custom printer, these default printing profiles were created by me. Speeds, accelerations, and various printer specific settings were different from the ones that the Prusa uses.

3DBenchy
To make the comparison fair, the settings that directly affect the properties of the printed object were kept identical :

- Same material (high-speed PLA),
- 0.2 mm layer height,
- shell with 2 walls,
- 10% infill density,
- 0.6mm top and bottom.

These setting were used to print the infamous 3DBenchy model. Available from [22]. It is a challenging print which tests several areas of printing quality.

Figure 5.2: Front view of printed 3DBenchy

Under good lighting conditions it can be observed that the quality of both parts are acceptable. Overhangs, sharp corners, bridges and the thin chimney printed well on both machines. However, the Prusa performed better in layer line consistency. The walls of the print are more aligned.

The back side of the benchy shows other areas to compare. It may be hard to see from the picture, but the small text on the hull is visible and readable on both of the prints. The window and other edges came out nicely on both of them. But the finish of the top surfaces was better on the Prusa. This could have been tuned with some changes in the printing profile of the custom printer.

There is also a very slight stringing issue that an be spotted on the print of the custom built printer. This wasn't resolved with more filament retraction, so it may be caused by the combination of the material, nozzle and printing temperature.

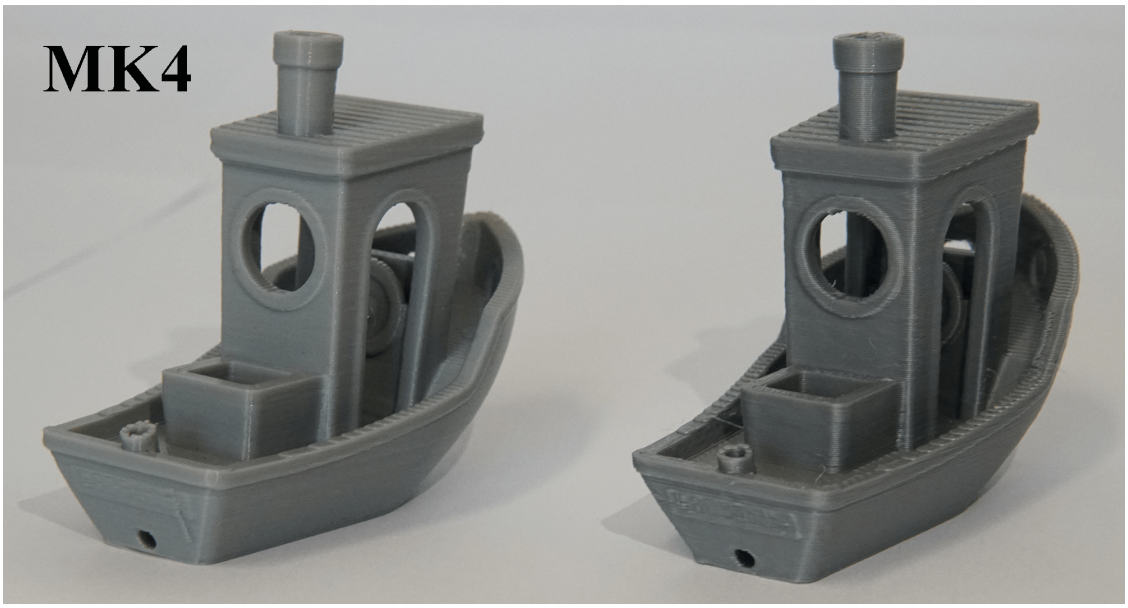


Figure 5.3: Back view of printed 3DBenchy

One of the biggest differences was the printing time. Even with input shaping enabled, the Prusa managed to print this object in 39 minutes. While the custom build machine, that uses higher speeds, acceleration and flow rates, did it in only 18 minutes. This is a big difference, which highlights the properties of the finished printer. While the quality is not quite on par, the printing time can be a great benefit in certain applications.

It is true that the Prusa may be capable of higher speeds and accelerations than the ones predefined in the used speed profile, but point of this comparison was not the testing of the limits of an off-the-shelf printer. It is unknown whether it would deliver the same quality as it did with this profile.

Tolerance test
The other testing model was selected to be a tolerance test. Available from [23]. This model contains circular chamfered discs, which are printed in place with a base. The purpose of this object is to test the printer's ability to create walls with small gaps between them. The respective gap sizing in milimeters can be seen on the discs of the print.

Same slicing parameters were used as in the first test.

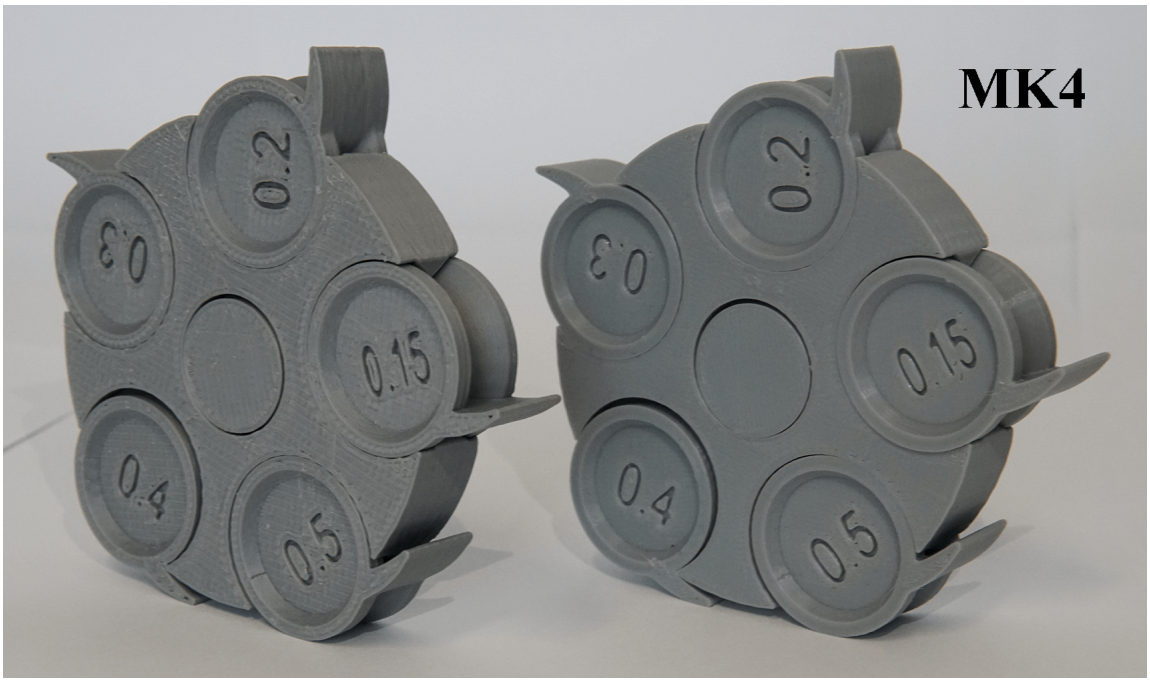


Figure 5.4: Tolerance test print

Both printers passed the test without issues. After printing, the discs can be rotated to check if the gap was printed well. All of them rotated freely, including the smallest one with a 0.15 mm gap.

As for the quality, the same can be said as in the first comparison. The Prusa produced a much nicer finish on the top surface. And the walls were more consistent with a nice alignment.

The printing time was one hour and 21 minutes, while the custom printer managed to do it in just 42 minutes.

6 Conclusion

The whole process that was described in the previous chapters has been a challenging journey from which many experiences were gained. The practical outcome of this process, besides this thesis, is a fully functional high performance 3D printer. It is basesd on the CoreXY motion platform and utilizes a capable firmware with advanced printing functions. The printer showed to be able to create high quality, dimensionally accurate parts at a rapid pace. Looking back, it was a challenging path to go through the many iterations of the designed parts, while using the same printer for production.

The involved theoretical concepts were first discussed in chapter 2. It was divided into four main parts. The first one gives background to how a motion platform works, the second covers motors and their driving electronics. In the third part, printer firmware is introduced, including the one that allowed the built printer to fulfill the design requirements. The last part delves into the specifics of how and why are advanced printing functions used.

The description of the practical part begins with 3. This chapter covers the design, calculations and the physical build of the printer. First a CAD model was created of the initial design, then a dynamic analysis was concluded, which shows how mass and other properties affect the motion system. After going through the mechanical parts, electronics were installed, which allowed for the implementation of the firmware. The process of installation and configuration is discussed in detail in section 3.5.

At this point, the printer became functional, but needed calibration and tuning. First, the basic procedures were discussed in section 4.1 after which, input shaping was discussed - the most important advanced printing function that was tested. Based on resonance measurements, frequency responses were created, which served as valuable data to determine mechanical issues and parameters for the shaping algorithms. The last section shows the tuning method and effects of another function called pressure advance.

After setting up these functions, the printer was ready to be tested and compared with a commercial one. A Prusa MK4 with input shaping was selected for this task. Conclusions were made from comparing prints that were printed with identical settings.

While the Prusa MK4 showed better quality and nicer surface finishes, the custom built printer was considerably faster. With the help of slicing software, it was later verified that the built printer can produce parts nearly in half the time compared to the other.

However, it is important to note, that further improvements could be made. Some of the issues and ideas are discussed in section 5.1.

References

[1] *Stepper vs Servo* [online]. 2024. [visited on 2024-02-19]. Available from: https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/stepper-vs-servo//.

[2] *NEMA size standards* [online]. 2024. [visited on 2024-02-05]. Available from: https://www.nema.org/.

[3] *Allegro MicroSystems A4988* [online]. 2024. [visited on 2024-03-18]. Available from: https://www.allegromicro.com/en/products/motor-drivers/brush-dc-motor-drivers/a4988.

[4] *Texas Instruments DRV8825* [online]. 2024. [visited on 2024-03-18]. Available from: https://www.ti.com/product/DRV8825.

[5] *Analog Devices TMC* [online]. 2024. [visited on 2024-03-18]. Available from: https://www.analog.com/en/product-category/stepper-motor-driver.html.

[6] *Analog Devices* [online]. 2024. [visited on 2024-03-18]. Available from: https://www.analog.com/en/index.html.

[7] *The Secret to Silent Stepper Motor Control* [online]. 2024. [visited on 2024-03-18]. Available from: https://www.analog.com/en/lp/001/secret-silent-stepper-motor-control.html.

[8] *Marlin Firmware* [online]. 2024. [visited on 2024-03-22]. Available from: https://marlinfw.org/.

[9] *Prusa Research* [online]. 2024. [visited on 2024-02-19]. Available from: https://www.prusa3d.com/.

[10] *Smoothieware GitHub repository* [online]. 2024. [visited on 2024-02-14]. Available from: https://github.com/Smoothieware/Smoothieware.

[11] *Klipper GitHub repository* [online]. 2024. [visited on 2024-02-14]. Available from: https://github.com/Klipper3d/klipper.

[12] *Klipper documentation* [online]. 2021. [visited on 2024-02-13]. Available from: https://www.klipper3d.org/.

[13] TRONVOLL, S.A. Investigating pressure advance algorithms for filament-based melt extrusion additive manufacturing: theory, practice and simulations. In: *Rapid Prototyping Journal*. Emerald Publishing Limited, 2019, pp. 830–839. Available from DOI: 10.1108/RPJ-10-2018-0275.

[14] *Klipper Pressure Advance Tuning* [online]. 2024. [visited on 2024-03-16]. Available from: https://www.obico.io/blog/klipper-pressure-advance/.

[15] *Full control models* [online]. 2024. [visited on 2024-02-26]. Available from: https://fullcontrol.xyz/#/models.

[16] *Bambulab* [online]. 2024. [visited on 2024-03-05]. Available from: https://bambulab.com/en-eu.

[17] *VoronDesign* [online]. 2024. [visited on 2024-03-04]. Available from: https://vorondesign.com/.

[18] *HevORT DIY 3D Printer* [online]. 2024. [visited on 2024-03-04]. Available from: https://hevort.com/.

[19] *BIGTREETECH SKR MINI E3* [online]. 2024. [visited on 2024-03-23]. Available from: https://biqu.equipment/products/bigtreetech-skr-mini-e3-v2-0-32-bit-control-board-for-ender-3.

[20] *Klipper Installation and Update Helper* [online]. 2024. [visited on 2024-04-21]. Available from: https://github.com/dw-0/kiauh.

[21] *Califlower Calibration Tool* [online]. 2024. [visited on 2024-04-28]. Available from: https://vector3d.shop/products/califlower-calibration.

[22] *3DBenchy* [online]. 2020. [visited on 2024-05-13]. Available from: https://www.3dbenchy.com/.

[23] *Clearance test* [online]. 2022. [visited on 2024-05-13]. Available from: https://www.printables.com/model/212736-clearance-test.