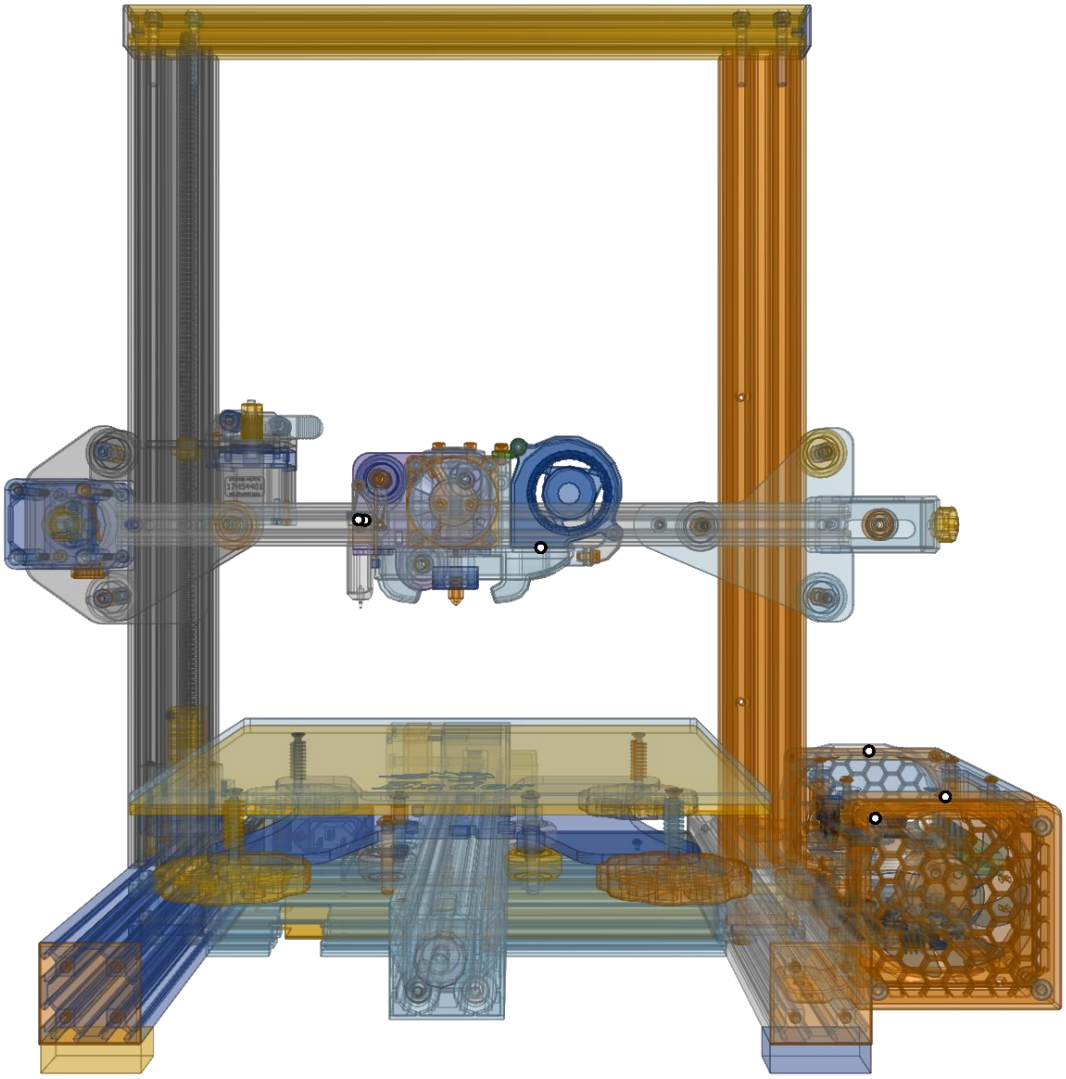


Project Ender 3 V2

An Electrical, Software and Hardware Overhaul

Created By

Ervince Allen

Completed: June 2025

Version: 1.0

Email: contact@ervince-engineering.com

Table of Contents

Portfolio Summary

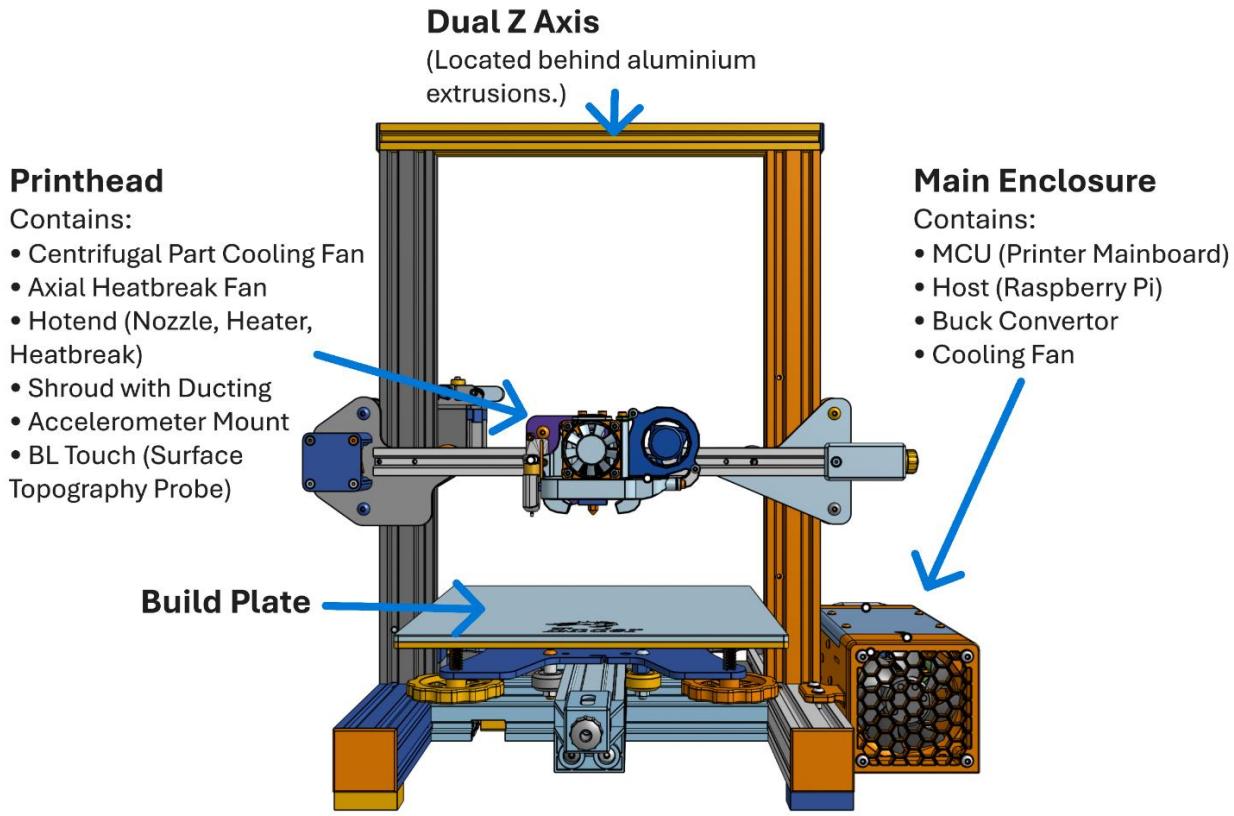


Figure 1: Shows enhanced printer with component list.

This project documents what started as a stock Ender 3 V2 3D printer—and how I used it to expand and refine my skills through hands-on engineering.

Core Skill Highlight

- Firmware configuration and G-code automation (Klipper)
- Parametric CAD modelling and constraint-based design (Onshape)
- Sensor-driven motion tuning (Input Shaping, Pressure Advance)
- Automated bed probing and real-time mesh compensation
- CFD-analysed airflow design and GPIO-based fan control
- Modular electrical rewiring with signal-integrity planning
- **Multi-system integration**: mechanical, electrical, and software
- **Remote monitoring** and diagnostics (Raspberry Pi, Mainsail)
- Iterative problem-solving under real-world constraints

1. Introduction

I have always been drawn to complex systems and problem-solving. Whether converting RC cars into boats as a child, training as an electrician, or building and maintaining PCs, I've looked for ways to take things apart, understand how they work, and improve them.

3D printing felt like a natural extension of that. The idea of being able to create obscure parts that would've otherwise taken hours of searching or bodging together was especially appealing. It felt like the most direct way to bring my imagination to fruition.

Before the printer even arrived, I had already started teaching myself CAD because I wanted to do more than just print models—I wanted to design them.

What began as a basic machine turned into a launchpad for skills in CAD, CFD, firmware, low-voltage electrical systems, motion mechanics, and real-world problem solving.

I've tackled everything from airflow tuning with CFD analysis and GPIO scripting to structural vibration and thermal management.

Along the way, I have come to understand not just how the machine works, but how to make it better.

This document is not a step-by-step guide, nor does it cover every minutia of the design decisions that were made. It is a thoughtful reflection on the work, the challenges, and the engineering thinking behind some key decisions—what worked, what didn't, and what I would improve.

2. Print Head & Cooling System

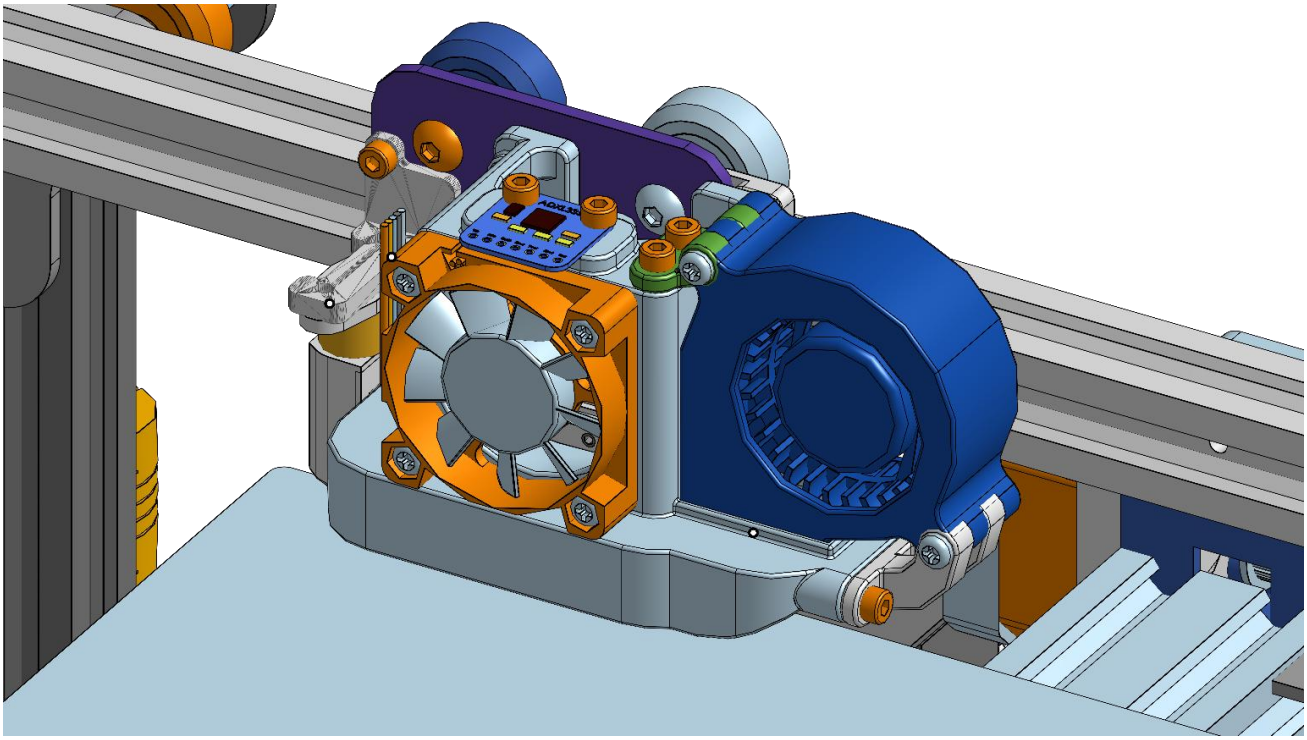


Figure 2: Shows enhanced print head.

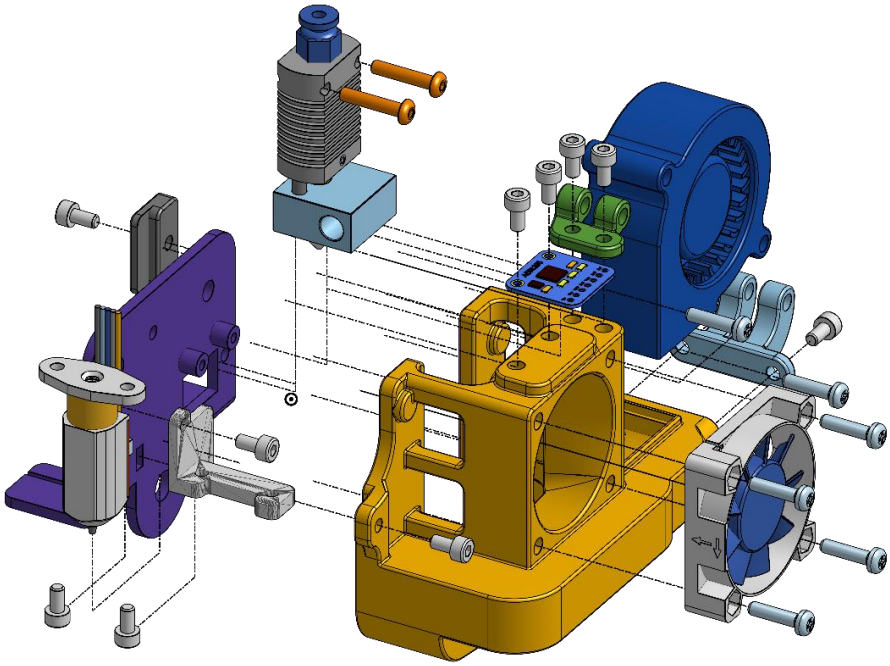


Figure 3: Shows explode view of enhanced print head assembly.

(See Appendix A for a multi-view of the print head shroud)

(See Appendix B for large format explode-view of the print head shroud)

Overview

Having already designed several models of my own—including the electrical enclosure—my intention was to try out a pre-existing shroud as a temporary enhancement before making my own. However, that quickly turned into a major overhaul, mainly because I liked the fundamentals of the design and saw clear opportunities to improve it.

The original design was appealing due to its low part count and reuse of stock components, including the (40×10mm) axial (heatbreak) heatsink fan and a (20×10mm) centrifugal part cooling fan. It offered a way to evaluate a new setup without any upfront investment, while still improving on the baseline configuration.

What follows is a breakdown of the original design's limitations, how my plan evolved, and the key changes I made to expand its scope—particularly in airflow dynamics, whilst maintaining structural stiffness.

Ducting

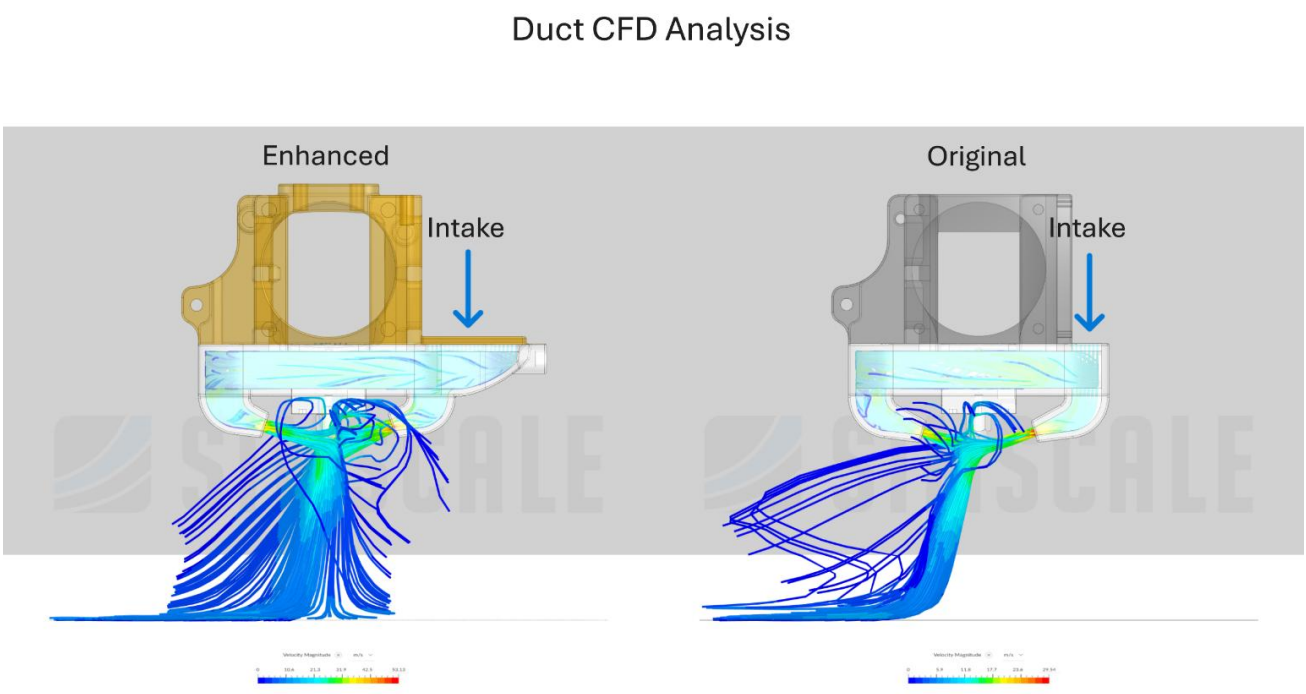


Figure 4: Shows CFD comparison of ducting. Enhanced(Left). Original (Right).

(See Appendix C for larger presentation).

Overview

The goal was quieter but more powerful part cooling with full Klipper/G-code control of the fan.

To achieve this, I increased the inlet size and duct volume, improving the flow paths to reduce backpressure and support a larger PWM-controlled (50×20mm) centrifugal fan.

Pre-CFD

The original design was allegedly CFD-evaluated, so I assumed that both nozzles were already airflow-balanced. To avoid disrupting that balance, I mirrored any structural changes made to one side of the duct on the other. While identical geometry doesn't guarantee equal performance—especially with asymmetries in duct length or curvature—it provided a reasonable basis for consistent flow behaviour.

Post-CFD

I've since validated this assumption through CFD simulation in SimScale, using steady-state incompressible flow, hex-dominant meshing, and Bernoulli's principle along with the inlet area to determine the average inlet velocity. The results confirmed that the mirrored design maintains balanced airflow across both channels, with only slight divergence due to expected path resistance differences.

Part Cooling Fan Efficiency

One issue I noticed early on was that the original fan orientation would have blocked one of the upgraded fan's intakes due to how compactly it was mounted. While the obvious route would have been to keep everything tightly packed—minimising overhangs and reducing lever arms that could introduce ringing—I leaned toward improving airflow performance instead.

To ensure both intakes remained fully open, I rotated the fan 90 degrees and continued splitting the airflow into two separate channels—accounting for the higher resistance on one side due to its added length and more corners.

ADXL345 Mounting

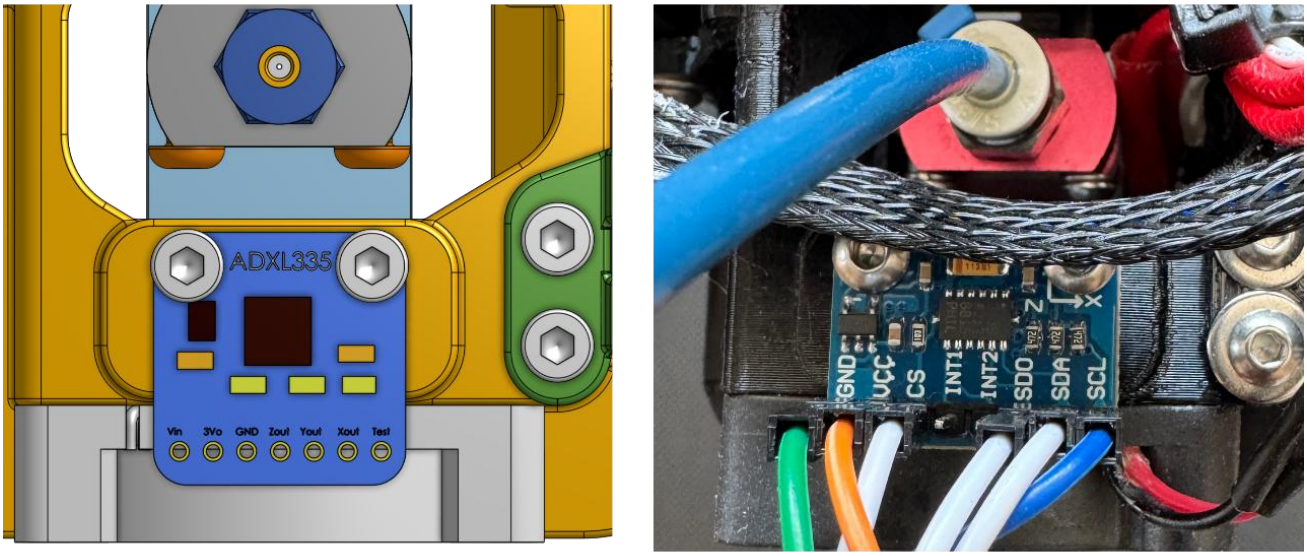


Figure 5: Show ADXL345 mount.

Overview

A mount was added to the print head to facilitate the use of Input Shaper on the X-axis.

The mounting point was chosen for its assumed stiffness—giving the accelerometer the cleanest signal—and because it allows it to remain in its default orientation, eliminating the need for additional configuration in Klipper.

More Information

For details on Input Shaper, see Section 9: Print Quality & Testing/Input Shaper.

Larger Fan Mount

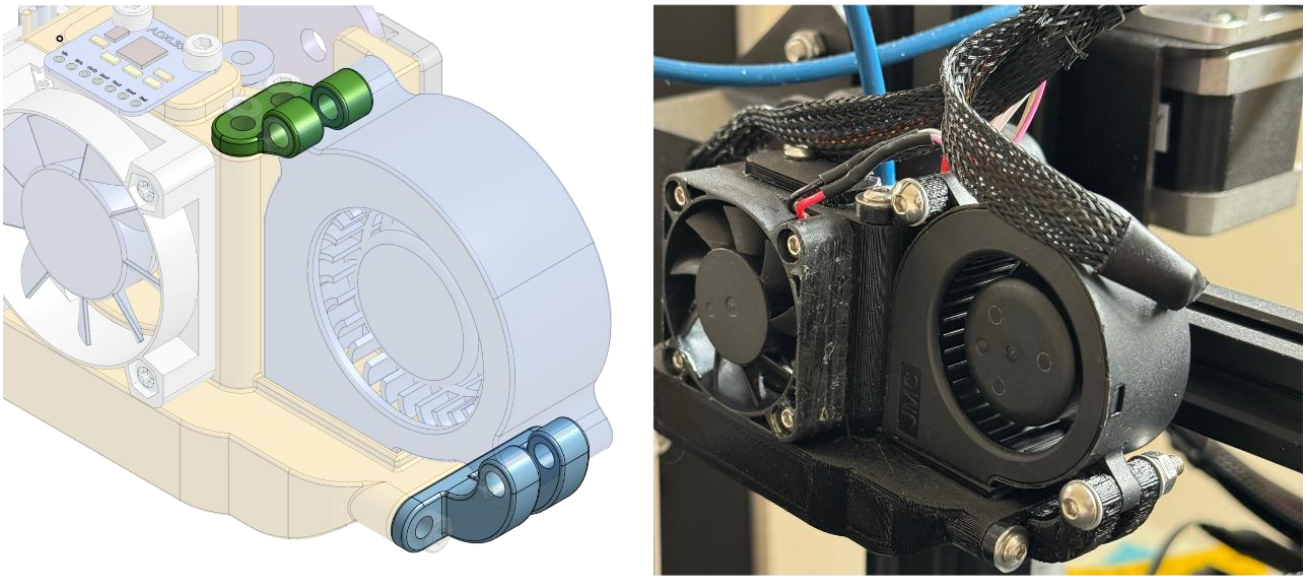


Figure 6: Shows part cooling fan mount.

Overview

The part cooling fan mount needed to be secure enough to reduce ringing, but still serviceable.

By making the mount a separate part, I could replace the fan without needing to reprint the entire shroud.

Reflections

Starting from Scratch

My changes leaned toward stiffness over minimal weight—in part because Input Shaping deals better with mass than with parts that flex or twist under inertia.

Looking back, I could have started from scratch and designed a new shroud entirely in Onshape.

Because the original model was not parametric, making changes was laborious. I had to strip away fillets just to move surfaces and spent hours visually inspecting and making more changes.

The original model also didn't fit my variation of mounting plate, so that had to be fixed.

That said, I do not regret sticking with it. Part of the value was in adapting someone else's model under constraints and I learnt a great deal from working within those boundaries.

Future Improvements

Strain Relief

I would also build in a proper strain relief system for cables, rather than relying on zip ties.

Water Cooling

For future enclosed setups, I have acquired a water-cooled heat break. It will allow me to eject heat-break thermal energy more effectively—regardless of chamber temperatures—and help improve thermal performance while reducing noise.

3. Dual Z-Axis Synchronisation

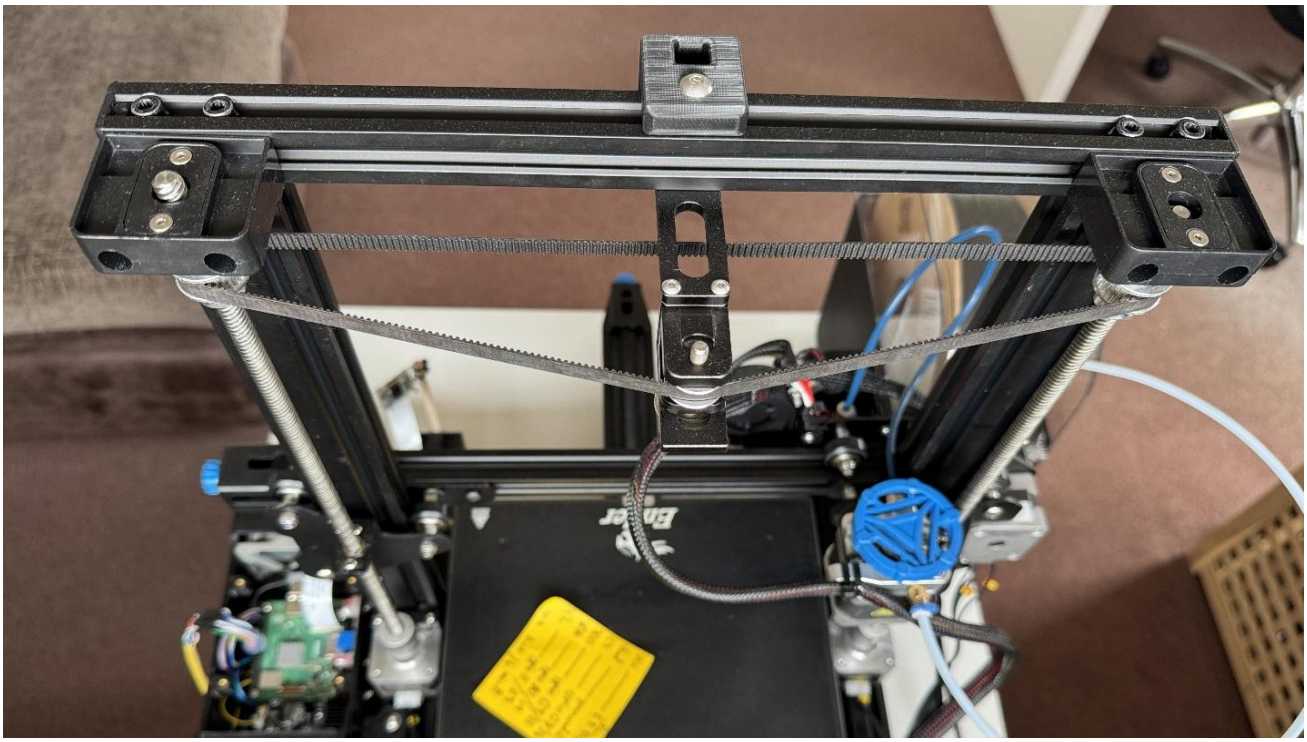


Figure 7: Shows Dual Z-Axis system with synchronisation belt and tensioner.

Overview

This enhancement was necessary to improve the dimensional accuracy of printed parts.

On the stock Ender 3 V2, the Z-axis lead screw only supports the gantry on one side, which allows the unsupported side to sag slightly—especially over time. This results in layer misalignment and poor first-layer consistency.

Before this enhancement, I avoided most Z-banding issues by manually adjusting the unsupported side of the gantry slightly higher and allowed it to settle. It worked well enough to avoid print defects, but it was a hassle to maintain and prone to drift over time—especially after gantry bumps or manual movement.

Solution

I initially purchased a kit that included, a stepper motor, a lead screw, and a simple parallel splitter cable—that connects the two stepper motors in series.

Electrically, the motors are not independently controlled—this is a completely passive system. This kit also provides no physical link between them, so synchronisation over time cannot be guaranteed.

To fix this, I added a mechanical synchronisation system that included matching pulleys on each Z-axis leadscrew—that's connected with a continuous timing belt.

This ensures that both sides of the gantry stay level, even after power loss or manual movement.

Reflections

Tensioner Issue

This enhancement was mechanically simple in theory, but I didn't fully account for real-world tolerances—such as the belt arriving too long. I had calculated a "perfect" belt length based on pulley pitch and spacing, aiming for a zero-slack system.

While I'm familiar with typical manufacturing variance, I now realise that assuming a perfect fit left no margin for adjustment. In hindsight, I should have planned for an adjustable range from the beginning.

To correct the mismatch, I retrofitted a tensioner to bring the belt into spec. It works well, but the preload adjustment is fixed and a little awkward to modify.

That said, adding a tensioner wasn't just a workaround—it's a sound design choice. It allows for correct belt tension without much work, making the system more adaptable and easier to fine-tune.

The mechanical synchronisation system itself has proven reliable. It eliminated the need for manual gantry correction and made the Z-axis far more robust and repeatable over time—even after power loss or mechanical disturbance.

Future Improvements

Tensioner

In the future, I would incorporate adjustable tensioning from the outset and treat "perfect fit" as a tolerance range, not a hard target.

4. Filament Drying System

Overview

Before I acquired a dedicated dryer, I assumed vacuum-sealed filament came pre-dried. I was wrong.

Early PETG prints were riddled with bubbling and stringing—obvious signs of excessive moisture in the filament. It quickly became obvious that I needed better moisture control, especially when printing hygroscopic plastics.

Initial Solution

My first attempt was using a conventional oven. I added a thermocouple to monitor the temperature and used a PC fan for basic air recirculation.

This setup worked surprisingly well but had several major drawbacks. It was energy-inefficient, monopolised the kitchen oven, and made filament drying a chore rather than a background process.

Solution

To resolve these issues, I purchased the Creality Filament Dry Box 2.0, chosen for its solid build and at a reasonable. It provided a dedicated and space-efficient alternative.

Reflections

Thermal Performance

Beyond moisture control, preheating filament closer to its glass transition temperature reduces the thermal load on the hotend. This can slightly increase maximum flow rate by improving extrusion efficacy.

However, overheating can soften filament prematurely or degrade its properties—so control is critical.

Ventilation Limitations

The dryer lacks active ventilation so over time, the chamber saturates with humid air, stalling further evaporation. My workaround has been to manually open the lid every few hours during long drying sessions to vent moisture buildup.

Temperature Stratification

Thermocouple tests confirmed significant temperature gradients, in the drying chamber. The air near the heater is noticeably hotter than the air near the top—this leads to uneven drying across the spool.

To address this, I manually rotate the spool to even out heat exposure when not printing. (Printing naturally rotates it.)

Current Material Limitations

Currently, I have only printed with PETG and PLA.

Higher-performance materials like Nylon and ABS are not practical with this machine due to the 250°C nozzle temperature limit and the lack of a heated chamber to prevent warping.

Future Improvements

Ventilation

Automatic ventilation based on internal humidity.

Spool rotation

A low-RPM spool rotation system to ensure even drying across the roll.

5. Host & MCU

Overview

A host device was added because, even with custom firmware—the Ender 3 V2's stock motherboard (MCU) is heavily constrained by its processing power and expandability. To offload computation and enable advanced features, I added a Raspberry Pi 3A+ as a dedicated host.

Host Functions (Raspberry Pi 3A+)

Klipper Host Firmware

The core of Klipper and interfaces with the MCU via USB.

Mainsail Web GUI

Provides a lightweight, browser-based interface for monitoring and control.

Wi-Fi Networking

Manages all wireless communication, enabling remote access and control.

GPIO-Based Fan Control

Uses Raspberry Pi GPIO pins to control fans and read their state.

(For details on Fan Control, see Section 8: Software Setup & Automation.)

Remote Camera Monitoring

A Pi camera module provides real-time print monitoring via the web interface.

ADXL345 (Input Shaper Accelerometer)

Connects via the GPIO pins to offer resonate frequency compensation.
(For details on Input Shaper, see Section 9: Print Quality & Testing.)

MCU Functions (Ender 3 v2 Mainboard)

The stock Ender 3 V2 mainboard remains in use. It is flashed with Klipper firmware and now manages the topographic build plate probe and real-time stepper motion execution, with all high-level processing offloaded to the host.

Reflections

Stability

This setup has been stable, with no crashes or hangups across days and weeks of print time. The camera system has also worked without issue.

More Information

For details on software configuration, including Klipper setup, fan control logic, and macros, see Section 8: Software Setup & Automation.

For detail on advanced testing features, including Topographic Build Plate Probing, Pressure Advance and Input Shaper, see Section 9: Print Quality & Testing.

6.Electrical Enclosure Design & Construction

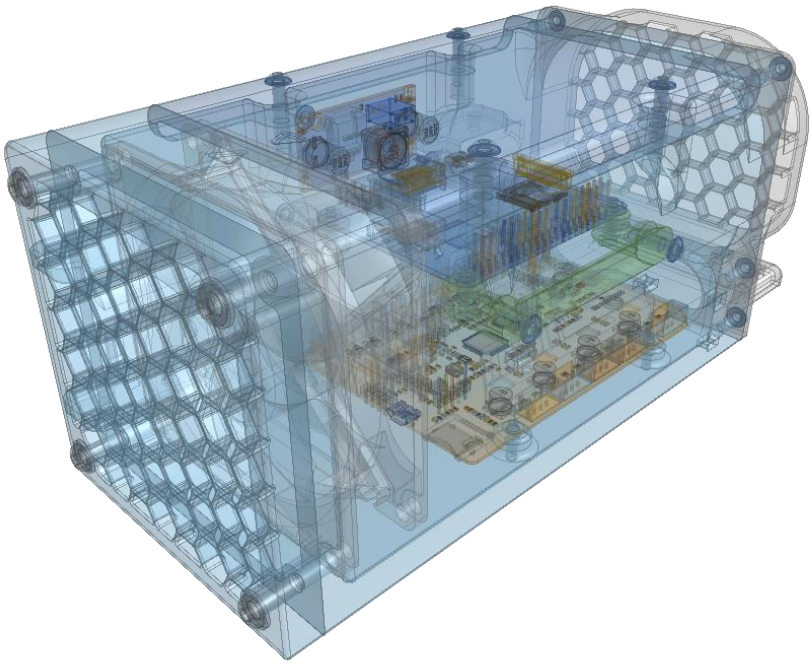


Figure 8: Shows transparent CAD model of the electrical enclosure.

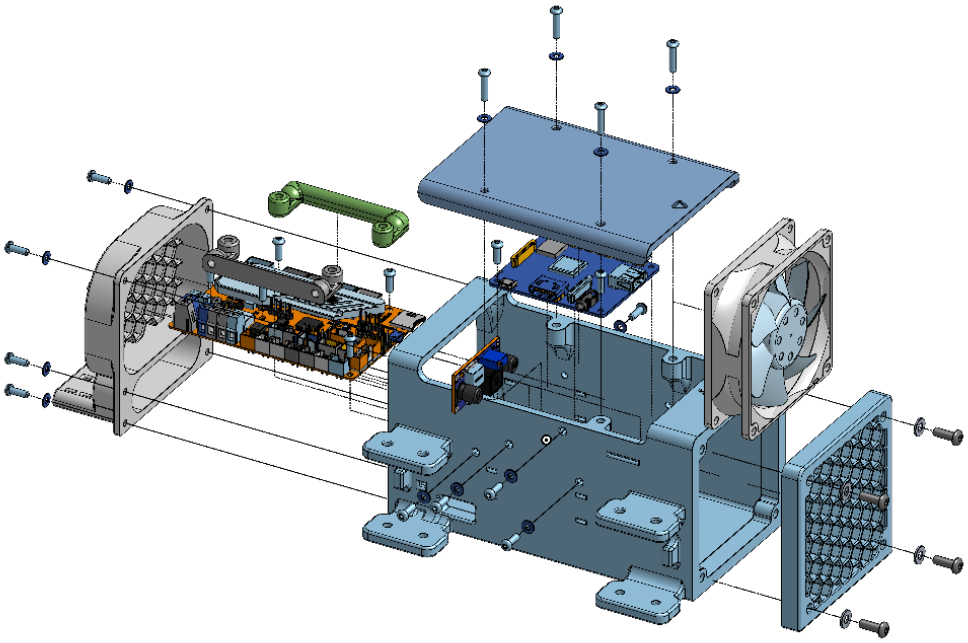


Figure 9: Shows explode view of the electrical enclosure.

(See Appendix D for a larger presentation).

Initial Design

The stock printer cooling intakes are located on its underside—less than 2 cm from the surface it sits on.

Due to airflow restriction. I had designed ducts to route airflow upward through the mounting surface. But soon after I finished modelling the parts, I realised I was engineering around a fundamentally bad design. Even if airflow had a direct path, the fans were still too small and loud.

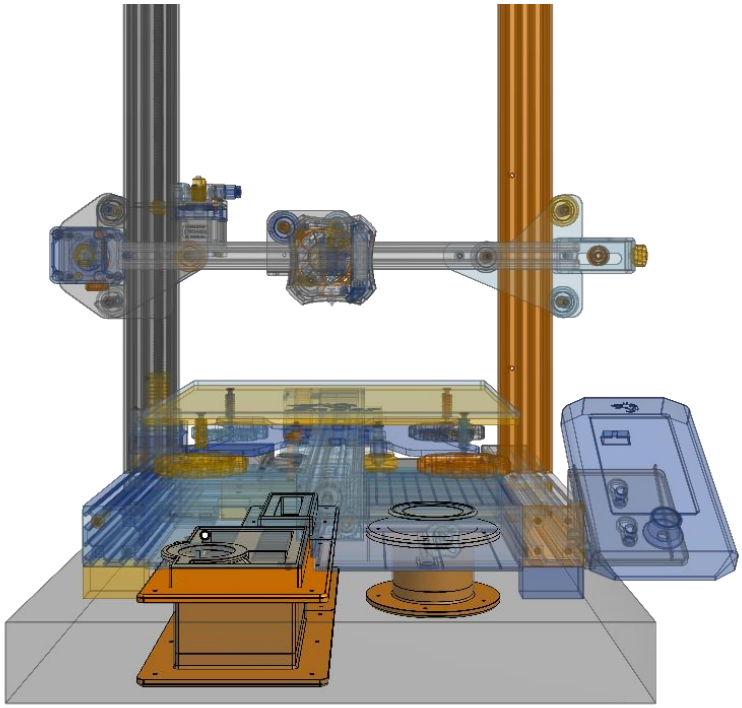


Figure 10: Shows through table grommet design with stock printer.

The initial grommet design intended to improve airflow. Although this was not implemented, the exercise informed my final electrical enclosure redesign.

Electrical Enclosure Design

With the original system and design being scrapped, I set out to redesign the printer's electrical enclosure from the ground up.

Design Goals

- Compact footprint for an efficient use of space
- Larger fan support for higher airflow and less noise
- Housing for Raspberry Pi, buck converter, and printer's mainboard
- Secure internal and external mounting for all components and cables
- Enhanced serviceability

Implementation Challenges

At face value, none of these goals are particularly complicated—but integrating them all into one coherent design turned out to be one of the hardest parts of this entire build.

Considerations:

- Physical tolerances and alignment
- Cable runs and connector accesses
- Internal airflow routing
- Electromagnetic interference

Integration

It is what people sometimes call "integration hell"—fixing one thing would often break something else.

Thankfully, it wasn't too much of an issue. Onshape's parametric modelling really helped me during this phase.

It let me keep dependencies connected and respond to changes without breaking the model.

Reflections

This electrical enclosure was the culmination of everything I learnt in CAD. It is the most iterated design in this whole project—and though it was frustrating at times, learning how to build something where multiple systems meet and finding a way to make it all work is part of why I enjoy engineering.

Even though the result is a marked improvement over the stock configuration, there are still things I would like to improve.

Improvements

External Connectors

Routing all power and communication through external connectors instead of direct passthroughs—would make the system much easier to maintain or reconfigure.

Electromagnetic interference

Even though I moved the buck converter away from the Raspberry Pi's GPIO pins, I'd like to design EMF mitigation into the electrical enclosure.

Reinforcement

Increase the electrical enclosure infill and reinforce mounting points to the aluminium extrusion.

PSU & Electrical Enclosure Ducting

Enclose the PSU in a dedicated housing and add ducting for both it and the electrical enclosure. This would allow external airflow to be routed in and out without affecting internal chamber temperatures, helping to prevent print warping if the printer is ever used in a heated chamber.

This is important because internal ambient temperatures could reach 40–50 °C, in a heated chamber—which would be too high for reliable operation without dedicated airflow.

Despite these areas for improvement, the electrical enclosure achieved all major design goals.

The airflow is more efficient thanks to the larger fan and the less restrictive internal layout.

7. Electrical Systems

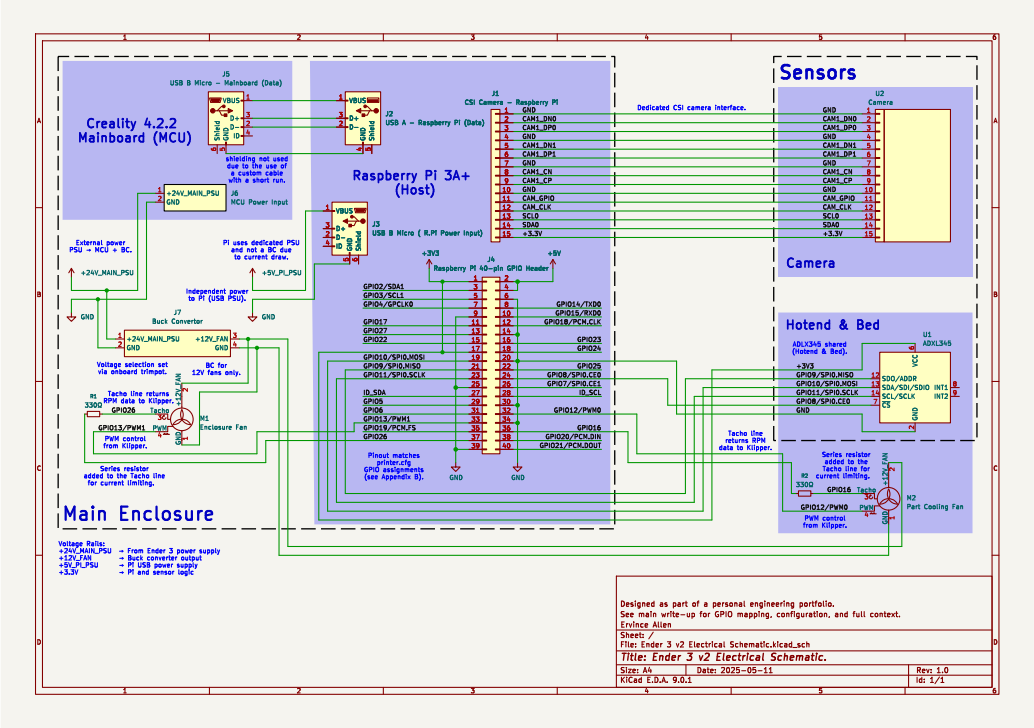


Figure 11: Shows electrical schematic of Ender 3 v2 after enhancement.

(See Appendix E for a larger presentation).

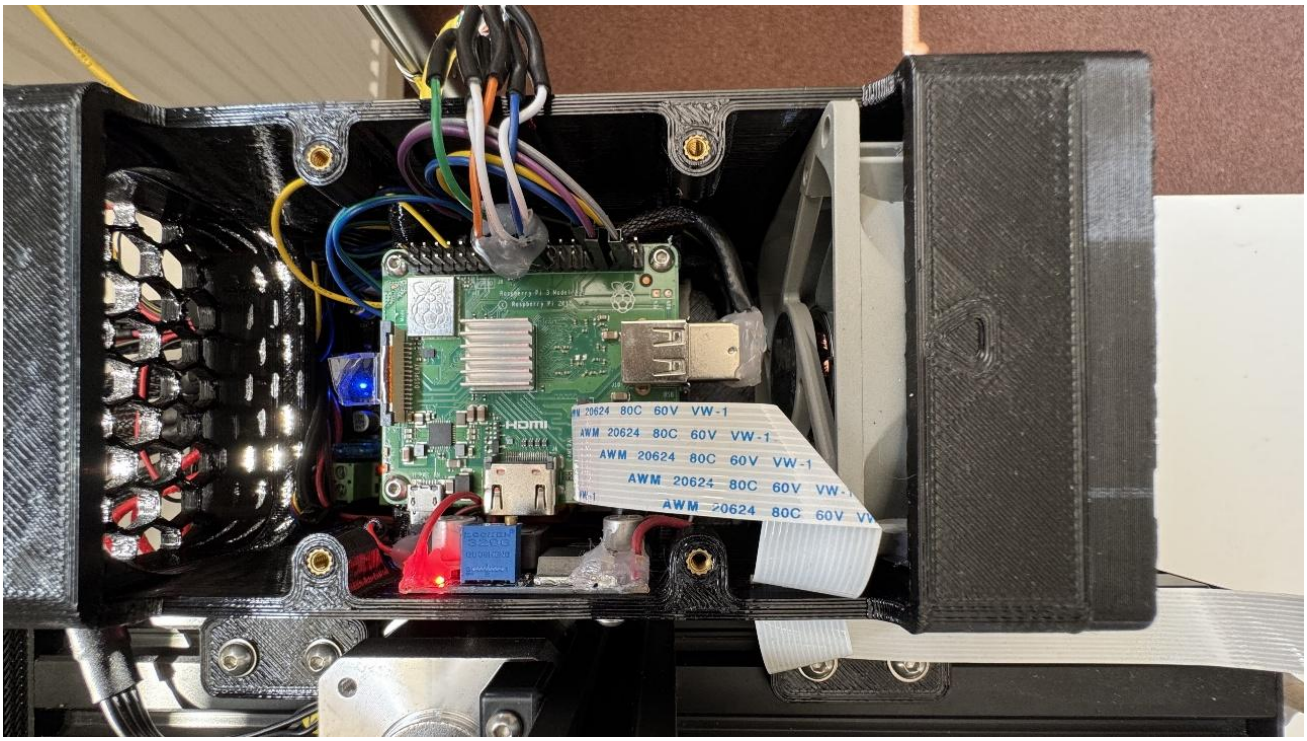


Figure 12: Shows photograph of the components in the electrical enclosure.

Overview

The stock Ender 3 V2 wiring loom is mostly adequate, but its biggest weakness is serviceability.

In the event of a failure, any work would have to be conducted with the printer on its side, as all terminations are made inside the original printer's cramped electrical enclosure on the underside, with no alternative access.

While I would have liked to rewire the loom entirely with proper modular terminations, that was outside the scope.

Design Goals

- Space efficient custom cable for Host-to-MCU communication.
- JST connecters for the part cooling fan to improve serviceability.
- Longer 24V main power cable with buck converter spur.
- Ethernet cable twisted pairs connect ADXL to Host to aid with signal integrity.
- Tune Vref on MCU to lower stepper motor temperature while maintaining torque.

Reflections

This part of the project was straightforward, but it could grow significantly more complex in the future.

Improvements

Print Head Daughterboard

Supplying a single 24V line to the print head with onboard 12V, 5V, and 3.3V regulation for any fans, sensors, or logic. This would reduce the amount of power lines, which run to the print head.

Using CAN bus for all data communication, providing a reliable and condensed data path that would also simplify the wiring loom.

External Termination

The most important thing I would change in hindsight is to add external terminals to all major connections—especially between the print head and electrical enclosure.

Right now, every cable replacement requires cutting zip ties in the electrical enclosure and re-routing cables, which makes even simple maintenance a hassle.

I had originally considered reusing an Ethernet jack for the ADXL345, but I was not comfortable with the small risk of someone accidentally plugging in a live network cable.

In the future, I will add a small USB-powered MCU like a Raspberry Pi Zero to function as a signal bridge.

That way, even if someone plugs in another device, nothing gets damaged.

8. Software Setup & Automation

Klipper & MCU Overview

Custom firmware like Klipper was vital to the success of this project.

It allows rapid iteration of configuration settings and hardware changes without the need to manually recompile the firmware, and supports advanced features such as flow rate tuning and harmonic compensation.

Why Klipper over Marlin

For many enthusiasts, the capabilities mentioned have long given Klipper an edge over Marlin, and they're part of the reason many newer consumer machines now ship with Klipper derivatives by default.

While Marlin has begun to incorporate some similar features, it still doesn't match Klipper's flexibility or ease of customisation.

MCU & Klipper Firmware

Klipper offloads all computational workload from the MCU to a host device—allowing the MCU to focus solely on executing motion instructions for the stepper motors and deploying the topographic build plate probe, reading the probe state, and retracting it.

This architecture eliminates the performance bottlenecks of a low-power control board.

As a result, I saw no practical reason to upgrade the Ender 3 V2's stock mainboard—it performs well within this distributed setup.

Mainsail Frontend

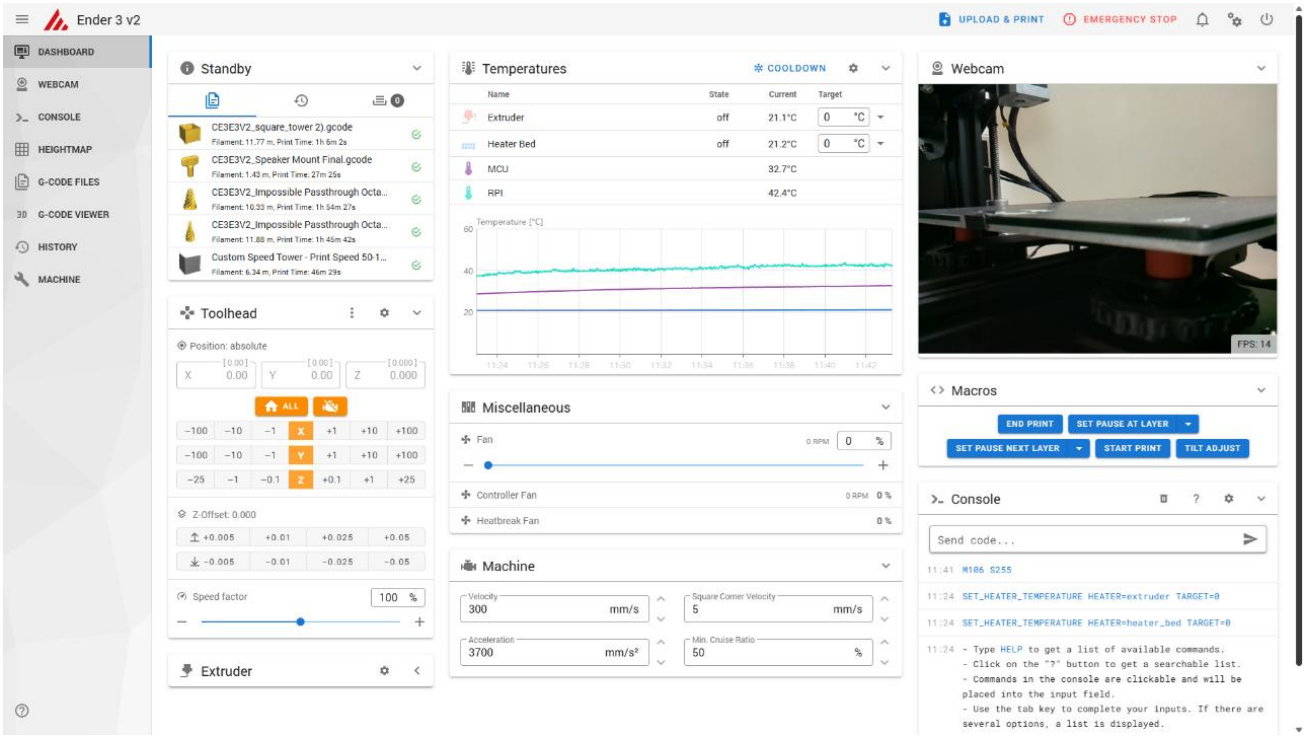


Figure 13: Shows Mainsail dashboard.

Mainsail was chosen for the frontend due to its active development, responsive UI, and strong integration with Klipper's configuration model.

It provides real-time control, temperature graphing, print monitoring, and command input through a web browser on the local network.

This combination with a Raspberry Pi camera module, offers full remote access and monitoring—critical for checking first-layer adhesion and print progress.

Fan Control and Macros

Using Klipper's config system, G-code macros, and the Pi's GPIO support, I set out to implement dynamic fan control and general automation behaviour.

Design Goals

- Custom activation thresholds
- RPM-based feedback via tachometers
- Custom start and end print macros
- Live monitoring via Mainsail

Custom Fan Logic

Electrical Enclosure Fan

Activates when stepper motors are enabled and or Raspberry Pi CPU reaches 50°C; turns off after idle timeout if Raspberry Pi CPU is below 50°C

Heat Break Fan

Turns on when the hotend exceeds 50 °C; Turns off if below 50°C

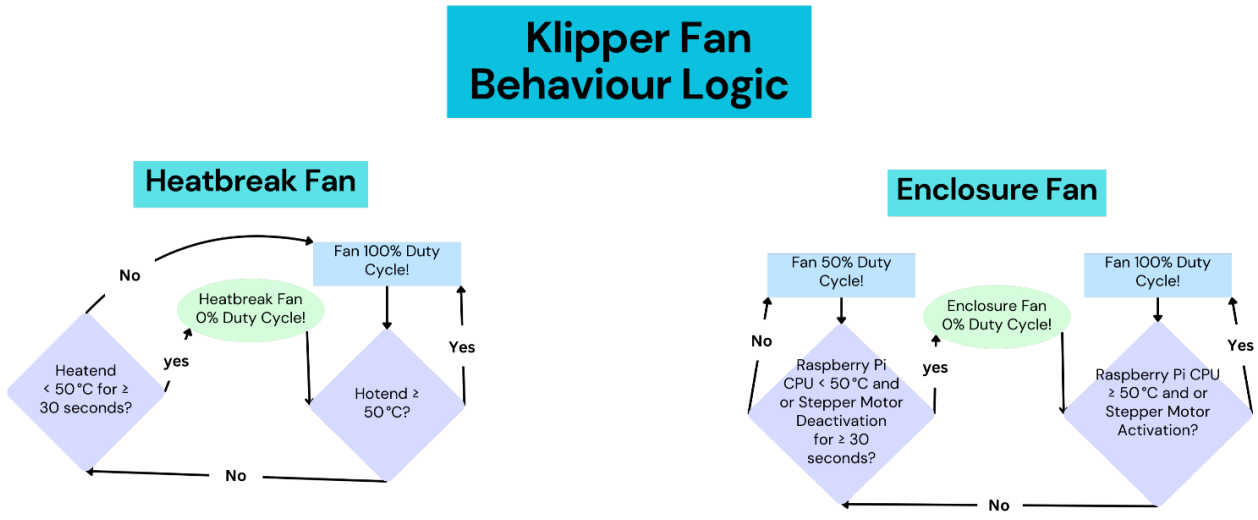


Figure 14: Shows custom fan logic flowchart.

(See Appendix G for large format of fan logic flowchart)

Start and End G-code

Start and end sequences are managed through Klipper macros rather than slicer settings.

This avoids redundant maintenance across multiple slicers and centralises configuration.

- START_PRINT macro; Lays down two purge lines to prime the hotend and clear any degraded plastic.
- END_PRINT macro; Performs a long retraction to reduce residual pressure and help prevent clogging or oozing during cooldown.

(See Appendix Y for full start and end G-code.)

Configuration File

See Appendix Z for full printer configuration, including motion tuning parameters and GPIO fan control setup.

Reflections

Klipper Fan Logic Constraints

Klipper's fan control logic has some constraints—for example, it does not allow a single fan to respond to multiple independent triggers.

Because of that, I chose to prioritise stepper driver cooling over Pi CPU temperature, as the CPU stays under 50 °C passively and is less sensitive to heat buildup.

Though Klipper recommends hardware PWM in high-load environments to avoid timing jitter from software-based control. In practice, software PWM has been completely dependable in my setup, due to the Pi's light computational load.

Aside from that, the macros and fan behaviours have worked well—and I am especially happy with how centralised and configurable this setup has been.

9. Print Quality & Testing

Overview

Due to their complex kinematics, varied part selection, and the performance of individual components, 3D printers require some form of calibration—at the very least.

As print speeds increase, this calibration becomes more involved and may require sensor-based real-time feedback loops to maintain quality.

This is not an exhaustive list of every test used, but I will outline the most notable ones in my printer's setup.

The three that I will go over are Topographic Build Plate Probing, Pressure Advance and Input Shaping, the last two of which are mandatory for high-speed printing. I will briefly explain how I interpret their function, how the tests were conducted, and how I validated the results.

Build Plate Probing

Overview

The BLTouch replaces the Z-axis end-stop and adds two critical functions that allow the nozzle to maintain the optimal distance from the build plate throughout printing.

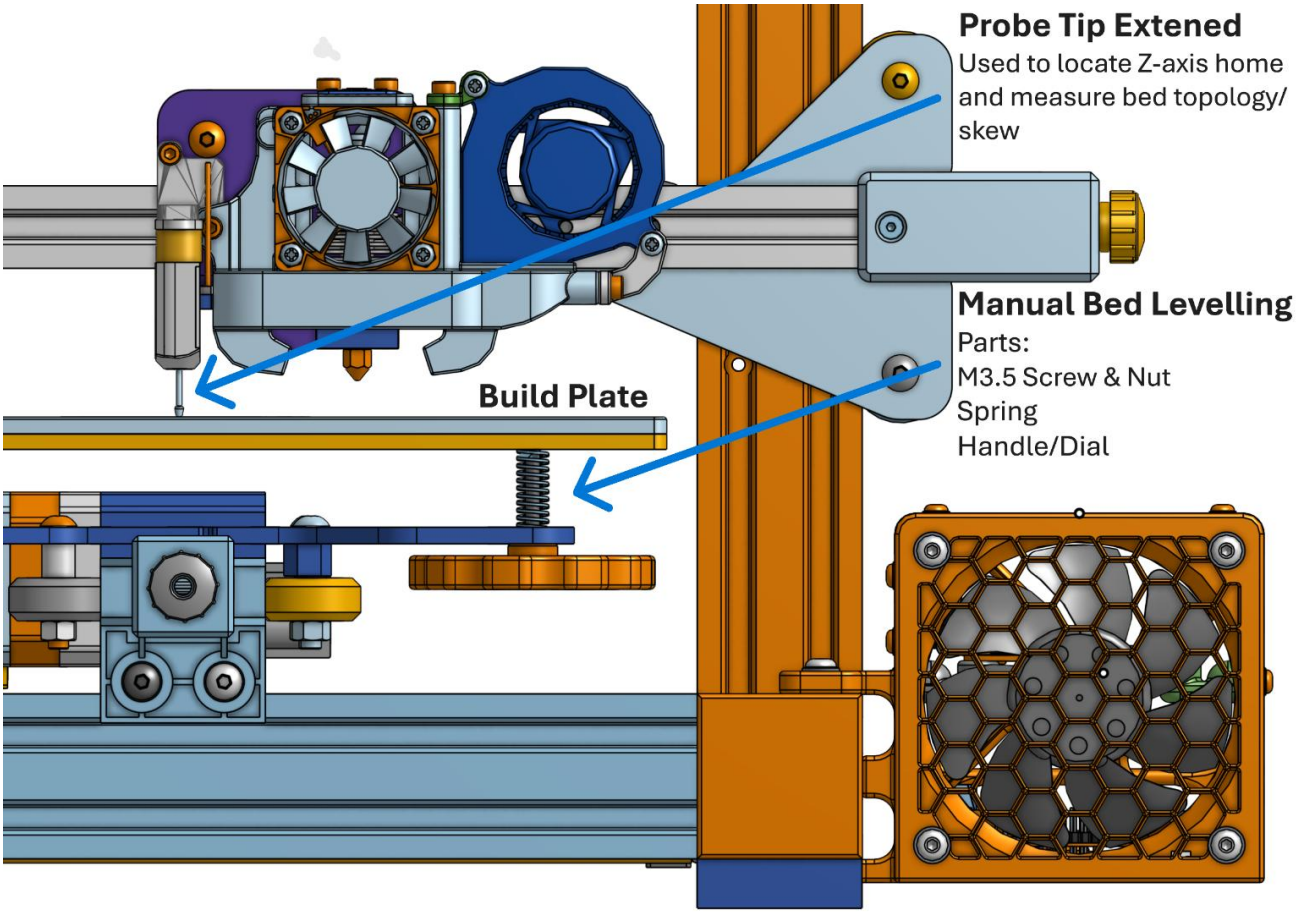


Figure 15: Shows bed probing operation and some of the components involved.

Skew Correction

To correct for skew across the X and Y axes, the probe samples all four corners of the bed and reports how far each point is from the home reference (Z = 0). These offsets are then used to guide manual adjustment of the bed levelling screws—usually via corner-mounted dials.

Klipper makes this easier by factoring in the Z-offset, thread pitch, and dial diameter, then outputting precise adjustment instructions, often in a clock-face format (e.g. “turn anticlockwise 10 minutes”).

Topology Compensation

Once skew is minimised, the probe samples a grid of points across the usable build surface, building a topographic mesh. Klipper then applies real-time Z-offset adjustments during printing to account for surface irregularities.

This enables consistent nozzle height across warped or uneven areas, which improves first-layer reliability, reduces the need for manual tuning, and enhances print dimensional accuracy.

Klipper also supports multiple saved meshes—useful when compensating for thermal expansion at different build plate temperatures.

Setup Considerations

To avoid interference from bed bolts or clips, the probing area is constrained within a user-defined “keep-in” region. The resolution of the mesh is configurable in firmware, but probing time increases with density. For this printer, a 5×5 grid provides a good balance between accuracy and efficiency, as the bed is relatively flat at this scale.

Topographic Probing Test Methodology

To evaluate the mesh compensation, I used a single-layer flatness test print.

This consists of a large grid pattern that covers the full build surface. The goal is even extrusion across all regions, without areas of over-squish or nozzle-to-bed contact.

I also manually adjusted the Z-offset after probing to fine-tune the first-layer height.

Test Results

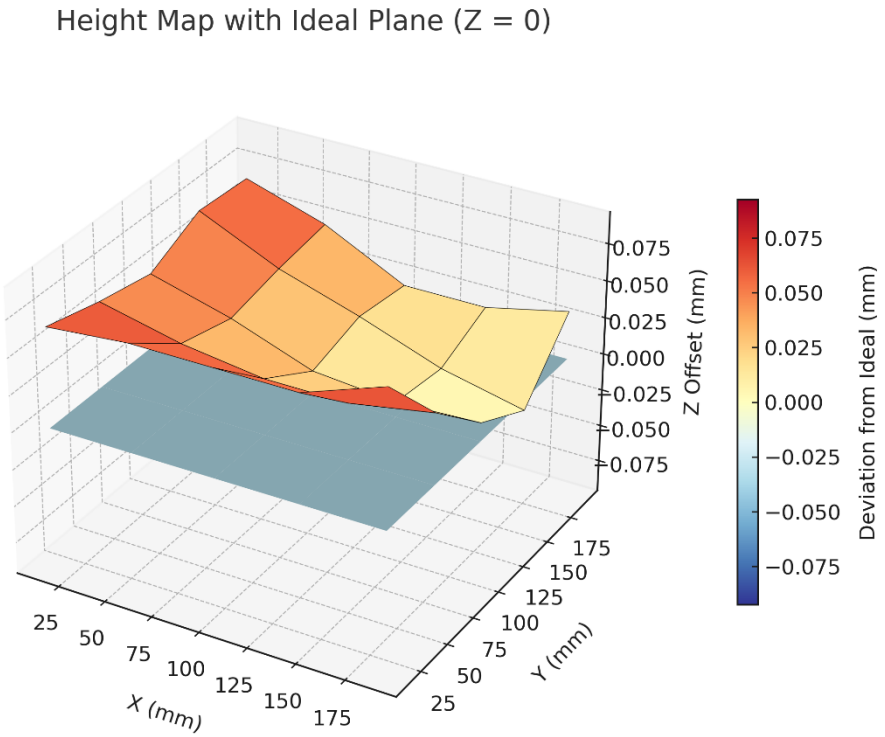


Figure 16: Shows height map of build plate.

Using a 5×5 probing grid in Mainsail, the mesh revealed approximately 0.1 mm of total deviation across the bed. The highest point was near the front-left corner (+0.092 mm), and the lowest was slightly behind and to the right of centre (–0.007 mm).

After enabling mesh compensation in Klipper, first-layer test prints showed consistent and even extrusion across the full build surface, confirming both the accuracy of the mesh and the mechanical stability of the system.

Pressure Advance

Overview

Pressure Advance works by pre-emptively extruding or retracting filament based on upcoming motion—anticipating what the nozzle is about to do rather than reacting in real time.

The importance of this tool becomes more obvious as print speeds increase. As the print head moves faster, the delay between commanded and actual extrusion becomes more noticeable—especially during rapid directional changes and at seam junctions.

An example is when the extruder is printing a sharp right angle—as the nozzle arrives at the corner, it momentarily pauses. Due to residual pressure in the system, filament keeps flowing—resulting in over-extrusion on the corner. The same logic applies as the nozzle exits the corner.

Pressure Advance counteracts this by slightly retracting the filament before reaching the corner, reducing flow just in time for the slowdown.

Without compensation, the sudden acceleration and delay in flow would initially cause under-extrusion.

Pressure Advance solves this by commanding extra flow just before it is needed, effectively smoothing the transition.

Setup Consideration

One key consideration is that Pressure Advance needs to be calibrated for every filament type, brand, and print temperature, since it is highly dependent on material viscosity.

This is where ecosystems with proprietary filament shine—the manufacturer has often done most of the tuning work and the printer can apply the settings for you.

Test Methodology

Tuning Pressure Advance is straightforward.

All that was needed was a custom-sliced test object (usually a simple tower), a few lines of G-code, and a pair of vernier callipers.

The G-code instructs the printer to incrementally adjust the Pressure Advance value at each layer.

Once the print is complete, the callipers are used to find the point on the tower where print quality is the cleanest, without bulging or gaps.

Test Results

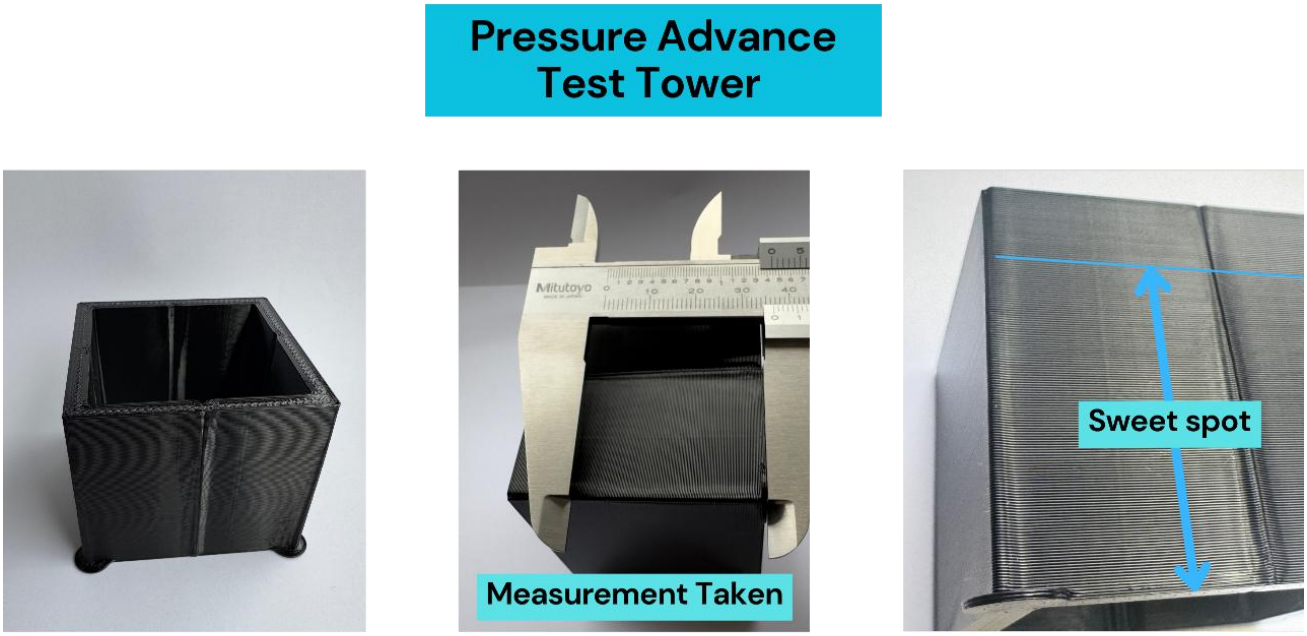


Figure 17: Shows pressure advance calibration tower with measurement example.

Input Shaper

Overview

To further improve motion accuracy, I used Klipper's Input Shaping feature—enabled via an accelerometer (ADXL345).

The goal here is to reduce resonant vibrations caused by frame movement, belt tension, and the inertia of moving parts.

Input Shaper works by identifying dominant resonance frequencies in the motion system and applying an equal and opposite force—cancelling out the unwanted vibration.

Think of Newton's third law: every action has an equal and opposite reaction—If both opposite forces are applied to the same structure, the motion cancels out.

Of course, it is not a perfect system—real-world physics introduces signal noise, frame flex, and other chaos—but the results are impressive, especially on less rigid machines.

Test Methodology

There are two ways to tune Input Shaper in Klipper

Manual Tuning

This requires manual visual pattern analysis to find the frequency nodes. It works well but less accurate and thus results will be limited.

Automatic Tuning

This uses direct measurement from an accelerometer.

I used the latter because it is faster, far more accurate, and produces a clean dataset for comparison.

The first step is to measure the signal-to-noise ratio (SNR) of the accelerometer because a clean signal ensures more accurate data—noise in = noise out.

Once verified, the sensor is mounted on each axis.

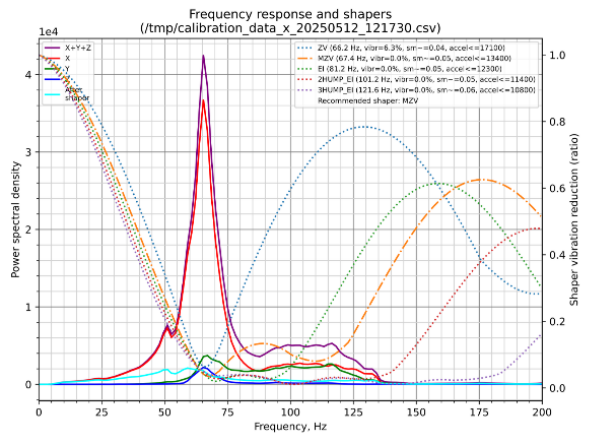
The test is then run for the X-axis (print head carriage) and Y-axis (build plate).

The Z-axis is excluded from this process since Cartesian printers do not move that axis during normal printing (only on layer changes) and Z-Hops, but it contributes nothing to system resonance.

Test Results

Input Shaper Graphs

X Axis



Y Axis

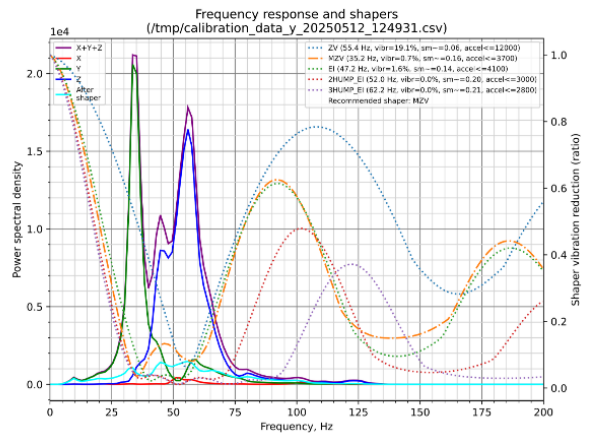


Figure 18: Show Input Shaper calibration results (X & Y Axes).

(See Appendix H and I for larger graphs of Input Shaper test results).

Resonance analysis was performed using an ADXL345 sensor mounted on the toolhead and Build-plate.

The X-axis (toolhead) showed a peak at 67.4 Hz and was tuned using the MZV shaper.

The Y-axis (build plate) showed a much lower peak at 35.2 Hz, also tuned with MZV.

To prevent over-smoothing, maximum acceleration was capped at 3700 mm/s^2, as recommended.

10. Final Reflections & Future of the Build

This project was the result of long nights and days spent thinking about CAD.
It gave me a platform to learn new skills and stretch my problem-solving muscles. I often found myself daydreaming about how to improve a part—constantly iterating on ideas, even when I was away from the printer.

Honestly, finishing the build was bittersweet because I enjoyed the process more than I expected. It made me realise that this kind of work is something I want to pursue more seriously.

As for the printer itself, it has reached the limits of what makes sense to modify. With enough tuning, I could squeeze out a bit more performance—but like that very first duct I designed, I would just be engineering around the platform's limitations.

The frame lacks stiffness, so motion accuracy suffers at higher speeds. And if I am honest, print quality is not hugely better than stock—though it was particularly good to begin with and it is much easier to achieve consistent results when printing slowly.

That said, the machine is now faster, more capable, and far more serviceable than it ever was out of the box.

Looking ahead, I hope to move to a CoreXY printer—ideally something open source—so I can keep full control and continue building on what I have learned here.

Finally, if I may indulge for a moment.

This project was about more than just 3D printing. Yes, it helped me in practical ways, and gave me a platform to experiment, refine, and grow.

But more than that, it gave me something solid to hold onto during a time when I really needed it—something that challenged me, absorbed me, and reminded me what I'm capable of when I follow my curiosity.

It's a technical document, but it's also a human story—about learning through doing, and finding focus and meaning by building something I genuinely care about.

It is the best £200 I've ever spent.

Thank you for reading.

Appendices

Supporting Diagrams, Photographs, Macros, and Configuration Files

Appendix A: Print Head Shroud

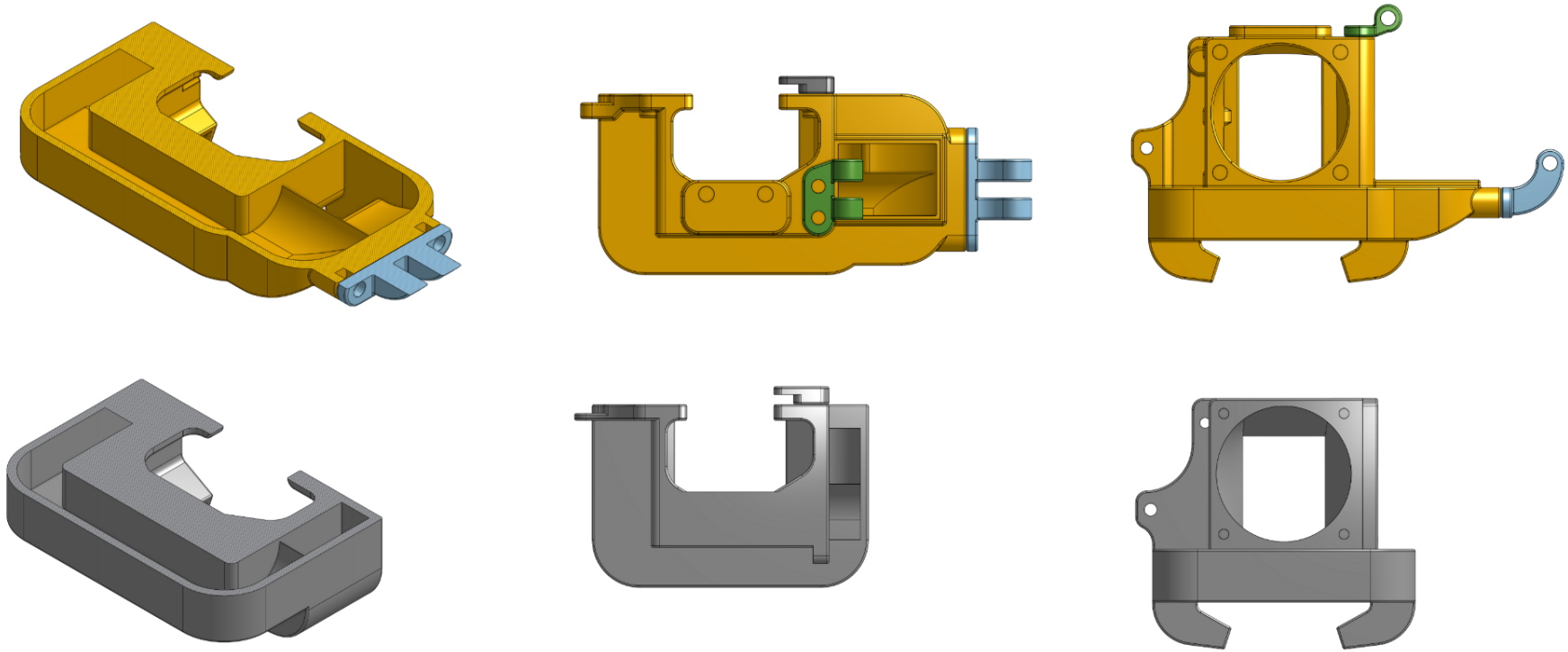


Figure 19: Shows print head shroud. (Enhanced Top. Original Bottom).

Appendix B: Print Head Explode View

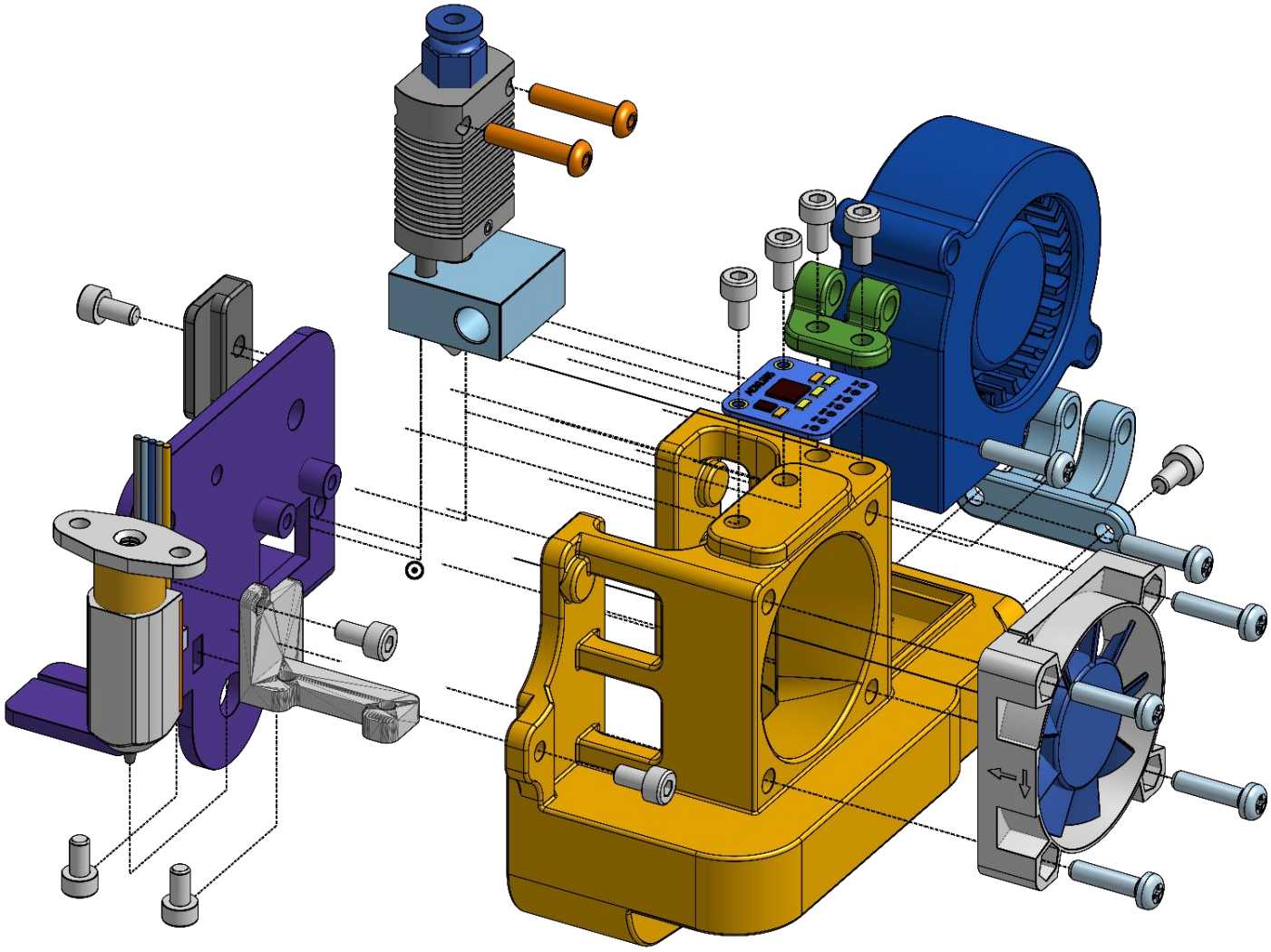


Figure 20: Shows explode view of print head shroud.

Appendix C: Duct CFD Analysis

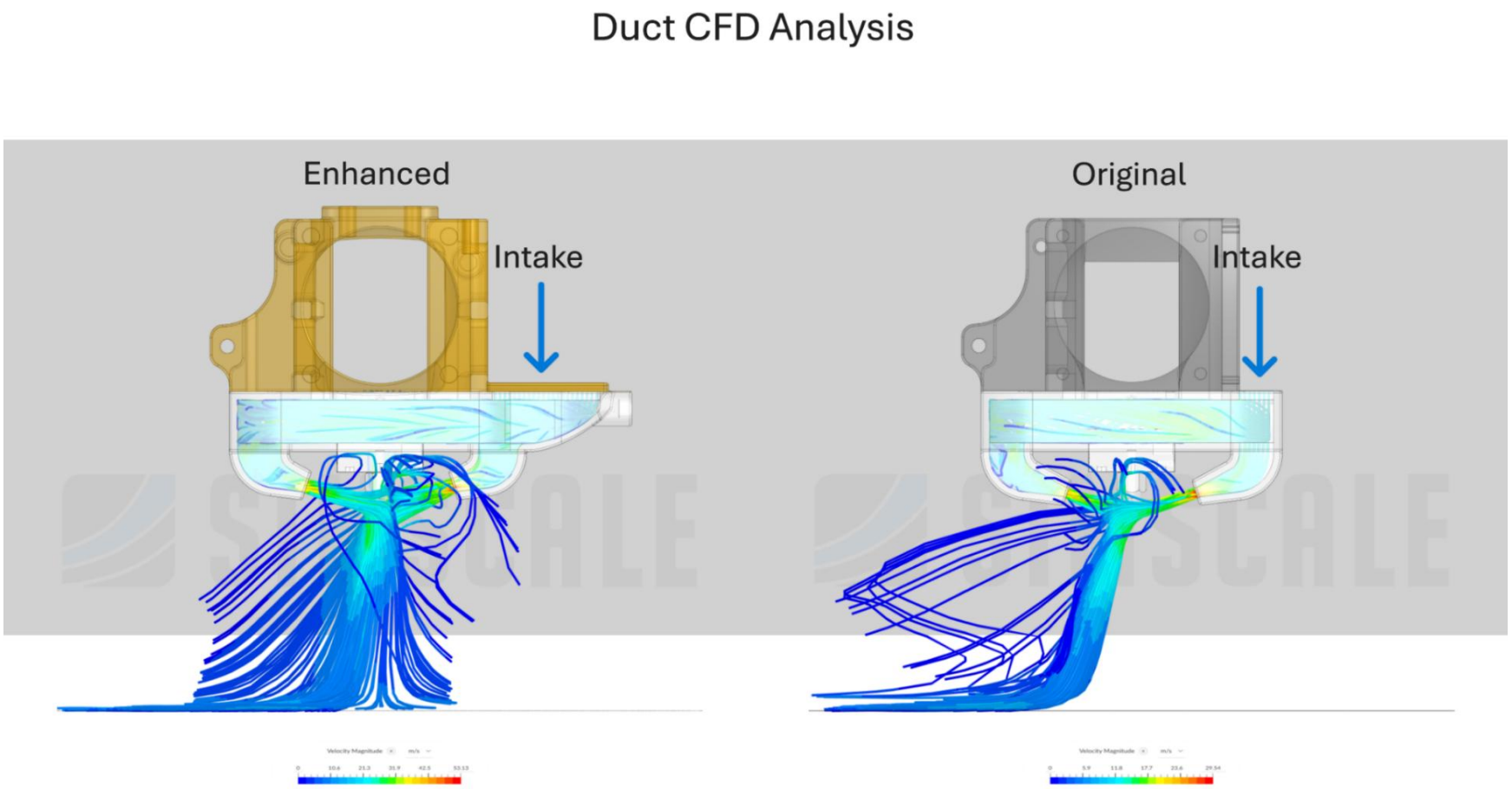


Figure 21: Shows SimScale CFD analysis of the enhanced and original print head shroud.

Appendix D: Electrical Enclosure Explode-View

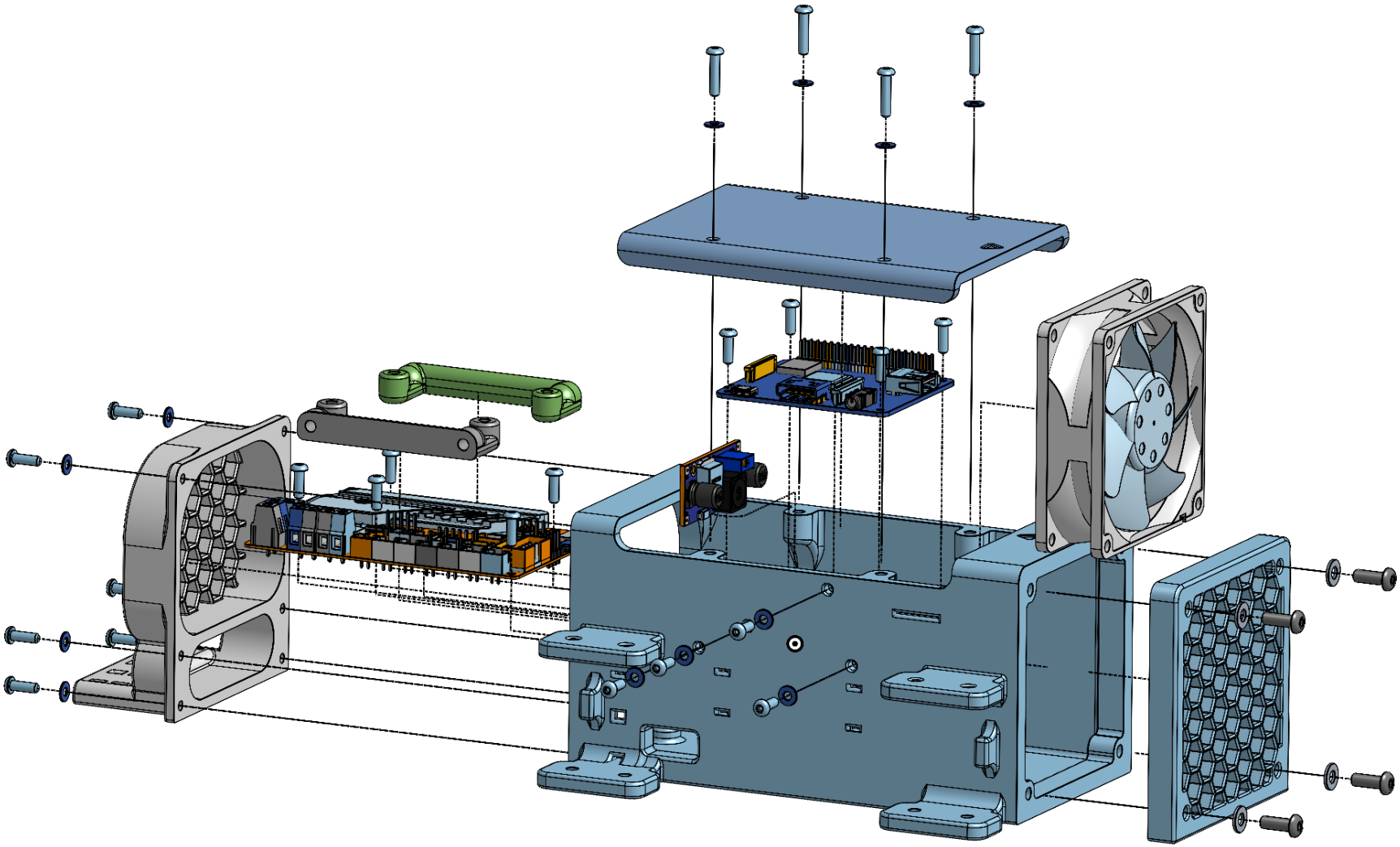


Figure 22: Shows explode view of assembled the electrical enclosure.

Appendix E: Electrical Schematic (KiCad)

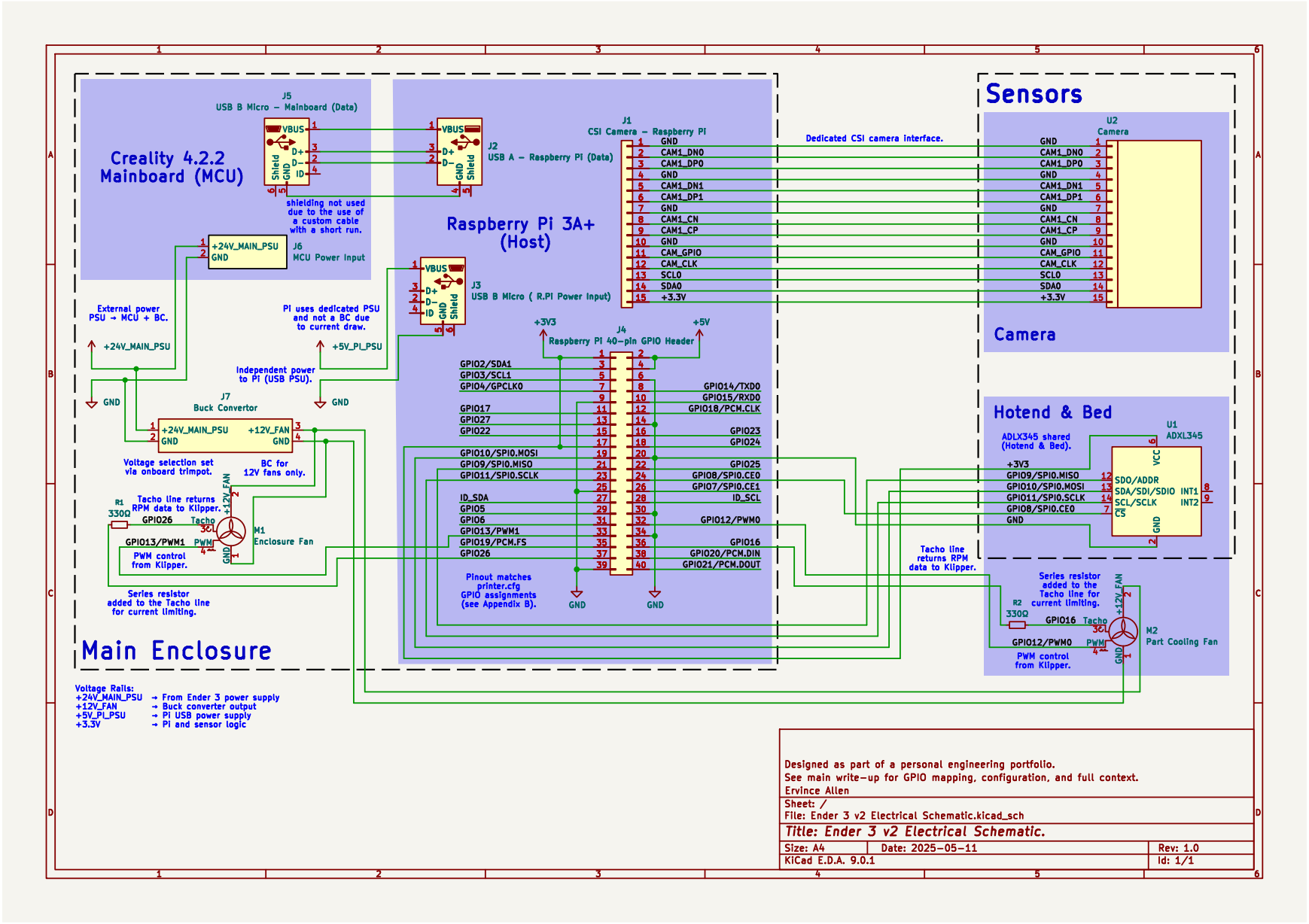


Figure 23: Shows enhanced printers electrical schematic.

Appendix F: Electrical Enclosure Components

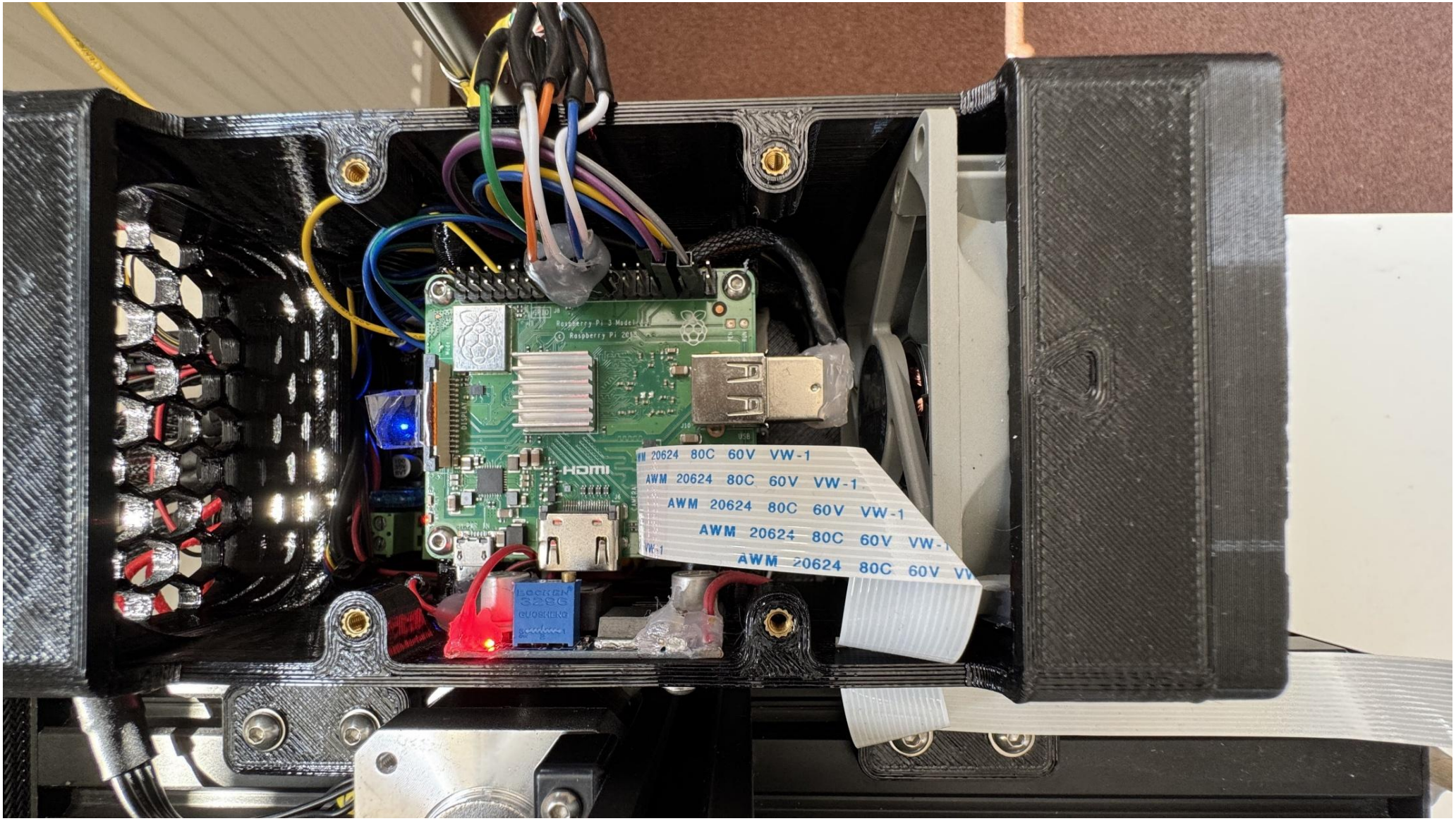


Figure 24: Shows photograph of the components in the electrical enclosure.

Appendix G: Fan Control Logic Flowchart

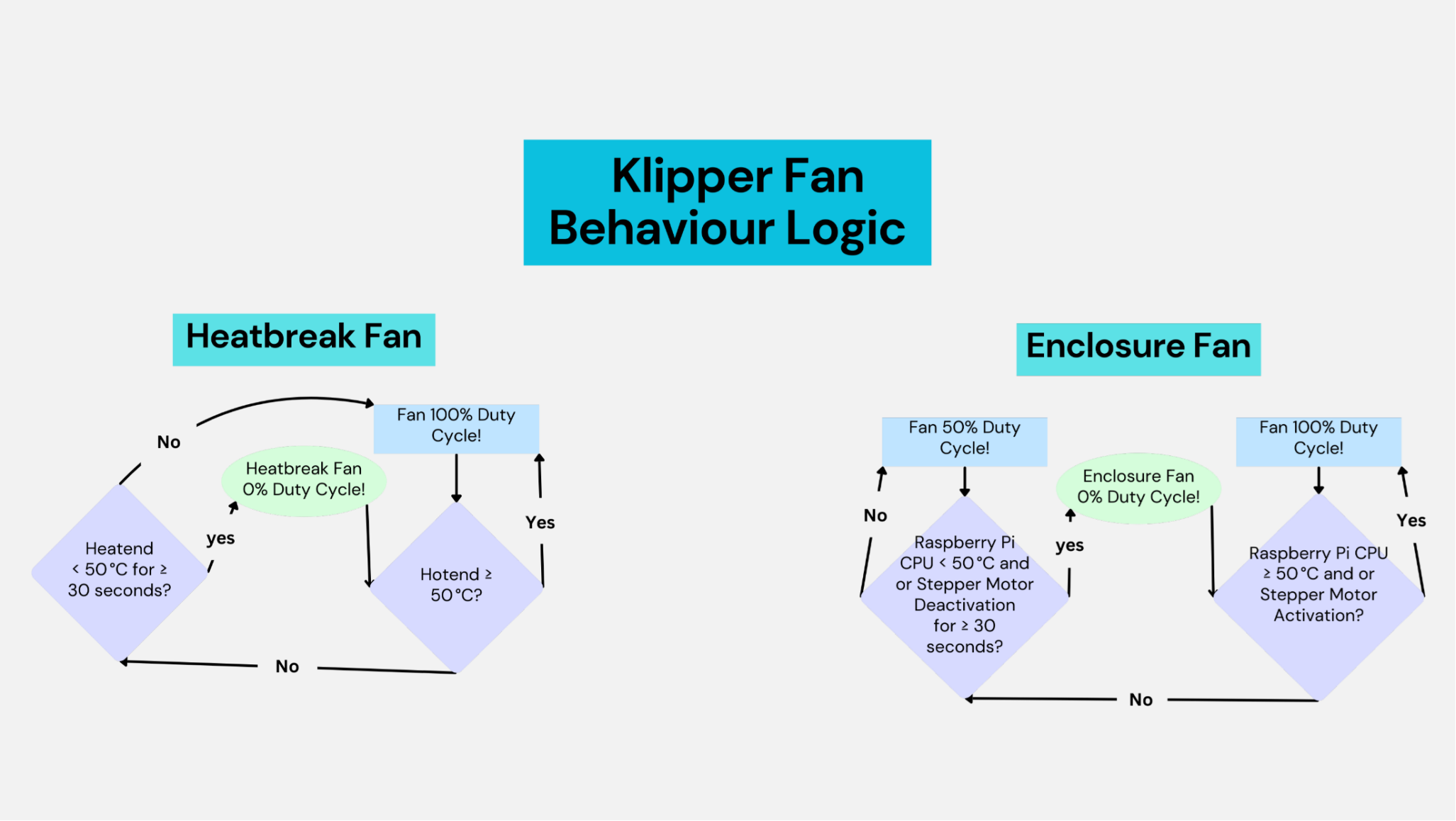


Figure 25: Shows fan logic for the heat break and electrical enclosure.

Appendix H: X-Axis Input Shaper Graph

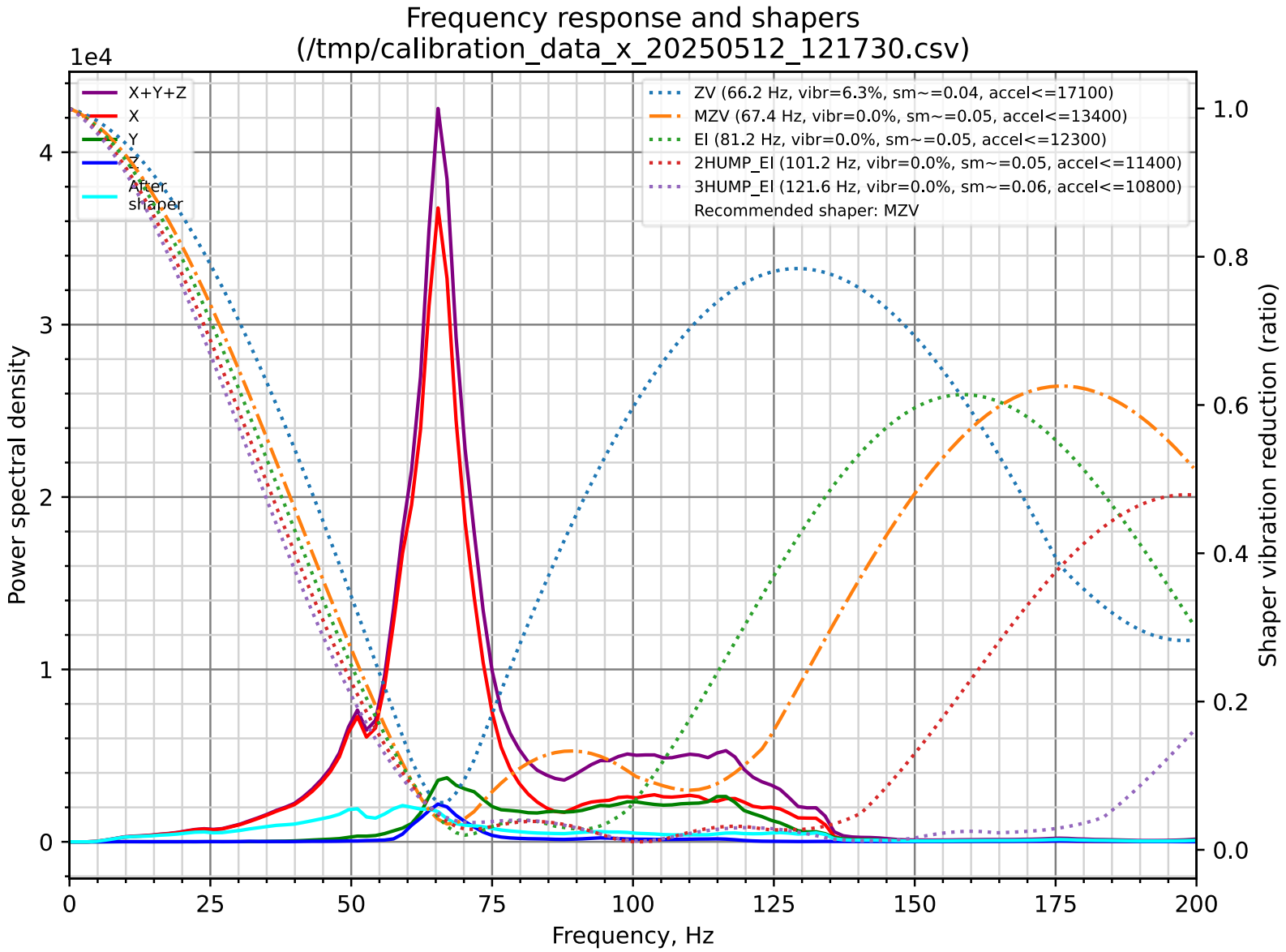


Figure 26: Shows Input Shaper test results for X and Y axis of the print head.

Appendix I: Y-Axis Input Shaper Graph

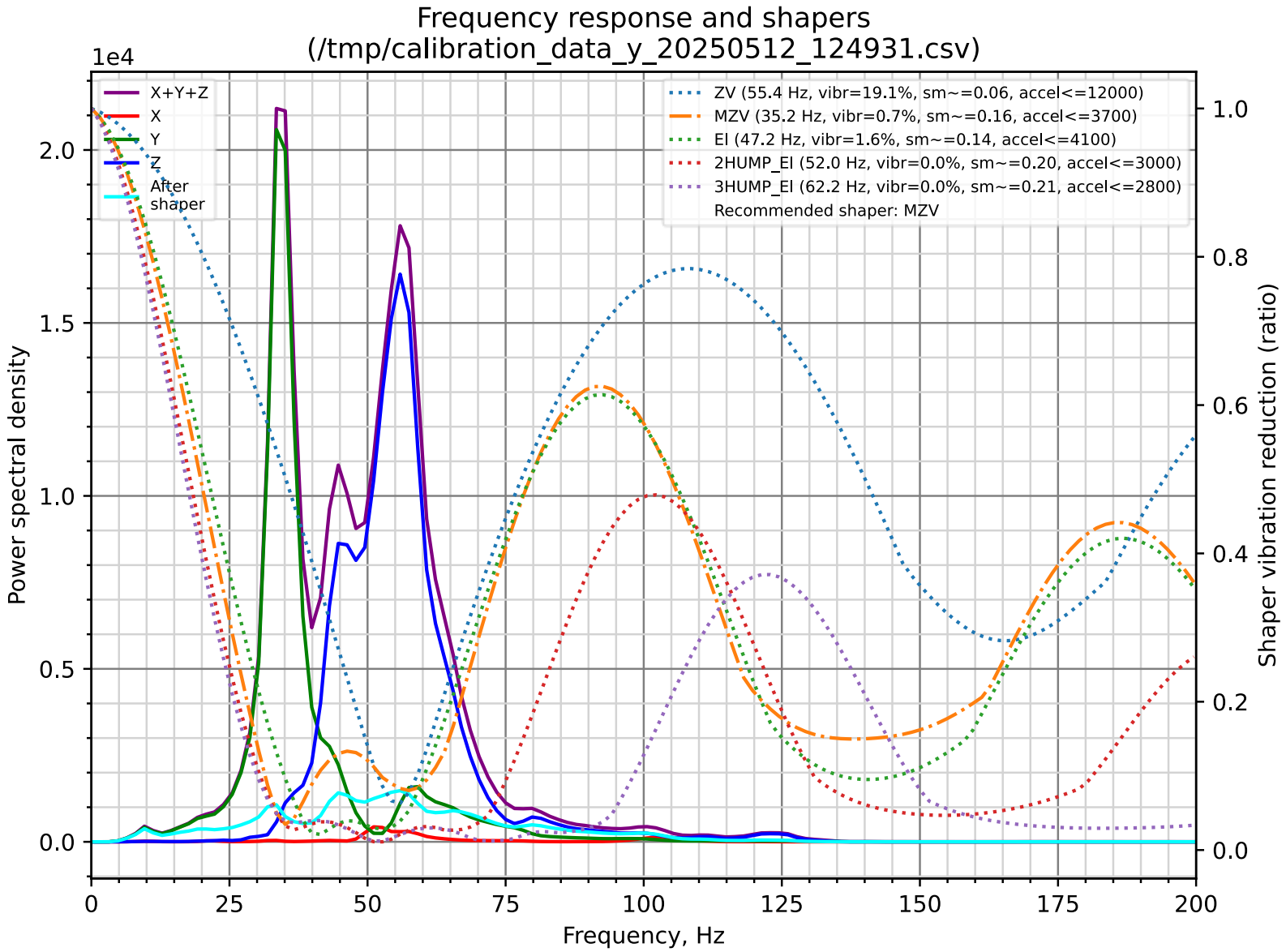


Figure 27: Shows Input Shaper test results for Y-axis of the build plate.

Appendix J: Rear Vent Technical Drawing.

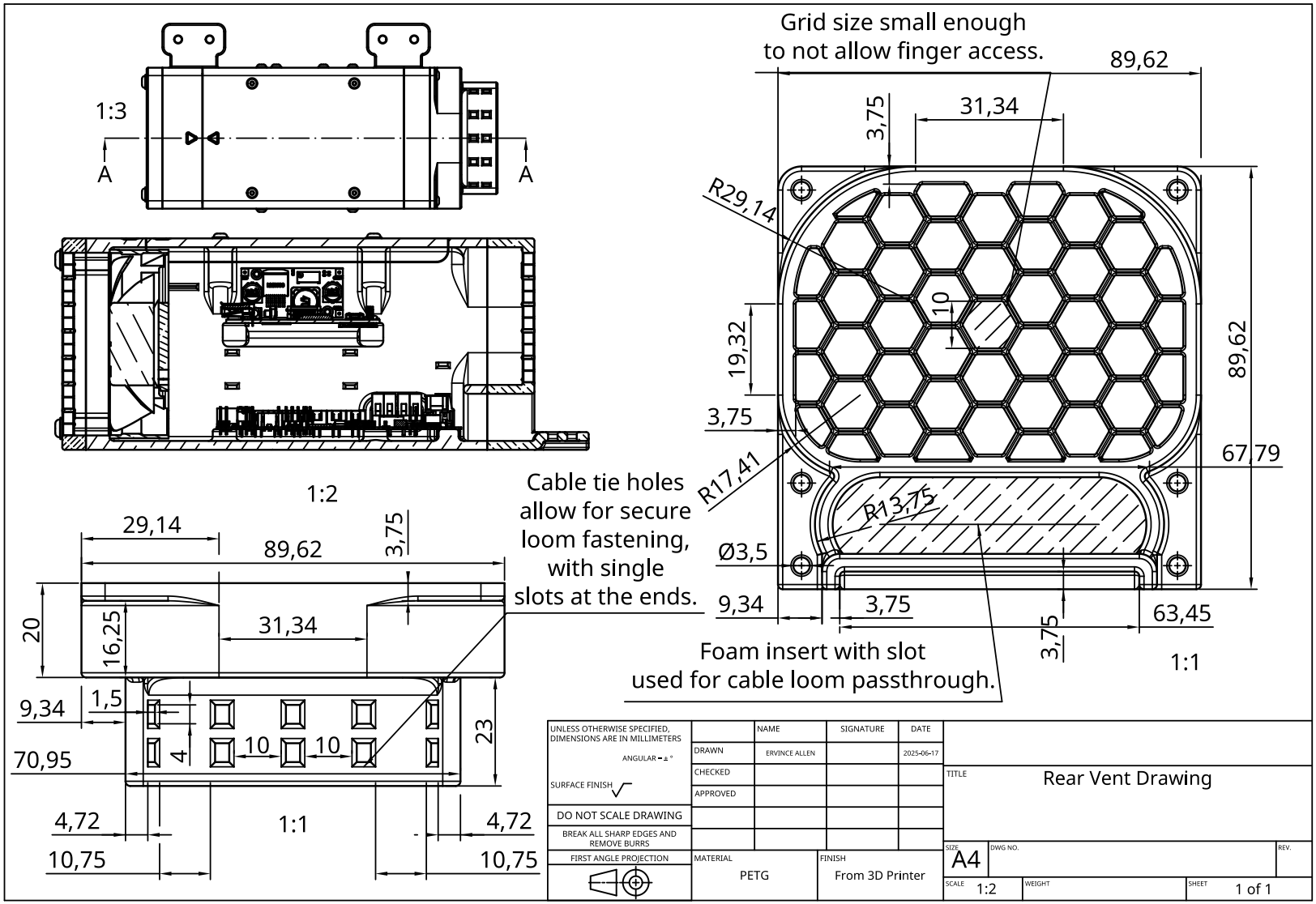


Figure 28: Shows technical drawing of the rear vent.

Appendix Y: Start & End G-Code

```
[gcode_macro START_PRINT]
gcode:
# Start bed heating
#M140 S75
# Use absolute coordinates
G90
# Reset the G-Code Z offset
(adjust Z offset if needed)
SET_GCODE_OFFSET Z=0.0
# Home the printer
G28
# Wait for bed to reach
temperature
#M190 S75
BED_MESH_PROFILE LOAD=default
# Set and wait for nozzle to
reach temperature
#M104 S250
#M109 S250
# Move to wait position
G1 X0 Y0 Z30 F4000.0
# Move Z-axis up
G1 Z2.0 F3000
# Move to start position
G1 X10.1 Y20 Z0.28 F5000.0
# Reset extruder
G92 E0
# Draw the first line
G1 X10.1 Y200.0 Z0.28 F1500.0 E15
# Move to the side
G1 X10.4 Y200.0 Z0.28 F5000.0
# Draw the second line
G1 X10.4 Y20 Z0.28 F1500.0 E30
# Reset extruder
G92 E0
```

```
[gcode_macro END_PRINT]
gcode:
# Finish moves
M400
# Set positioning to relative
G91
# Retract the filament by 18 mm
G1 E-18 F800
# Retract the filament more and
raise Z-axis
G1 E-2 Z0.2 F2400
# Set positioning to absolute
G90
# Move the bed forward
G1 X0 Y220 F1000
# Turn off the hotend
M104 S0
# Turn off the bed
M140 S0
# Turn off the fan
M106 S0
# Disable steppers
M84
M84
```

Appendix Z: Printer Configuration

Source and Modifications

This configuration file began as Klipper's official printer-creality-ender3-v2-2020.cfg template and was adapted for this custom Ender 3 V2 rebuild. Many parameters were modified for hardware compatibility and performance tuning, while others were added to support advanced features such as input shaping and mesh levelling.

Section / Parameter	Klipper Default Value	Custom Value or Addition
position_max (X)	235	240
position_max (Y)	235	230
position_min (Z)	–5	–2.176
nozzle_diameter	0.400	0.600
pressure_advance	*Not present*	0.6
[bltouch]	*Not present*	Added (sensor setup and offsets)
[safe_z_home]	*Not present*	Added (BLTouch homing setup)
[bed_mesh]	*Not present*	Added (5×5 grid)
[screws_tilt_adjust]	*Not present*	Manual bed levelling configuration
[fan], [controller_fan]	*Not present*	GPIO-controlled fans + tachometer
[heater_fan Heatbreak_Fan]	*Not present*	Fan tied to extruder temperature
[adxl345], [resonance_tester]	*Not present*	Added for vibration testing
[input_shaper]	*Not present*	Added (based on calibration results)

Summary

This configuration reflects a tailored firmware setup that retains the structure of Klipper's defaults while layering on targeted modifications to accommodate new hardware, improve print quality, and enable advanced features. The file has evolved iteratively through testing and calibration during the rebuild.

```
# This file contains pin mappings for the stock 2020 Creality Ender 3
# V2. To use this config, during "make menuconfig" select the
# STM32F103 with a "28KiB bootloader" and serial (on USART1 PA10/PA9)
# communication.

# If you prefer a direct serial connection, in "make menuconfig"
# select "Enable extra low-level configuration options" and select
# serial (on USART3 PB11/PB10), which is broken out on the 10 pin IDC
# cable used for the LCD module as follows:
# 3: Tx, 4: Rx, 9: GND, 10: VCC

# Flash this firmware by copying "out/klipper.bin" to a SD card and
# turning on the printer with the card inserted. The firmware
# filename must end in ".bin" and must not match the last filename
# that was flashed.

# See docs/Config_Reference.md for a description of parameters.

[include mainsail.cfg]

[include macro.cfg]

[stepper_x]
step_pin: PC2
dir_pin: PB9
enable_pin: !PC3
microsteps: 16
rotation_distance: 40
endstop_pin: ^PA5
position_endstop: 0
position_max: 240
homing_speed: 50

[stepper_y]
step_pin: PB8
dir_pin: PB7
enable_pin: !PC3
microsteps: 16
rotation_distance: 40
endstop_pin: ^PA6
position_endstop: 0
position_max: 230
homing_speed: 50

[stepper_z]
step_pin: PB6
dir_pin: !PB5
enable_pin: !PC3
microsteps: 16
rotation_distance: 8
endstop_pin:
probe:z_virtual_endstop
#position_endstop: -0.1
position_max: 250
position_min: -2.176

[bltouch]
sensor_pin: ^PB1
control_pin: PB0
x_offset: -44
y_offset: -6

[safe_z_home]
home_xy_position: 117.5,117.5
z_hop: 10
z_hop_speed: 10

[bed_mesh]
speed: 120
horizontal_move_z: 5
mesh_min: 15,15
mesh_max: 188,191
probe_count: 5,5
algorithm: bicubic
fade_start: 1
fade_end: 10
fade_target: 0

[screws_tilt_adjust]
screw1: 74, 42
screw1_name: Front left
screw2: 235, 42  # Adjusted to
fit within the bed size
screw2_name: Front right
screw3: 235, 213  # Adjusted to
fit within the bed size
screw3_name: Back right
screw4: 74, 213
screw4_name: Back left
screw_thread: CW-M4
```

```
horizontal_move_z: 10
speed: 200

[extruder]
max_extrude_only_distance: 100.0
step_pin: PB4
dir_pin: PB3
enable_pin: !PC3
microsteps: 16
rotation_distance: 34.406
nozzle_diameter: 0.600
filament_diameter: 1.750
heater_pin: PA1
sensor_type: EPCOS 100K
B57560G104F
sensor_pin: PC5
#control: pid
# tuned for stock hardware with
200 degree Celsius target
#pid_Kp: 21.527
#pid_Ki: 1.063
#pid_Kd: 108.982
min_temp: 0
max_temp: 260
pressure_advance: 0.6

[heater_bed]
heater_pin: PA2
sensor_type: EPCOS 100K
B57560G104F
sensor_pin: PC4
control: pid
# tuned for stock hardware with
50 degree Celsius target
pid_Kp: 54.027
pid_Ki: 0.770
pid_Kd: 948.182
min_temp: 0
max_temp: 130

[temperature_sensor MCU]
sensor_type: temperature_mcu

[temperature_sensor RPI]
sensor_type: temperature_host

[fan]
pin: rpi:gpio12
max_power: 1.0
shutdown_speed: 0
cycle_time: 0.010
hardware_pwm: False
kick_start_time: 0.100
off_below: 0.0
tachometer_pin: ^rpi:gpio16
tachometer_ppr: 2
tachometer_poll_interval: 0.0015

[controller_fan Controller_Fan]
pin: rpi:gpio13
max_power: 1.0
shutdown_speed: 0.0
cycle_time: 0.010
hardware_pwm: False
kick_start_time: 0.100
off_below: 0.0
tachometer_pin: ^rpi:gpio26
tachometer_ppr: 2
tachometer_poll_interval: 0.0015
#enable_pin:
#   See the "fan" section for a
description of the above
parameters.
fan_speed: 1.0
#   The fan speed (expressed as a
value from 0.0 to 1.0) that the
fan
#   will be set to when a heater
or stepper driver is active.
#   The default is 1.0
idle_timeout: 30
#   The amount of time (in
seconds) after a stepper driver
or heater
#   was active and the fan should
be kept running. The default
#   is 30 seconds.
idle_speed: 0.5
#   The fan speed (expressed as a
value from 0.0 to 1.0) that the
fan
#   will be set to when a heater
or stepper driver was active and
#   before the idle_timeout is
reached. The default is
fan_speed.
heater: heater_bed, extruder
#stepper: stepper_x, stepper_y,
stepper_z, extruder
#   Name of the config section
defining the heater/stepper that
this fan
#   is associated with. If a
comma separated list of
heater/stepper names
#   is provided here, then the
fan will be enabled when any of
the given
```

```
#   heaters/steppers are enabled.
The default heater is "extruder",
the
#   default stepper is all of
them.

[heater_fan Heatbreak_Fan]
pin: PA0
#max_power:
#shutdown_speed:
#cycle_time:
#hardware_pwm:
kick_start_time: 0.5
#off_below:
#tachometer_pin:
#tachometer_ppr:
#tachometer_poll_interval:
#enable_pin:
#   See the "fan" section for a
description of the above
parameters.
heater: extruder
#   Name of the config section
defining the heater that this fan
is
#   associated with. If a comma
separated list of heater names is
#   provided here, then the fan
will be enabled when any of the
given
#   heaters are enabled. The
default is "extruder".
heater_temp: 50.0
#   A temperature (in Celsius)
that the heater must drop below
before
#   the fan is disabled. The
default is 50 Celsius.
#fan_speed: 1.0
#   The fan speed (expressed as a
value from 0.0 to 1.0) that the
fan
#   will be set to when its
associated heater is enabled. The
default
#   is 1.0

[mcu]
serial: /dev/serial/by-id/usb-
1a86_USB_Serial-if00-port0
restart_method: command

[mcu rpi]
serial: /tmp/klipper_host_mcu

[adxl345]
cs_pin: rpi:None

[resonance_tester]
accel_chip: adxl345
probe_points: 117.5, 117.5, 20

[input_shaper]
#shaper_freq_x: 70.0
#shaper_type_x: mzv
#shaper_freq_y: 38.2
#shaper_type_y: mzv

[printer]
kinematics: cartesian
max_velocity: 300
max_accel: 3500
max_z_velocity: 25
max_z_accel: 500

#*# <----------------------
SAVE_CONFIG ----------------------
->
#*# DO NOT EDIT THIS BLOCK OR
BELOW. The contents are auto-
generated.
#*#
#*# [bltouch]
#*# z_offset = 2.350
#*#
#*# [bed_mesh default]
#*# version = 1
#*# points =
#*#  0.067500, 0.072500,
0.072500, 0.072500, 0.092500
#*#  0.057500, 0.045000,
0.037500, 0.037500, 0.047500
#*#  0.050000, 0.035000,
0.015000, 0.010000, 0.012500
#*#  0.067500, 0.042500,
0.025000, 0.005000, -0.007500
#*#  0.065000, 0.047500,
0.020000, 0.020000, 0.032500
#*# x_count = 5
#*# y_count = 5
#*# mesh_x_pps = 2
#*# mesh_y_pps = 2
#*# algo = bicubic
#*# tension = 0.2
#*# min_x = 15.0
#*# max_x = 188.0
#*# min_y = 15.0
#*# max_y = 191.0
```

```
#*#
#*# [extruder]
#*# control = pid
#*# pid_kp = 30.322
#*# pid_ki = 1.743
#*# pid_kd = 131.901
#*#
```

```
#*# [input_shaper]
#*# shaper_type_x = zv
#*# shaper_freq_x = 67.4
#*# shaper_type_y = mzv
#*# shaper_freq_y = 36.8
```