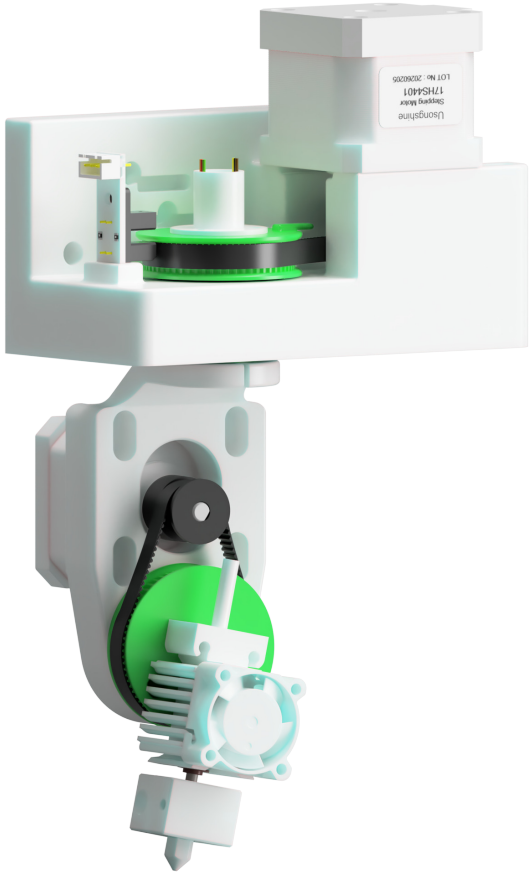
DEMOCRATISING MULTI-AXIS 3D PRINTING

Making an open-source system to bring multi-axis 3D printing to everyday makers



AUTHOR

Dennis Klappe
s2400758

GRADUATION COMMITTEE

Dr.ir. T. Vaneker (chair)
Ir. J. Andersons (supervisor)
Dr. M. Pereira (external member)

PROGRAMME

MSc Industrial Design Engineering
DPM, Faculty of ET

PLACE & DATE

Enschede, 2 July 2026

CONTENTS

ABSTRACT

Multi-axis 3D printing promises support-free overhangs, smoother curved surfaces, and parts whose layers follow their load paths. Yet it stays confined to research labs and industry. The technology isn't the obstacle: earlier work already showed that an affordable five-axis retrofit of a consumer FDM printer works. The obstacle is accessibility. A working prototype existed, but reproducing it meant commercial software, specialist knowledge, and far more effort than the makers it was meant for could reasonably take on.

This thesis asks why a working prototype isn't being reproduced, and what it takes to change that. Following a Design Science Research approach, the adoption barriers are diagnosed through Rogers' diffusion of innovations and the open-source-hardware reproducibility literature, then addressed directly. The result is Rep5x: a five-axis retrofit redesigned to adapt to several common desktop printers; a Marlin fork with native five-axis kinematics; a suite of browser-based tools that replace the command line and commercial licences with no-install workflows for firmware, setup, printer control, and G-code; and the documentation, website, and community that let someone actually build it, all released under GPLv3.

The system is evaluated through a pre-build survey, community-driven iteration, and independent build attempts. The finding is that the barrier to multi-axis FDM isn't the mechanism. It's the toolchain and the infrastructure around it. Removing the specialist-software dependency, packaging the workflow for non-experts, and building a community where feedback drives the design are what turn a working prototype into something independent makers can reproduce.

ACKNOWLEDGEMENTS

This thesis builds directly on the work of Janis Andersons, my supervisor, and wouldn't exist without it. I'm grateful for that, but just as much for how welcome he made me feel from the start, gave me the room to take the project where it needed to go, and stayed genuinely interested in it the whole way through.

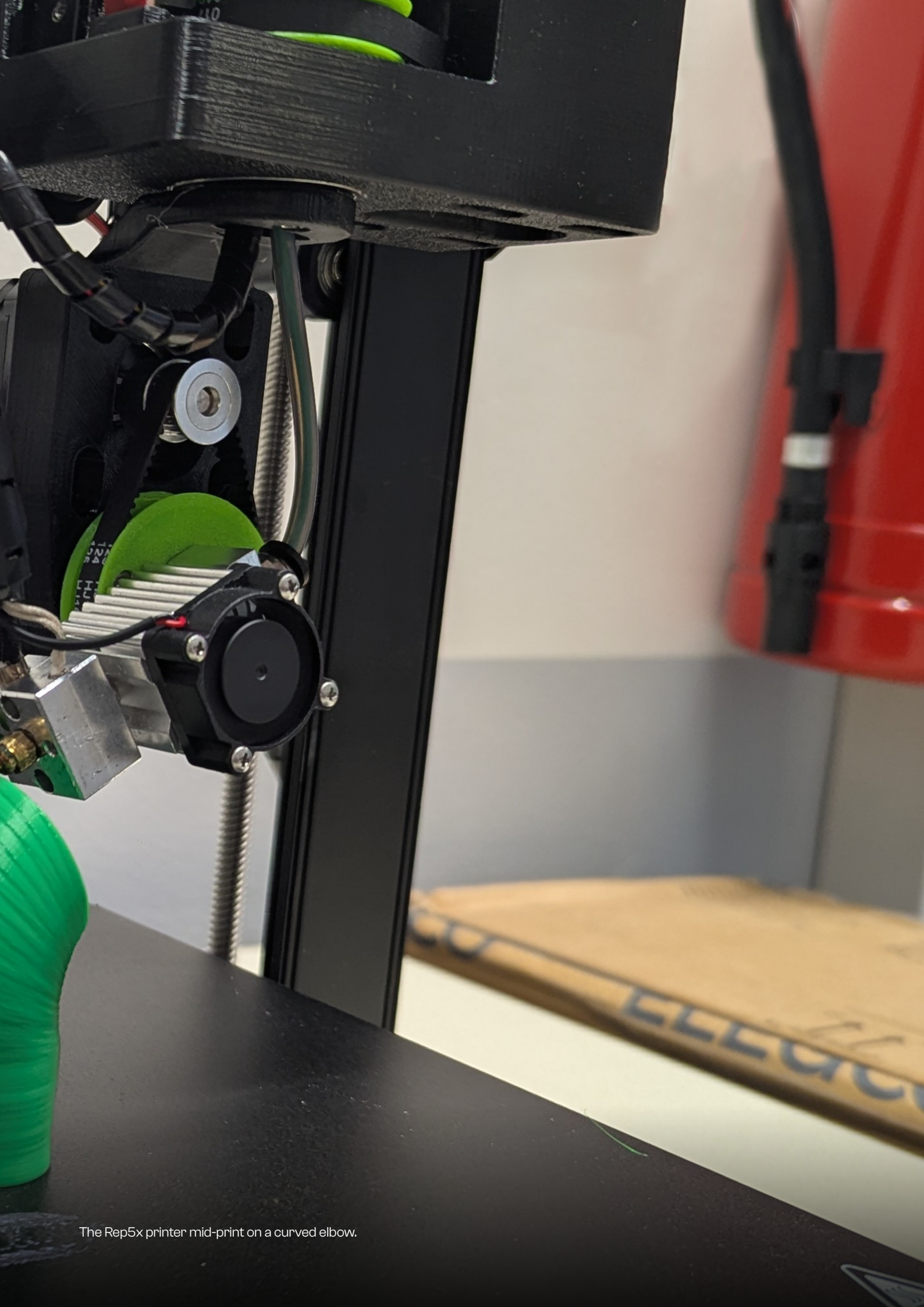
Rieks and Laura were in the lab with me almost every day for the past nine months. We each worked on our own project, but we sparred together and made sure none of us went crazy in the Horst. It made the long days a lot easier.

Thanks also to the members of the Rep5x Discord, who answered questions, gave input on the project, and were always up for a discussion. Some went further and contributed directly, and the project is better for it.

Finally, thanks to my girlfriend Eline, who dealt with me being a lot less available than I'd have liked. And to my friends and family, for putting up with me talking about nothing but 3D printers for the better part of a year.

ABBREVIATIONS

AM	Additive manufacturing
API	Application programming interface
BOM	Bill of materials
CAD	Computer-aided design
CAM	Computer-aided manufacturing
CNC	Computer numerical control
COTS	Commercial off-the-shelf
CRM	Customer relationship management
DLP	Digital light processing
DoF	Degrees of freedom
DSR	Design Science Research
DSRM	Design Science Research Methodology
FDM	Fused deposition modelling
FFF	Fused filament fabrication
GPL	GNU General Public License
GUI	Graphical user interface
HH	Head-head (kinematic configuration)
HT	Head-table (kinematic configuration)
IK	Inverse kinematics
JSON	JavaScript Object Notation
LB	B-axis offset: distance from the B-axis (tilt) rotation centre to the nozzle tip
LC	C-axis offset: distance from the C-axis (yaw) rotation centre to the nozzle tip
NEMA	National Electrical Manufacturers Association (stepper frame size, e.g. NEMA 17)
OCL	Open Community Licence
OHL	CERN Open Hardware Licence
OSHW	Open-source hardware
OSHWA	Open Source Hardware Association
OSS	Open-source software
PETG	Polyethylene terephthalate glycol
PLA	Polylactic acid
RMS	Root mean square
SLA	Stereolithography
SLS	Selective laser sintering
SM	Subtractive manufacturing
STEP	Standard for the Exchange of Product model data
STL	Standard tessellation language (3D model file format)
TT	Table-table (kinematic configuration)
UART	Universal asynchronous receiver-transmitter
UI	User interface
UX	User experience



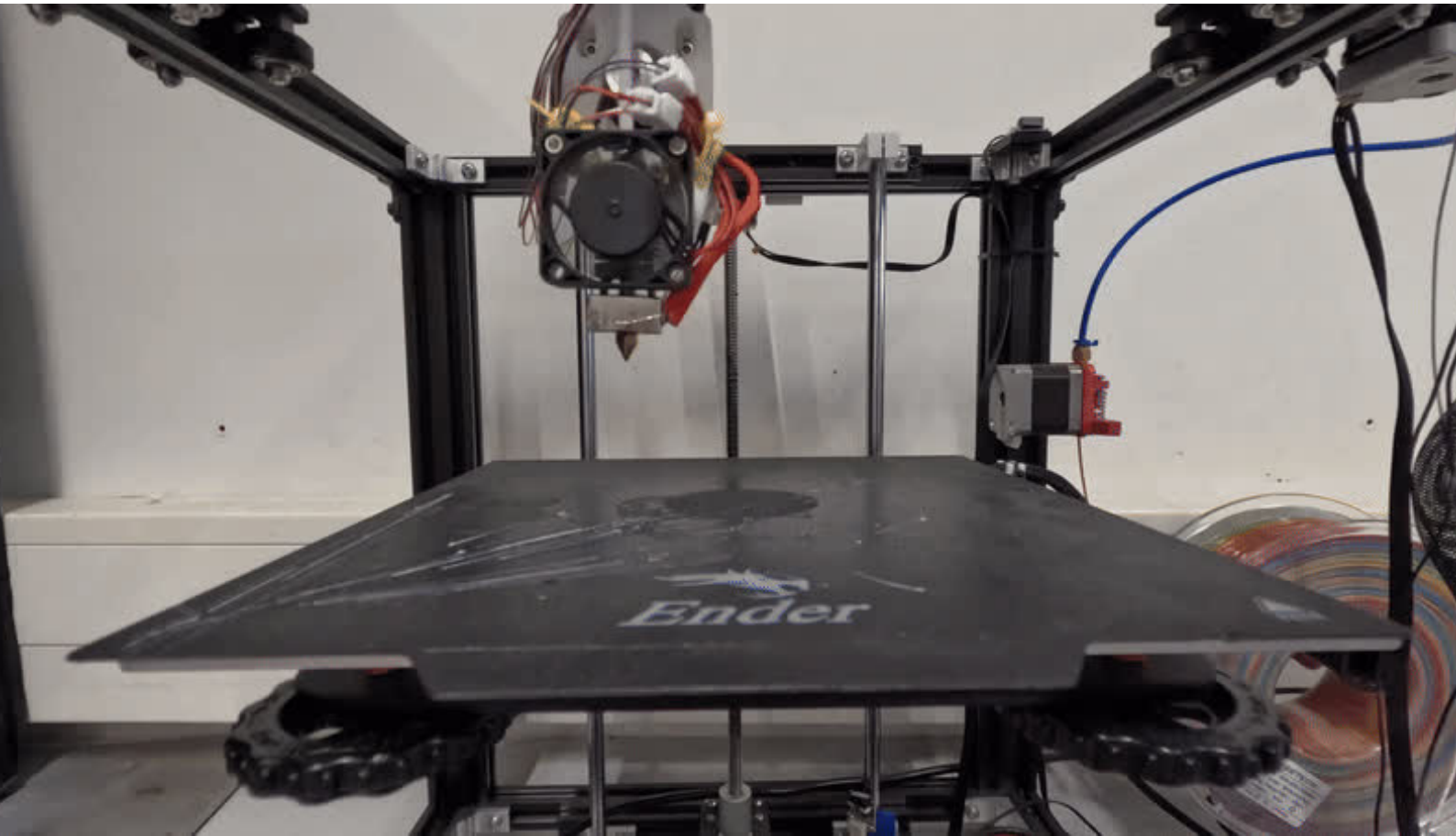
The Rep5x printer mid-print on a curved elbow.

CHAPTER 01

01

INTRODUCTION

This chapter covers the basics of additive manufacturing and FDM, the role of open-source hardware, the limitations of conventional FDM that multi-axis printing addresses, and the prior work by Andersons that this research builds on. It closes with the problem statement, research questions, and scope.



Additive manufacturing has transformed from an expensive industrial process into something that everyone can use. Fused Deposition Modelling (FDM) printers can be bought for under €200, and FDM is the most popular AM process [1]. But there are limitations.

These printers deposit material in flat layers, one on top of the other, and this planar constraint causes well-known problems: support structures for overhanging geometry, visible staircase effects on curved surfaces, and mechanical weakness between layers. Multi-axis printing addresses all three by adding rotational degrees of freedom. Yet multi-axis printers aren't widespread, remaining confined to research labs and industrial settings.

The problem isn't that multi-axis printing doesn't work. The problem is that the technology still isn't accessible to the general public. The hardware is expensive, the software requires specialist knowledge, and the documentation doesn't exist. This thesis is about changing this.

This chapter provides the background needed to understand the problem. It covers the basics of **additive manufacturing** and **FDM**, the role of **open-source hardware** in making technology accessible, the **limitations of conventional FDM** that multi-axis printing addresses, and the **prior work** by Andersons that this research builds on. It concludes with the **problem statement**, **research questions**, and **scope**.

1.1 BACKGROUND

Additive manufacturing

History and growth. Additive Manufacturing, generally known as 3D printing, is the process of fabricating a part layer by layer [1]. Material is added incrementally, in contrast to subtractive manufacturing (SM), where material is removed from a solid block. The first patent for what could be considered an AM technology was granted in 1971, for depositing molten metal through a nozzle onto a carrier. Developments on stereolithography (SLA) followed in the 1980s, when Charles Hull patented a tool for curing successive layers of photopolymer with UV light [2]. Within a few years, several additional AM processes were identified, now compartmentalised into seven general process categories by the ISO/ASTM 52900 standard [3]: material extrusion, vat photopolymerisation, powder bed fusion, material jetting, binder jetting, directed energy deposition, and sheet lamination.

S. Scott Crump invented Fused Deposition Modelling and patented it through Stratasys in 1992 [4], with the original mechanism shown in Fig. 1.1. For most of its history, AM was an industrial technology. Machines cost hundreds of thousands of euros and required trained operators. They were used primarily for rapid prototyping, making physical models to check designs before committing to expensive tooling.

That changed in the late 2000s. Stratasys' FDM patent expired after its 20-year term around 2009. Combined with the rise of open-source hardware initiatives like RepRap [6], this made AM accessible to consumers. The AM industry has since grown into a multi-billion dollar market [1]. FDM has become the most commonly used

FIG. 1.1 The origins of FDM: S. Scott Crump's 1992 patent drawing [4], and the Stratasys 3D Modeler, the first commercial FDM machine [5].

FIG. 1.2 Three common consumer FDM printers: the Creality Ender 3 V3 SE [7], the Bambu Lab A1 mini [8], and the Prusa MK4S [9].

process type, widely adopted for both prototyping and end-use parts [1]. The current consumer landscape is wide, with capable machines available from under €200 (Fig. 1.2).

1.2 FUSED DEPOSITION MODELLING

Working principles

Fused Deposition Modelling, or Fused Filament Fabrication (FFF) to avoid trademark issues with Stratasys, is an AM process based on material extrusion. A heated nozzle (the *hot-end*) melts a thermoplastic filament and deposits it on a bed in thin beads, as shown in Fig. 1.3. The nozzle follows a predetermined path, completing one layer at a time. The bed (or nozzle) moves one layer height in the Z-direction once each layer is finished, and the next layer is deposited on top of the previous one [1].

An FDM printer typically consists of three linear axes (X, Y, and Z), a heated bed, and an extruder. A *slicer* takes a 3D model (typically an STL file), cuts it into horizontal layers, and generates G-code for the printer. Cura, PrusaSlicer, and OrcaSlicer are widely used open-source slicers.

FDM works with a range of thermoplastic materials. PLA (polylactic acid) is the most common for hobbyists due to its ease of printing. PETG, ABS, TPU, nylon, and carbon-fibre-reinforced composites are also widely used, each requiring different print parameters for temperature, speed, cooling, and bed adhesion.

The open-source hardware movement

The RepRap project, initiated by Adrian Bowyer at the University of Bath (Fig. 1.4), made desktop 3D printing what it is today [10]. Today's consumer FDM ecosystem, from the €150 Ender 3 to the Prusa MK4 to community-driven designs like the Voron Trident, descends directly from it. The assumption isn't just that a design works, but that others can build it, change it, and contribute back. That's the environment in which Rep5x operates. The development model itself, what makes open-source hardware reproducible (or not), and the projects that succeeded or failed at it, are reviewed in the literature review.

Limitations of conventional FDM

Conventional 3-axis FDM printers deposit material exclusively in flat layers stacked along the Z-direction. This planar constraint leads to three well-known problems.

Support structures. Overhanging geometry can't be printed in mid-air. The slicer generates support structures under these features, which must be removed after printing. The commonly cited threshold is an overhang angle of approximately 45°, beyond which quality degrades and support material becomes necessary [12]. This wastes material, increases print time, and leaves surface marks where the supports were attached.

Staircase effect. Curved and angled surfaces exhibit visible layer lines, known as the "staircase effect." How pronounced it is depends on the layer height and the angle of the surface relative to the build direction. Smaller layer heights reduce the effect, but at the cost of proportionally longer print times [1].

Mechanical anisotropy. The bond between layers is weaker than the material within a layer. Zohdi et al. [13] report that mechanical anisotropy in FDM parts can reach nearly 50%, meaning the strength in the build direction can be roughly half of what it is in the XY plane (Fig. 1.5). This is the highest mechanical anisotropy among polymer AM methods, with comparable figures of around 10% for SLS, 5% for DLP and 1% for SLA [13], caused primarily by insufficient interlayer bonding and air voids between deposited filaments. In a standard 3-axis printer, the layer orientation is always perpendicular to the Z-axis, meaning the part's weakest direction is fixed regardless of where the loads are. There's no way to align the layer direction with the stress distribution of the part.

These three limitations constrain what FDM can reliably produce. And they all stem from the same root cause: the restriction to planar deposition.

1.3 MULTI-AXIS PRINTING

Principles and benefits

Adding two rotational axes changes this. By tilting and rotating the part (or the nozzle) during printing, the direction of deposition can be adjusted dynamically. Rather than exclusively stacking horizontal layers, the printer can

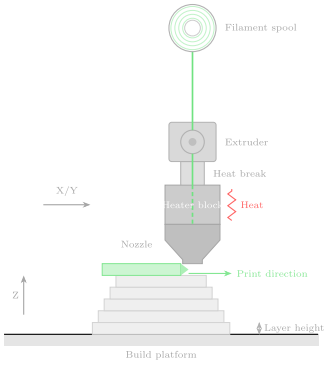


FIG. 1.3 Working principle of FDM: filament is melted in the hot-end and deposited bead by bead, one layer at a time.

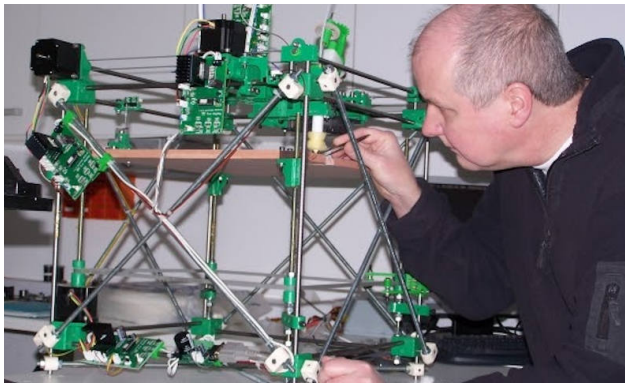


FIG. 1.4 Adrian Bowyer, who initiated the RepRap project at the University of Bath [11].

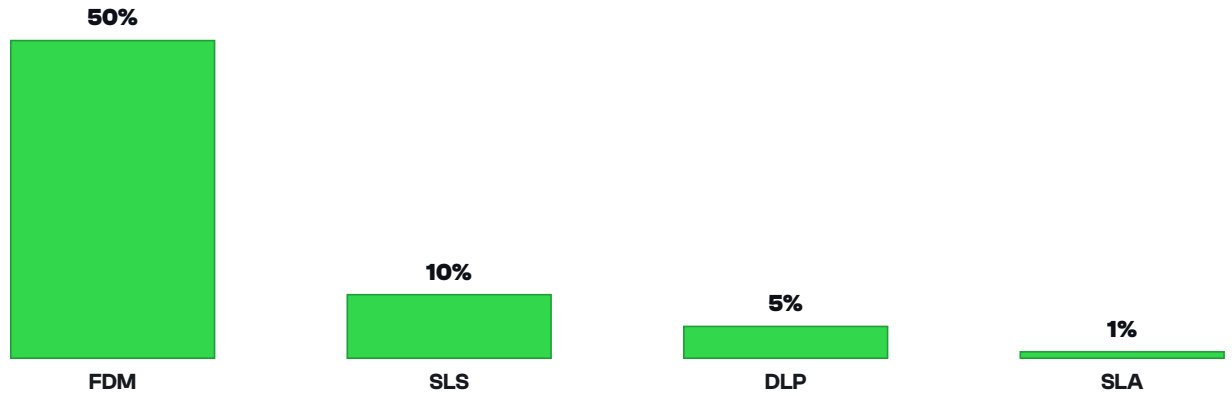


FIG. 1.5 Mechanical anisotropy across polymer AM methods. FDM reaches roughly 50% (build-direction strength about half of in-plane strength), much higher than SLS, DLP, and SLA [13].

deposit material at varying angles [14]. This opens up three possibilities.

Support-free printing. Overhangs that would normally require support can be printed directly by reorienting the part during printing. The nozzle approaches the surface from an angle where the overhang is no longer an overhang. This eliminates the material waste, time, and surface damage associated with support removal.

Improved surface quality. When the nozzle can follow curved surfaces rather than just horizontal passes, non-planar toolpaths can reduce or eliminate the staircase effect entirely, producing smoother surfaces without post-processing [14]. Fig. 1.7 shows an aerofoil printed with conformal layers that follow the surface curvature.

Controlled anisotropy. By changing the build direction during printing, layers can be aligned with the expected stress distribution of the part. This directly addresses the anisotropy problem by making sure the weak interlayer bonds are oriented away from the primary load paths.

Fig. 1.6 illustrates the difference. The same part printed with 3-axis planar layers requires support material under the overhang and shows visible staircase stepping on the curved surface. With 5-axis non-planar toolpaths, the layers follow the geometry, the overhang is printed directly by tilting the build direction, and no support is needed.

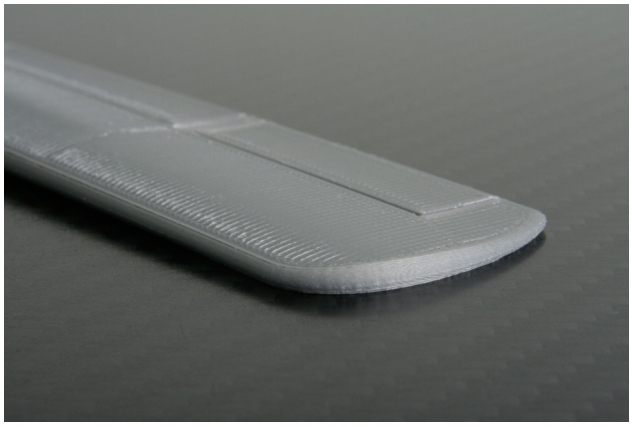


FIG. 1.7 An aerofoil printed with non-planar layers conforming to the wing surface [15].

The trade-off is complexity. Five-axis printing requires more advanced kinematics, more sophisticated toolpath planning, and calibration. But the benefits still make it worth the added complexity.

Current state

In industry, multi-axis AM isn't new. Systems using robotic arms have been depositing material along non-planar paths for years. These systems typically use six-axis industrial robots with extrusion heads, intended for large-scale applications such as boat hulls, architectural components,

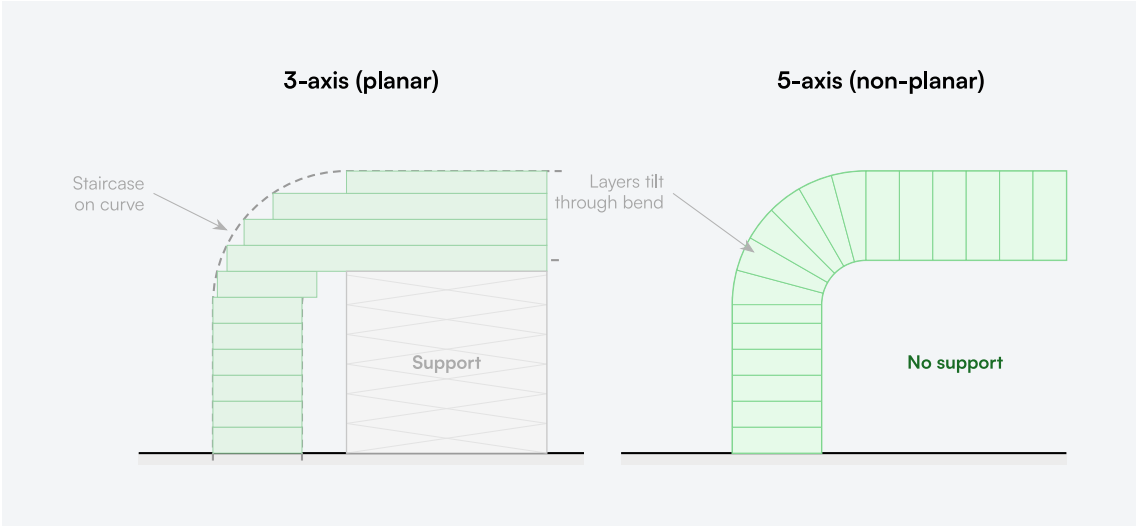


FIG. 1.6 The same 90° bent tube printed with 3-axis planar layers (left) and 5-axis conformal layers (right). Dashed lines show the intended tube profile.

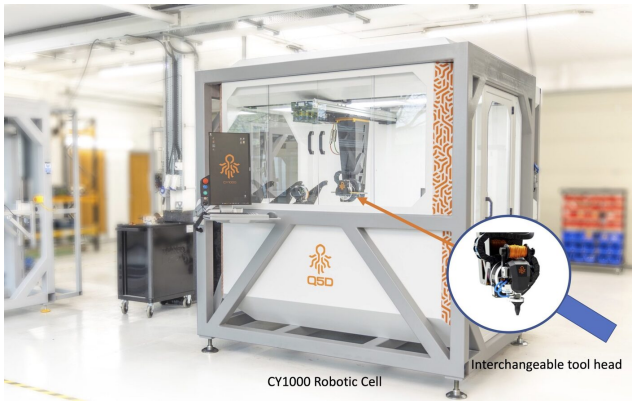


FIG. 1.8 Industrial multi-axis AM: a Caracol industrial robotic-arm 3D printer [16] and Q5D's CY1000 industrial 5-axis robotic cell [17].

or aerospace tooling (Fig. 1.8). Smaller industrial 5-axis cells, like the Q5D CY1000, target wire-harness automation and aerospace electronics.

Research groups have explored different kinematic configurations for desktop-scale multi-axis FDM. These are typically categorised into three groups based on where the two additional rotational axes are placed: head-head (HH), table-table (TT), and head-table (HT) [18]. Each has different trade-offs in workspace, complexity, and print quality; these are looked at in detail in the literature review.

What's missing is a solution that the average maker can actually use. Industrial systems cost tens of thousands of euros at minimum, require specialised software, and target professional users. Research prototypes are built in university labs and work, but nobody else can reproduce them. There is no accessible path for a maker to explore multi-axis printing.

Barriers to adoption

Rogers [19] identifies five attributes that determine whether a technology gets adopted: relative advantage, compatibility, complexity, trialability, and observability. Applying this framework to multi-axis FDM reveals that while the technical advantage is clear, the other four attributes are all problematic. Fig. 1.9 summarises five specific barriers that keep multi-axis printing from reaching the broader maker community: single-printer dependency, a software gap (no consumer-grade slicer, commercial dependencies), calibration difficulty, missing documentation, and the absence of community infrastructure. Together they create a chicken-and-egg problem. Without accessible hardware, there's no incentive to develop accessible software. Without accessible software, there's no motivation to build hardware. The framework behind these barriers is examined in detail in the literature review, and each barrier is broken down into concrete technical detail in the analysis of requirements.

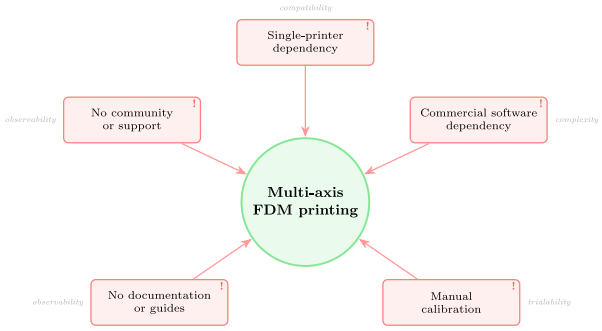


FIG. 1.9 Five adoption barriers identified for multi-axis FDM.

1.4 PRIOR WORK

Andersons' 5-axis retrofit prototype

In 2023, Andersons [18] completed a master's thesis at the University of Twente, supervised by the same research group that supervises this work. His thesis, *"Exploring the benefits and limitations of multi-axis 3D printing for improved part quality and reduced waste,"* looked into whether a consumer FDM printer could be converted into a 5-axis system.

Andersons designed a retrofit mechanism that adds two rotational axes to a Creality Ender 5 Pro. His final design, called the "Robot Wrist 2" (RW2), uses a head-head (HH) configuration: both rotational axes are placed at the nozzle, replacing the standard print head with a compact kinematic arrangement where the two axes intersect near the nozzle tip (Fig. 1.10). Andersons labels the yaw axis A and the tilt axis B. Rep5x re-labels the yaw axis as C to match standard 5-axis CNC conventions; the rationale is given in the design and development chapter. Throughout this thesis, Andersons' axes are referred to as A and B, and Rep5x's as C and B.

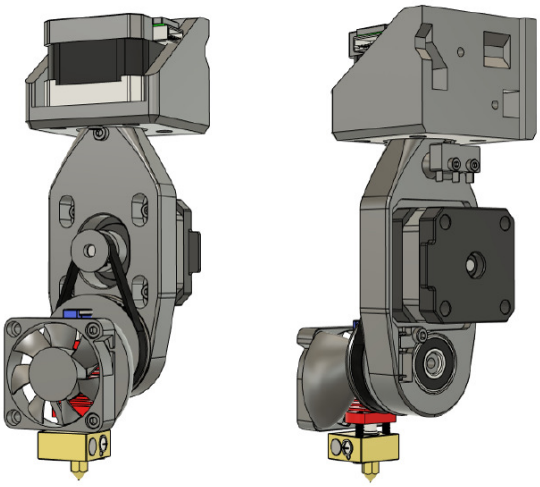


FIG. 1.10 Head-head (HH) 5-axis configuration: Andersons' Robot Wrist 2, with both rotational axes placed at the print head and intersecting near the nozzle tip [18].

The retrofit was designed with accessibility as a goal. Structural parts were 3D printed, the motors are conventional NEMA 17 steppers, and the total BOM cost was kept under €150. Andersons used a modified version of Marlin firmware to support the extra axes and developed inverse kinematics formulas to translate 5-axis coordinates into machine movements [18].

Findings and contributions

Andersons demonstrated several important results. Overhanging geometry was printed without support material, including tube-shaped parts with curves through 90° of arc. Surface quality of non-planar samples was superior to planar equivalents, maintaining low roughness (19—30 µm Sa) across the full range of surface angles, whereas planar samples degraded to 114 µm Sa at shallow surface angles. Dimensional accuracy was validated through 3D scanning: the range of deviation for non-planar samples was comparable to, and in some cases lower than, that of planar samples. Waste reduction was substantial, with the most extreme test cases showing material and print-time reductions in excess of 80% and 90% respectively [18].

His thesis also included a detailed survey of multi-axis AM hardware, benchmark parts, and path-planning algorithms. That technical groundwork doesn't need to be repeated here. The literature review takes a different approach: rather than focusing on the engineering, it examines open-source hardware practices, technology adoption theory, and community-driven development, the areas that explain why working technology doesn't get adopted and what it takes to change that.

On the software side, Andersons built a toolchain using Grasshopper (a visual scripting plugin for Rhinoceros 3D) for non-planar slicing and a Python script as a translation layer for inverse kinematics and G-code post-processing. Calibration was performed using a dial gauge for Z-axis errors and a reference cone for XY errors, with sinusoidal error functions fitted to the measured data for software compensation [18].

Remaining challenges

Andersons showed that the concept works. But by his own admission, the prototype was a first iteration, and several problems remained.

Reproducibility. The design was tested on a single printer in a university lab. There were no build guides, no assembly instructions, and no BOM formatted for independent reproduction. Anyone wanting to build the system would have to reverse-engineer it from the thesis. Antoniou et al. [20] surveyed 30 OSHW practitioners; 25 referred to documentation quality as a factor in project success. By that measure, and by the criteria of Bonvoisin et al. [21], the prototype simply wasn't packaged for someone else to rebuild, which is the gap this work sets out to close.

Software accessibility. The Grasshopper workflow requires a Rhinoceros 3D licence and experience with visual programming. The Python scripts were functional but not packaged in any user-friendly way.

Calibration complexity. The calibration process required precise instruments and did not have a flow of actually correcting the errors yet. This isn't something that a typical maker can be expected to do.

Single-platform design. The retrofit was designed for and tested on the Ender 5 Pro exclusively. Adapting it to other printers would require mechanical redesign, but no guidelines or parametric design tools were provided.

No community infrastructure. There was no website, no documentation platform, and no community forum. The work existed in isolation as an academic thesis.

These challenges don't undermine Andersons' work. Demonstrating that an affordable, open-source multi-axis retrofit is technically doable is a significant result. But feasibility and accessibility aren't the same thing. The gap between "a working prototype exists in a university lab" and "someone else can build one at home" is where this work begins.

1.5 PROBLEM STATEMENT

These remaining challenges aren't independent. They group into three problem areas: *reproducibility* of the hardware build, *accessibility* of the software workflow, and the absence of a *community* around the project. Each reinforces the others. Without a build guide, no one rebuilds the system; without rebuilds, no community forms; without a community, no one writes accessible software. The core problem is therefore one of *democratisation*: closing this loop, rather than fixing any single barrier, is what separates a working prototype from something makers can actually build.

1.6 RESEARCH OBJECTIVES

The main goal of this research is to redesign the multi-axis retrofit system so that it can be built by independent makers using parts that are affordable and easy to source.

In addition to the hardware, a set of open-source tools and documentation is developed, including software tools, build guides, part lists, and calibration procedures. The result should be a complete system that guides a maker from building the hardware to producing multi-axis prints. Finally, the system is put in front of independent makers to reproduce, and their feedback is incorporated into the design.

1.7 RESEARCH QUESTIONS

The primary research question is:

How can a 5-axis retrofit system be redesigned and documented so that independent makers can reproduce it?

This is broken down into the following sub-questions:

1. What factors in open-source hardware development and technology adoption determine whether a project is successfully reproduced by independent makers?
2. What software tools, documentation, and community infrastructure are needed for successful replication by makers without specialist knowledge?
3. How can community feedback be gathered and analysed to inform and iteratively improve the system?

1.8 SCOPE AND BOUNDARIES

This research aims to make 5-axis printing more accessible for the maker community, not to develop new printing technology. No novel kinematic concept or new slicing algorithm is being invented. Rather, a proven concept is being made reproducible. The contribution lies not in the technology itself, but in its *democratisation*.

Target audience

The intended audience is makers with basic knowledge of 3D printing and electronics. Not experts, but not complete beginners either. The assumption is that the builder owns or has access to a working FDM printer, is comfortable with basic wiring and soldering, and can follow detailed but non-trivial instructions. This is roughly the demographic that builds Voron printers or assembles Prusa kits: technically engaged hobbyists who enjoy the build process.

Budget constraints

The design should be affordable. The target is to keep the total cost of the retrofit, excluding the base printer and standard tools, under €200. This is in line with Andersons' original BOM cost of roughly €150 and keeps the system accessible to hobbyists.

Technical boundaries

This research doesn't aim to create entirely new slicing software or build firmware from scratch. Instead, it builds on existing tools and workflows, especially the Marlin firmware ecosystem. The focus is on the FDM process; other AM technologies such as SLA and SLS are outside scope. Industrial applications, certification, and commercialisation are also not addressed. The goal is a functional, well-documented, open-source system for personal and educational use.

The Creality Ender 3 V3 SE and Ender 5 Pro serve as the primary development platforms because they are widespread and affordable. But the design is explicitly intended to be adaptable to other printers, and community guidelines are provided for this purpose.

1.9 THESIS OUTLINE

The remainder of this thesis is structured as follows.

The literature review reviews the literature across three areas: the **technical landscape** of multi-axis FDM (building briefly on Andersons' research), **open-source hardware** development practices and what makes projects reproducible, and **technology adoption** theory using Rogers' diffusion of innovations framework. The emphasis is on adoption and accessibility rather than the technical foundations, which Andersons already covered in depth.

The methodology chapter describes the **research methodology**, including the design approach, the user validation strategy, and the tools employed throughout this work.

The analysis chapter presents the **analysis of requirements**, both functional and non-functional, that the redesigned system must satisfy.

The design and development chapter details the **design and development** of the Rep5x system: the hardware and the browser-based software tools.

The evaluation chapter presents the **evaluation** of the system through independent builder tests and community feedback.

The discussion chapter provides a **discussion** of the results, reflecting on what worked, what didn't, and what the findings mean for the field.

Finally, the conclusion draws **conclusions** and identifies directions for **future work**.

CHAPTER 02

02

LITERATURE REVIEW

Three areas frame this work: the technical landscape of multi-axis FDM, the open-source hardware development model, and technology adoption theory. The first is brief, since Andersons covered it in depth; the emphasis is on why working technology doesn't get adopted, and what it takes to change that.

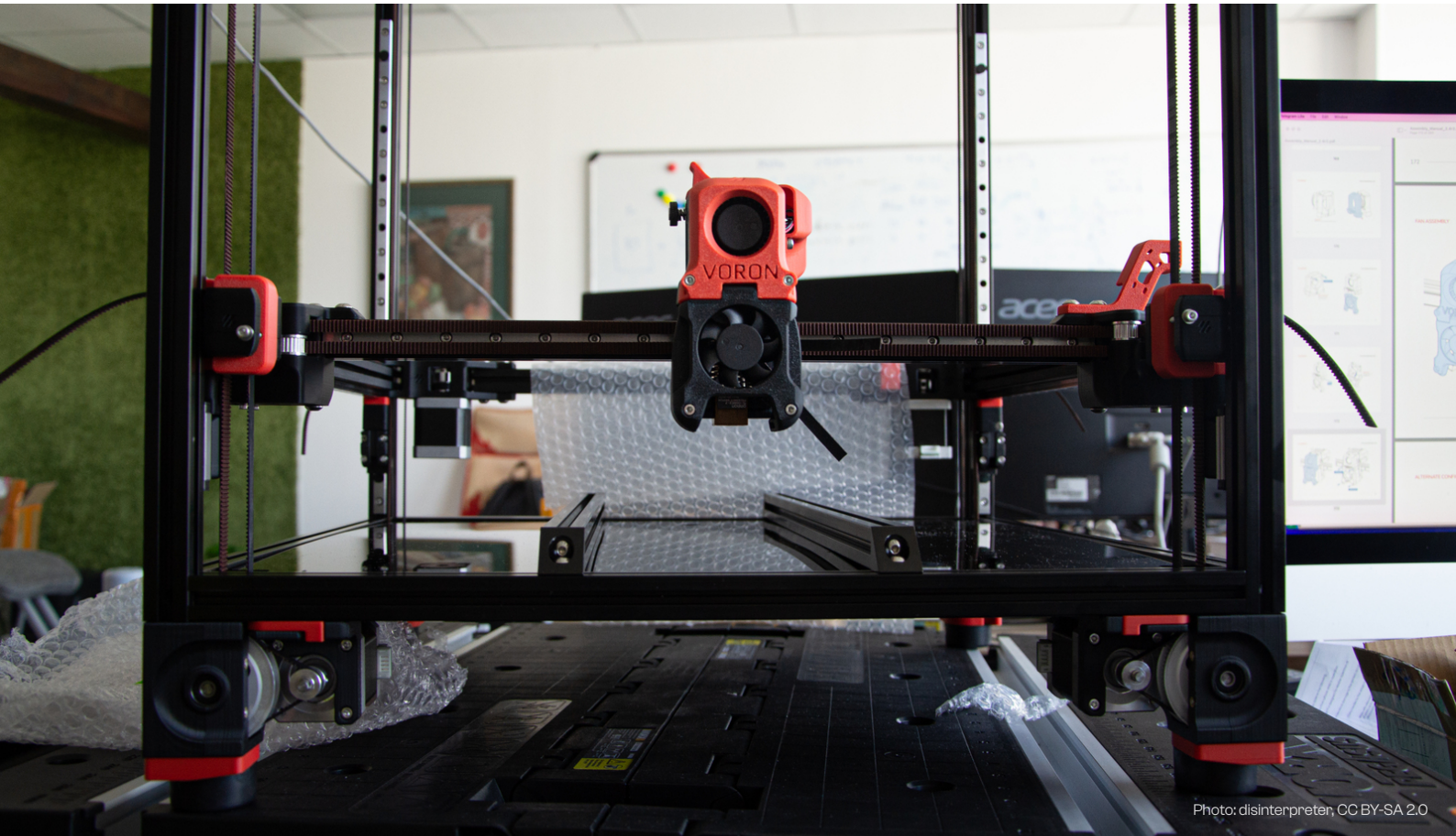


Photo: disinterpreter, CC BY-SA 2.0

The **technical landscape** comes first and shortest: Andersons [18] already documented it, so the focus here is what's changed since 2023. The **open-source hardware** section asks what makes a hardware project reproducible, drawing on the RepRap legacy and the wider maker community. The **technology adoption** section then turns to Rogers' diffusion of innovations as its analytical lens.

2.1 MULTI-AXIS ADDITIVE MANUFACTURING

Kinematic configurations

Multi-axis FDM systems extend the three linear axes of a conventional printer with two additional rotational axes, providing the 5 Degrees of Freedom (DoF) needed for non-planar deposition. Only 5 DoF are required because the nozzle output is rotationally symmetric, making a 6th axis redundant [18].[1] The two additional axes can be placed at the print head, the build plate, or split between them, producing three common categories: head-head (HH), table-table (TT), and head-table (HT) [18].

Industrial systems typically use 6-DoF robotic arms [14], which are versatile but expensive. As Andersons [18] puts it, they're "too expensive to be adopted in the consumer or prosumer AM market." For consumer-scale multi-axis printing, a more practical approach is the 3+2 configuration: a conventional 3-axis printer combined with a separate 2-axis mechanism [14], the same idea as adding a rotary table to a 3-axis CNC mill.

Andersons [18] provides a detailed comparison of TT, HH, and HT configurations. The key trade-offs: TT configurations keep the nozzle gravity-aligned but add mass to the bed and reduce build volume. HH configurations preserve build volume but introduce filament routing constraints and add mass to the gantry. HT splits the axes as a compromise. Each has been demonstrated in research prototypes, including the Open5x TT retrofit for the Prusa i3 MK3s [22] (Fig. 2.1), a variant by a community maker for Voron-style printers, and the Pentaxis HH retrofit [18].

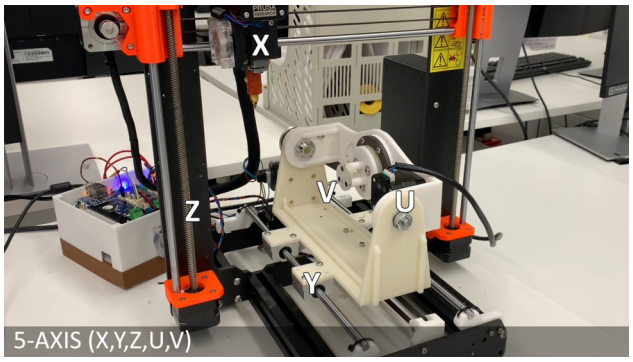


FIG. 2.1 The Open5x table-table 5-axis retrofit mounted on a Prusa i3 MK3s. Source: Hong et al. [22].

Multi-axis printing also introduces an interference problem that doesn't exist in conventional FDM. When the build plate or nozzle tilts, the part being printed can collide with the hot-end assembly. In planar printing, the flat-layer approach guarantees that no part geometry is ever higher than the nozzle tip. That guarantee disappears with non-planar motion [18]. Several solutions have been proposed, from extended nozzles with heat-insulating sleeves to slim-profile hot-ends that replace the conventional bulky heat block entirely, all trading thermal performance for clearance [18].

Path-planning and slicing

Conventional slicers cut a part into flat cross-sections along the Z-axis. To use the additional degrees of freedom in multi-axis FDM, parts need to be sliced differently, and the toolpath generation algorithms are considerably more complex [14]. Andersons [18] identifies three main approaches.

Feature-based decomposition treats different regions of a part as separate structures, each printed with its own build direction. An overhanging feature is printed in a direction where the overhang is no longer problematic, using a previously printed surface as its effective build plate. The main challenge is automatically decomposing complex geometry into printable sub-volumes [14].

Conformal slicing generates curved layers that follow the geometry of the part, using iso-surfaces of a scalar field rather than planar cross-sections. This is effective for reducing the staircase effect on curved surfaces. Dai et al. [23] demonstrated this for support-free volume printing using a 6-DoF robotic arm. Andersons [18] used a conformal approach implemented in Grasshopper, a visual scripting plugin for the commercial CAD package Rhino. The RotBot's conical slicing is a specific variant that uses only 4 axes, trading generality for simplicity [18].

G-code transformation avoids multi-axis slicing entirely. The part is sliced conventionally using a standard slicer, and the resulting G-code is mathematically transformed to include rotational axis commands. This uses the already mature toolpath generation of existing slicers (retraction, infill patterns, speed control) rather than reimplementing it from scratch. The downside is that the transformed toolpaths aren't aware of multi-axis constraints during generation, which can lead to collisions or suboptimal deposition angles.

Calibration and error compensation

Adding rotational axes introduces geometric errors that don't exist in conventional planar systems. The important parameters are the offsets between the rotational axes and the nozzle tip. In an HH configuration, these are the distances from the nozzle tip to the yaw rotation centre, and from the nozzle tip to the tilt rotation centre. Small errors in these offset values show through the inverse kinematics calculations, producing dimensional errors that increase with tilt angle. Additional error sources include axis misalignment, backlash in rotary mechanisms, and eccentricity in rotation [18].

Andersons [18] measured these errors by actuating the rotational axis in increments and recording the deviation at each position, using a dial gauge for Z-axis errors

[1] This wouldn't hold for non-circular nozzle geometries, which have been explored in some research.

and a reference cone for XY errors. The process is time-consuming and requires some skill, but doesn't need specialised equipment. Industrial 5-axis CNC uses more sophisticated methods (touch probes, laser interferometry, ball-bar testing), but these aren't practical for desktop printers. A middle ground is needed: good enough to produce quality prints, simple enough for a non-expert to perform.

Even after physical calibration, residual errors remain that vary periodically across the rotation range. Andersons [18] fitted sinusoidal error functions to measured data points, transforming a mechanical calibration problem into a software compensation problem. This is important for accessibility: the user doesn't need perfect mechanical alignment, just enough measurements for the software to correct the remaining errors.

Firmware and inverse kinematics

When rotational axes are added to an FDM printer, the G-code specifies the desired nozzle position and orientation, but the firmware must calculate where the physical motors need to move. Rotating the build plate around a pivot point displaces the workpiece relative to the nozzle, and the firmware (or a pre-processing step) must compensate for this displacement [18].

Neither Marlin nor Klipper, the two big open-source FDM firmware platforms, has native support for 5-axis FDM kinematics. Andersons [18] handled inverse kinematics as a pre-processing step in a Python script. This works, but it tightly couples the software pipeline to the printer's kinematic parameters. Changes to the physical machine require regenerating the entire toolpath, not just updating a firmware parameter. An alternative is to implement inverse kinematics directly in the firmware, so the printer accepts standard 5-axis G-code and internally computes motor positions. This makes the system more modular and enables real-time calibration corrections.

Recent developments

Several developments in multi-axis AM have appeared since Andersons' 2023 thesis. Two reviews of the field were published in 2024: Tang et al. [14] and Yao et al. [24] both catalogue the limitations of 3-axis AM, the potential of multi-axis approaches, and current implementation methods. Tang notes that "current multi-axis AM systems use robot arms extensively for manufacturing," with tool-chain standardisation between robots and CAD systems still an open problem [14]. Yao identifies recurring issues around strength, anisotropy, and the fabrication of large-scale structures [24].

On the hardware side, parallel academic work has begun targeting accessibility directly. Cocco et al. [25], in a 2025 paper titled *"Towards Accessible Non-Planar FFF"*, demonstrate non-planar FFF on a commercially available RatRig V-Core 3.1 modified with three independently actuated Z-axes, allowing the bed to tilt up to 30° during printing (shown in Fig. 2.2). The authors describe the modification as requiring "only simple extensions to the rails and bed screws," and release an open-source Python implementation that maps 3D toolpaths to machine commands. The motivation parallels this thesis: bring non-planar FFF within reach of the maker community without a robotic arm or custom-built hardware.

On the commercial side, Fractal Robotics released the Fractal 5 Pro [26], an open-source (GPLv3) benchtop 5-axis FDM printer with a CoreXY motion system inspired by the Voron Trident, accompanied by a custom 5-axis slicer (Fractal Cortex). The project's bill of materials totals roughly USD 1,900 for a self-build.

What remains absent is a complete, documented, open-source system that combines accessible hardware, user-facing software tools, and community infrastructure. Individual pieces exist, but none offer the full stack needed for a non-expert to go from owning a desktop printer to producing multi-axis prints. Fig. 2.3 shows the existing multi-axis FDM systems referenced throughout this thesis, and Appendix B provides a detailed comparison of each. Appendix C looks at the available slicing and toolpath tools. The pattern across both is consistent: no existing project supports multiple printer platforms, software accessibility is poor (requiring either commercial Rhino/Grasshopper licences or Python scripting knowledge),

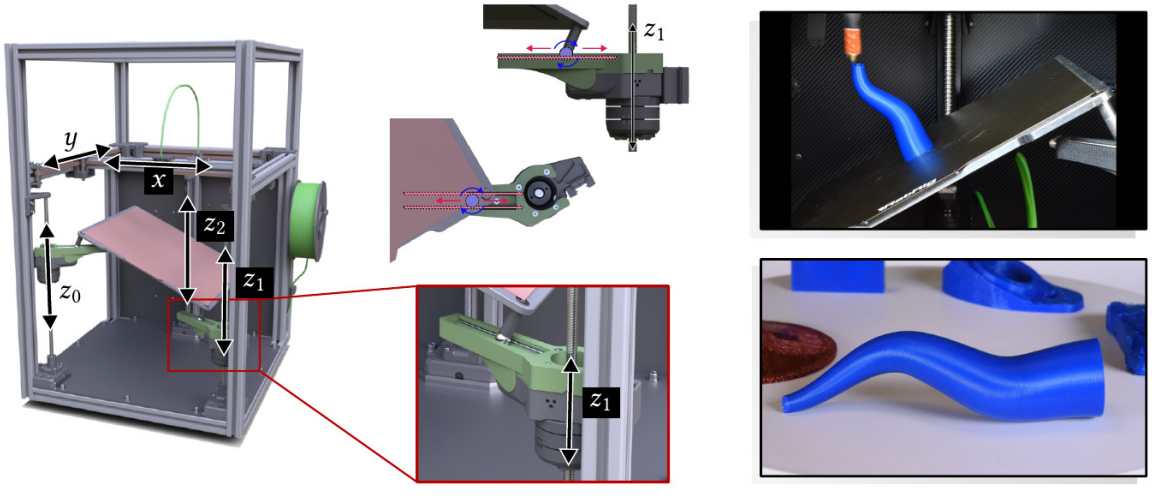


FIG. 2.2 Cocco et al.'s triple-Z modification of the RatRig V-Core 3.1, tilting the bed up to 30° for non-planar deposition [25].

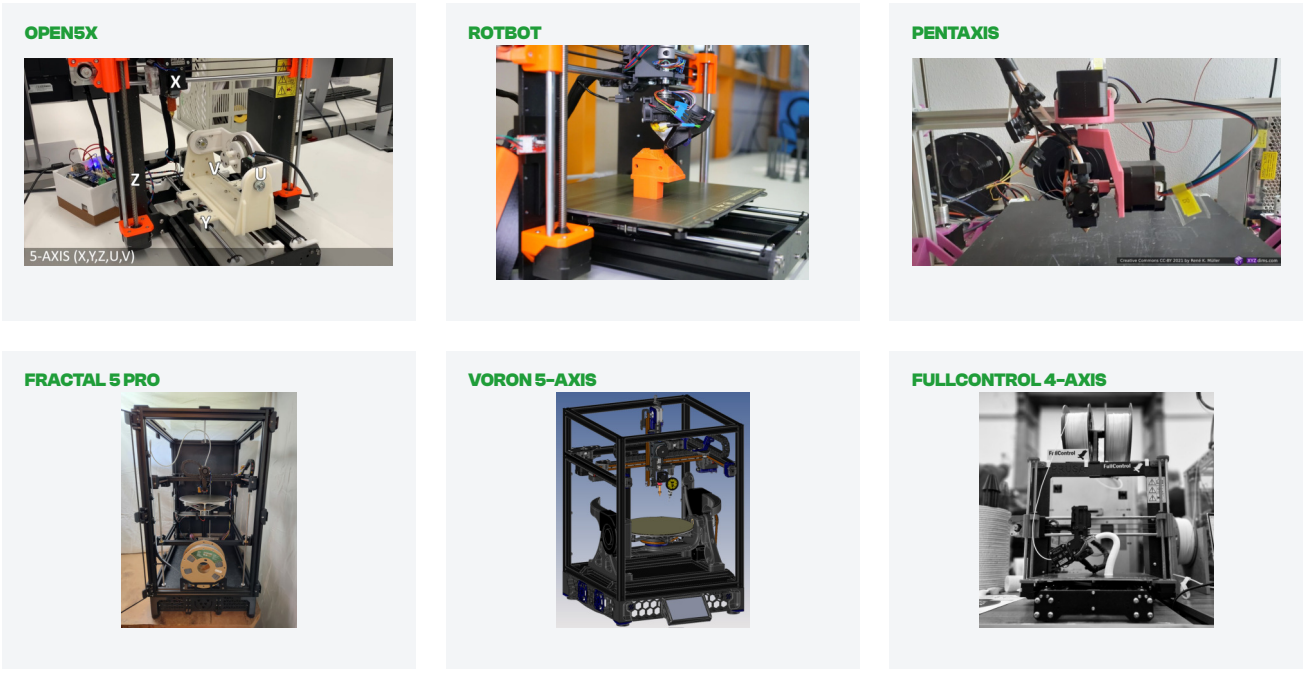


FIG. 2.3 Existing multi-axis FDM systems referenced throughout this thesis, left to right: Open5x [22], RotBot [27], Pentaxis [28], Fractal 5 Pro [26], Voron 5-axis [29], and FullControl 4-axis [30]. Appendix B gives a full comparison.

documentation rarely extends beyond a research paper, and none has built a community for ongoing builder support. The gap isn't any single barrier; it's that no project addresses all of them together.

2.2 OPEN-SOURCE HARDWARE AND THE MAKER ECOSYSTEM

Technical capability alone doesn't drive adoption. The history of consumer 3D printing shows that what matters equally is the development model: how hardware is designed, shared, documented, and reproduced. This section examines the open-source hardware movement and its relevance to making multi-axis printing accessible.

The RepRap project and its legacy

The RepRap project, initiated by Adrian Bowyer at the University of Bath, is what launched consumer 3D printing. The project's goal was to create a 3D printer that could manufacture a large fraction of its own parts, enabling exponential growth through self-replication [10]. The RepRap machine achieved self-replication of 48% of its parts by count, ignoring fasteners such as nuts, bolts, and washers [6]. The remaining components were standard commercial off-the-shelf (COTS) items available from hardware suppliers.

The timing was important. Stratasys held a patent on Fused Deposition Modelling (US5121329, filed 1989), and the RepRap community coined the term "Fused Filament Fabrication" (FFF) specifically to provide a synonym that could be "used by all unrestrictedly" [6]. Several companies formed to sell machines inspired by RepRap, including MakerBot Industries in the United States and Bits From Bytes in the United Kingdom [6]. Their RepRap-derived designs were distributed as free and open-source, as required by the GPL.

What made RepRap successful wasn't just the hardware. It was the development model. The project was entirely open-source, with designs shared freely under the GPL licence. Contributors could modify designs and share improvements using the same printer they were improving. Jones et al. [6] describe how the first child RepRap, built from parts made by its parent, immediately printed a timing-belt tensioner as its very first part, a "grandchild" component for the next generation. This capacity for a machine to produce parts for its own copies is unique to self-replicating open-source hardware. Fig. 2.4 shows a typical Prusa Mendel RepRap with a self-printed filament spool, illustrating the principle.

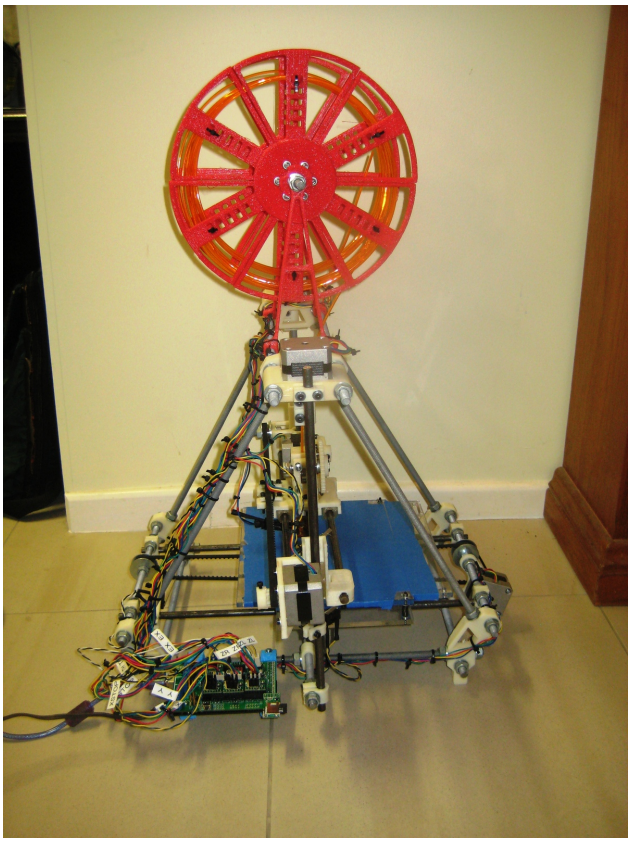


FIG. 2.4 A Prusa Mendel RepRap with a self-printed filament spool [31].

Pearce [32] argued that combining 3D printing with open-source microcontrollers (Arduino) running free software enables the creation of scientific equipment at a fraction of proprietary costs. Wittbrodt et al. [33] quantified this: households with open-source 3D printers achieve cost savings of $300-$2000 per year, with a simple payback period of 4 months to 2 years. The economic case for distributed manufacturing with open-source hardware is clear.

Open-source hardware principles

Open-source hardware (OSHW) borrows its philosophical framework from open-source software, but the translation isn't straightforward. Software can be copied at zero marginal cost. Hardware can't. Every reproduction of a physical design requires materials, tools, and labour. This fundamental difference means that "open" in the hardware context requires more than just publishing source files.

Licensing and intellectual property. The Open Source Hardware Association (OSHWA) defines open-source hardware as "hardware whose design is made publicly available so that anyone can study, modify, distribute, make, and sell the design or hardware based on that design." Several licences have been developed specifically for hardware: the CERN Open Hardware Licence (OHL), Creative Commons for documentation, and the GPL for firmware and software components. The choice of licence affects what downstream users can do with the design, particularly whether modifications must also be shared (copyleft) or not (permissive).

In practice, many projects that call themselves open-source don't meet even basic criteria for openness. Bonvoisin et al. [34] analysed the public documentation of 132 OSHW products and found that practitioners interpret "open source" inconsistently. Many projects share only a fraction of the information needed for reproduction. Whether a project is truly open depends not just on licensing but on the completeness of shared design documentation: CAD files, BOMs, assembly instructions, and firmware source code. Only 17% (22/132) of the products in their sample satisfied all four composite criteria of openness (transparency, accessibility, replicability, and commercial usability), and just 11% achieved the maximum openness score on the eight-criterion scale.

The challenge isn't unique to 3D printing. Oellermann et al. [35] review how open electronics in scientific research can improve reproducibility and democratise access to instrumentation, though they note practical barriers including limited documentation, poor awareness, and community fragmentation. Oberloier and Pearce [36] propose a systematic design procedure for free and open-source hardware for scientific equipment, reporting cost reductions of 90-99% compared to proprietary equivalents and emphasising that documentation must be maintained alongside the design from the start, not bolted on as an afterthought.

Self-replication and distributed development. The concept of self-replication in 3D printing extends beyond the RepRap project's original vision. When a printer can manufacture parts for copies of itself, it enables a fundamentally different development model. New hardware improvements can be pushed to existing users: a user with an older printer can print the parts for an upgrade, or even the parts for an entirely new machine. This development model has no real precedent outside 3D printing.

For multi-axis printing, self-replication is particularly relevant. The additional mechanical components needed to convert a 3-axis printer to a 5-axis system can themselves be 3D printed. This means an existing printer can bootstrap the hardware needed for its own upgrade. The Open5x project demonstrated this principle: most parts of the rotary table assembly are 3D-printable [22]. But self-replication only works if the printed parts are designed to be printable on the source machine. Large parts that approach the build volume limits, or parts requiring support structures that are difficult to remove, undermine the self-replication potential.

Design for reproduction

Publishing CAD files isn't enough to make a hardware project reproducible. The design itself must be engineered for reproduction by people with varying skill levels and access to different tools and materials.

Design for manufacturability in open-source projects. Bonvoisin et al. [21] developed the Open-o-Meter (Fig. 2.5), a tool for measuring how "open" a hardware project actually is. Their eight criteria include: an OSH-compatible licence, design files, a BOM, assembly instructions, original editable files, a version control system, a

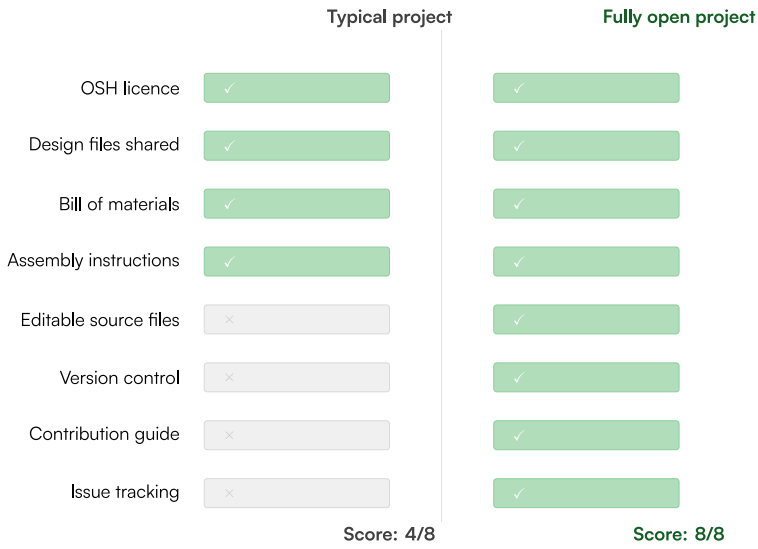


FIG. 2.5 The Open-o-Meter criteria for measuring hardware openness [21]. Most OSHW projects satisfy only a subset of these criteria.

contribution guide, and issue tracking. Most projects fail on several of these criteria. The Open-o-Meter makes explicit what many OSHW practitioners implicitly understand: openness is a spectrum, and most projects fall short of full openness.

Sells et al. [37] demonstrate that the combination of self-replication and open design maximises customisability, since users can modify and fabricate improvements using the same machine. This principle extends naturally to retrofit systems: an existing printer can produce the parts needed for its own multi-axis upgrade.

For 3D-printed hardware specifically, design for manufacturability means designing parts that print reliably on common consumer printers. This includes: avoiding excessive overhangs that require support material, keeping part dimensions within typical build volumes (roughly 180×180×180 mm), using common materials (PLA, PETG), and minimising the need for post-processing.

Bills of materials and sourcing strategies. A complete and accurate BOM is essential for reproduction. But a BOM is only useful if the listed components are actually available to the builder. Projects that specify custom-machined parts, obscure electronic components, or region-specific suppliers create barriers to reproduction. The most reproducible OSHW projects use globally available COTS components: standard metric fasteners, common stepper motors (NEMA 17), widely available controller boards (Arduino, STM32-based), and standard bearings. When a project specifies a component that's only available from a single supplier, it creates a fragile dependency.

Success factors and failure modes

Antoniou et al. [20] surveyed 30 OSHW practitioners to identify what makes projects succeed. Three themes emerged: value creation (the big-picture impact), quality of output (the quality of hardware and accompanying documentation), and effective processes (the activities that contribute to success). Successful projects create hardware that is "performant and highly accessible, reproducible and modifiable." Documentation quality was referred to by 25 of 30 respondents as a factor in project success.

The most successful OSHW projects, like Arduino and the Prusa i3, share certain characteristics: active community engagement, documentation maintained alongside the hardware, economic sustainability (Arduino through board sales, Prusa through commercial sales of kits and assembled printers), and a design that's genuinely reproducible without the original developer's involvement. The failure mode is the opposite: a project published as "open source" with minimal documentation, no community, and designs that only the original developer can reproduce. Bonvoisin et al. [34] found that most self-described OSHW projects occupy a middle ground of partial openness, with a mean openness index of 4.2 across eight binary criteria. Only 17% satisfied all four composite criteria (transparency, accessibility, replicability, commercial usability).

2.3 TECHNOLOGY ADOPTION, ACCESSIBILITY, AND COMMUNITY

The first two sections of this chapter address the *what*: the technical landscape of multi-axis AM and the open-source development model. This section addresses the *how*: the mechanisms by which a niche technology reaches a broader audience.

Diffusion of innovations

The canonical framework for technology adoption is Rogers [19], who identifies five attributes of an innovation that determine its rate of adoption: relative advantage (is it better than what it replaces?), compatibility (does it fit existing practices?), complexity (how hard is it to understand and use?), trialability (can potential adopters experiment with it?), and observability (can the results be seen by others?). These five attributes account for 49—87% of adoption variation across studies.

Applying this to multi-axis FDM is instructive. The *relative advantage* is well established: reduced support material, improved surface quality, controlled anisotropy.

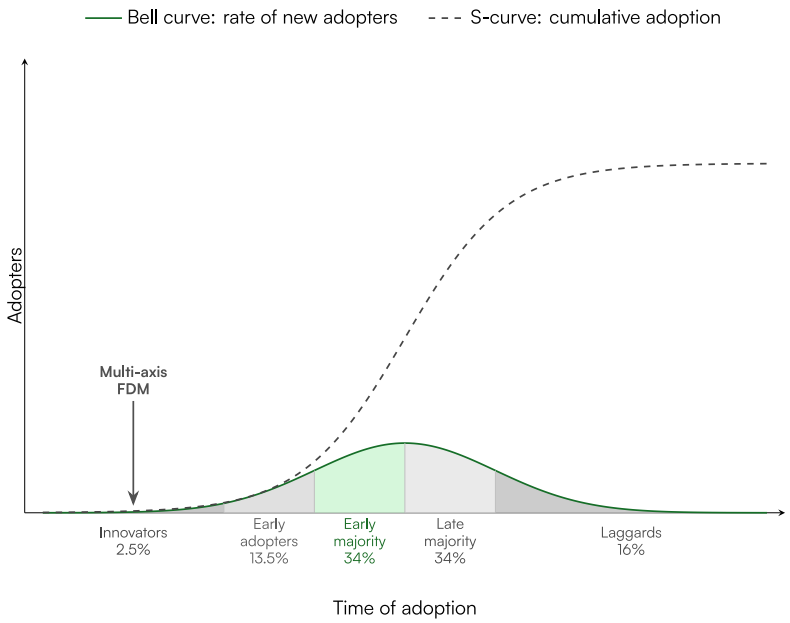


FIG. 2.6 Rogers' adopter categories and the diffusion S-curve [19].

But the remaining four attributes are all problematic. Multi-axis printing isn't *compatible* with existing workflows; it requires different slicers, different firmware, and modified hardware. It scores poorly on *complexity*; the learning curve is steep. *Trialability* is near zero because there's no easy way to experiment with multi-axis printing without first building or buying specialised hardware. And *observability* is limited because so few people have working systems that the results aren't visible to the broader community.

Rogers also describes the adoption curve (Fig. 2.6): innovators (2.5%), early adopters (13.5%), early majority (34%), late majority (34%), and laggards (16%). Multi-axis FDM is currently at the innovator stage, confined to researchers and a small number of dedicated makers. Crossing the chasm to the early adopter stage requires lowering the barriers across all five of Rogers' attributes.

The gap isn’t in what multi-axis printing can do. It’s in who can do it.

DIFFUSION OF INNOVATIONS, APPLIED TO MULTI-AXIS FDM

THE ADOPTION GAP

Rogers' five attributes explain why a working technology stalls. For multi-axis FDM the relative advantage is strong, but compatibility, complexity, trialability and observability all remain unresolved [19]. And only 17% of open-source hardware projects meet full openness criteria [34], so even published designs are rarely reproduced.

Barriers to adoption in niche manufacturing

Hardware cost and complexity. The most obvious barrier is the hardware itself. Industrial multi-axis AM systems (robotic arms, custom 5-axis platforms) cost tens of thousands of euros. Commercial off-the-shelf 5-axis FDM machines do exist. VSHAPER's 5AX is marketed to industrial customers in the automotive, aerospace, and medical sectors [38], and the 5AXISMAKER hybrid launched as the "first affordable 5-axis multi-fabricator" at around $5,000 [39]. But both sit well above the €200—500 range typical for consumer FDM. Even DIY approaches like Open5x require a compatible base printer, electronic components, and a substantial time investment for assembly. The additional mechanical complexity of multi-axis systems is itself a barrier, as the challenges catalogued by Tang et al. [14] illustrate. Each additional axis introduces new sources of error, requires additional motors and other components, and makes the system harder to assemble, calibrate, and maintain.

The cost barrier isn't just financial. It's also a barrier of effort. A maker who wants to try multi-axis printing must first get or build the hardware, learn how to calibrate it, find or create compatible slicing software, and troubleshoot the inevitable problems that arise when combining modified hardware with firmware that wasn't designed for it. The total effort required is orders of magnitude greater than setting up a conventional 3-axis printer, which typically works out of the box.

Software ecosystems and tool accessibility. Even when the hardware exists, the software ecosystem for multi-axis FDM is fragmented and inaccessible. The two closest peers that ship both hardware and software, Andersons' retrofit [18] and Open5x [22], both rely on Grasshopper (a visual programming plugin for Rhino, a commercial CAD package) for conformal slicing and G-code generation. This stack works, but it requires: a Rhino licence, familiarity with visual programming, knowledge of the specific Grasshopper components, and an understanding of the underlying mathematics. The Open5x authors themselves acknowledge the constraint: "this dependency on a commercial product means that our solution is not completely open source" [22].

The contrast with conventional 3D printing is big. A user buying a new FDM printer today can download a free slicer, import an STL file, click "slice," and start printing within minutes. No comparable experience exists for multi-axis printing. The software tools that do exist are scattered across research papers, GitHub repositories, and individual makers' personal projects, with no unified workflow.

Knowledge gaps and documentation. The knowledge required to build and operate a multi-axis printer spans mechanical design, electronics, firmware configuration, slicing algorithms, inverse kinematics, and calibration. No single resource covers all of these in the context of consumer-grade multi-axis FDM. Research papers present results but rarely provide step-by-step build instructions. Maker projects on forums and YouTube show results but often lack the technical depth needed to replicate the work from scratch. Builders are left to piece together understanding from disparate sources, which scales poorly: each new builder repeats the discovery process the previous one already went through.

Documentation as a design problem

Build guides and instructional design. A build guide isn't just a list of steps. It needs to anticipate where builders will make mistakes, provide visual references for ambiguous assembly steps, and explain the *why* behind design decisions so that builders can troubleshoot when things go wrong. The most successful OSHW projects (Prusa, Voron) treat documentation as a first-class deliverable alongside the hardware and software. A mediocre design with great documentation gets reproduced more successfully than a brilliant design with poor documentation.

Web-based tools and no-install software. A real barrier to tool adoption is the installation and configuration overhead. Desktop applications require downloading, installing, resolving dependencies, and often purchasing licences. Browser-based tools eliminate this entirely: the user opens a URL and the tool is ready to use. The Web Serial API, available as an origin trial from Chrome 78 (2019) and shipped as stable in Chrome 89 (March 2021), extends this to hardware communication, letting a web page read from and write to serial devices such as 3D printers and microcontroller boards.

This shift is particularly relevant for niche hardware like multi-axis printers, where the target audience is already juggling hardware assembly, new concepts, and mechanical troubleshooting. A tool that runs in any modern browser, with no installation and a guided interface, has a fundamentally lower barrier to entry than any desktop application.

Community-driven development

Community as infrastructure. Von Hippel [40] established that users of products increasingly innovate for themselves, and that user innovation communities enable distributed, collaborative development. In the 3D printing space, this is obvious: the entire consumer FDM ecosystem was built by user-innovators, from the RepRap project to the Klipper firmware to the Voron printer. Communities on Discord, Reddit, GitHub, and dedicated forums serve as infrastructure: they provide channels for support, feedback, and contribution that no individual developer can replicate alone.

For a niche technology like multi-axis printing, community isn't optional. It's a prerequisite for adoption. Without a community, every new builder has to solve every problem independently. With a community, solutions are shared, common failure modes are documented, and the accumulated knowledge of all builders is available to each new one. The community also provides social proof: seeing that others have successfully built and used the system lowers the perceived risk for potential adopters, directly addressing Rogers' *observability* attribute.

Feedback loops and iterative improvement. Open-source hardware development is inherently iterative. Users

build the system, encounter problems, and report them. The developer fixes the design, and the next builder benefits from the improvement. This feedback loop is the engine of OSHW development, but it only works if there are enough active users to generate feedback, and if there are mechanisms (issue trackers, forums, direct communication) for that feedback to reach the developer.

Antoniou et al. [20] identified effective processes as one of the three themes of OSHW success, covering the activities and practices that contribute to a project's development. Version control, structured issue tracking, and clear contribution guidelines all help. But the most important factor is simply having active users. A project with five active builders who report problems and suggest improvements will develop faster than a project with fifty passive observers.

Peer-to-peer knowledge transfer and makerspaces. Knowledge transfer in maker communities happens through multiple channels: documentation (build guides, wikis), visual media (YouTube videos, photos of builds), real-time communication (Discord, forums), and in-person interaction (makerspaces, conferences). Each channel serves a different purpose. Documentation provides the reference. Visual media provides the context. Real-time communication provides the troubleshooting support. And in-person interaction provides the tacit knowledge that's difficult to transfer through any other medium.

Makerspaces can be particularly relevant for complex hardware projects. Seeing a multi-axis printer in person, watching it operate, and talking to the person who built it conveys information that's hard to match online. But makerspaces are geographically limited. Online communities, while lacking the tacit knowledge transfer of in-person interaction, provide global reach. A Discord server with 100 members across different countries provides a support network that no single makerspace can.

Case studies in accessible 3D printing

The history of consumer 3D printing provides several examples of how niche technology became accessible. Five projects are examined in detail in Appendix A, each analysed against Rogers' adoption framework. The key findings are summarised here.

RepRap proved that self-replicating, open-source hardware can catalyse an entire industry [10]. But RepRap itself never crossed into mainstream adoption. Its complexity, fragmented documentation, and inability to "try before you build" kept it confined to the innovator demographic. The commercial derivatives that followed addressed those barriers.

Marlin firmware became the dominant FDM firmware not through individual adoption, but through manufacturers embedding it in commercial products [41]. Its GPLv3 licence ensures improvements flow back to the community. But Marlin's configuration model (editing C++ header files, recompiling, and flashing) is a real accessibility barrier for users who need to make custom modifications, which is directly relevant to multi-axis printing.

Prusa Research is the clearest example of taking an open-source design from innovator to mainstream [42]. The key wasn't the hardware but the *packaging*: documentation with assembly photos and troubleshooting, a reliable supply chain, and professional support. The company's recent shift from GPL to the more restrictive Open Community Licence (OCL) [43] illustrates the tension between openness and commercial sustainability when competitors can clone open-source designs [44].

Voron Design demonstrates that a purely community-driven project, with no commercial entity, can still reach broad adoption if the community infrastructure is strong enough [45]. Over 15,000 printers have been serialised as completed builds [46]. The Discord community effectively replaces the professional support that Prusa provides commercially, but this only works for an audience willing to engage with the community. Fig. 2.7 shows a Voron 2.4 mid-build.

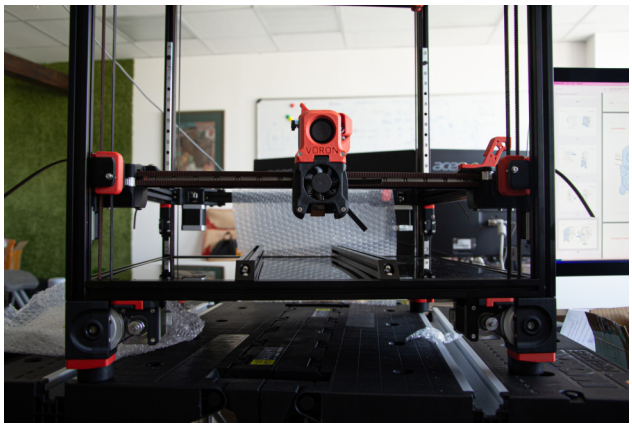


FIG. 2.7 A Voron 2.4 mid-build [47].

Klipper shows that architectural choices can be adoption choices [48]. By moving configuration out of the compiled binary and into a text file, and by pushing computation onto a host computer, the project removed two of the most significant barriers Marlin still imposes: recompilation and firmware-level debugging. The result is a thriving ecosystem of web-based front-ends like Mainsail and Fluidd, plus community-contributed macros, validating the approach of building browser-based tools rather than desktop applications. Klipper demonstrates that accessibility isn't just about documentation or community; it's also about the technical decisions made early in a project's development.

Fig. 2.8 compares the five projects against Rogers' five adoption attributes. The pattern is clear: relative advantage is universally strong, but compatibility, complexity, trialability, and observability vary widely between projects, and the projects that achieved broad adoption (Prusa, Klipper) score consistently higher across the latter four.

2.4 DISCUSSION

Multi-axis FDM works. Andersons [18] and others have shown that, and the kinematic configurations, path-planning strategies, and calibration methods are all well understood. The problem isn't solely technical.

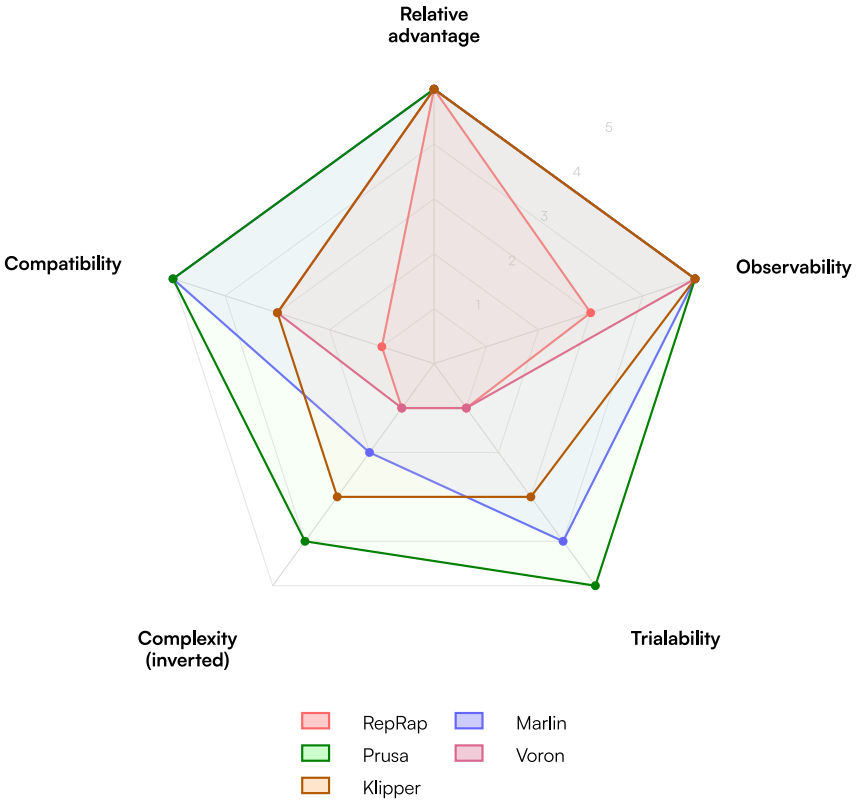


FIG. 2.8 Rogers' adoption attributes for the five case study projects.

The projects that took niche 3D printing into the mainstream didn't get there on technical aspects only. They got there through documentation, community, and tools that made the thing easy to try. Bonvoisin et al. [34] show why open licences by themselves don't cut it, and Rogers' framework shows what's still missing for multi-axis FDM: compatibility, trialability, complexity, and observability are all unresolved [19]. The gap isn't in what multi-axis printing can do, it's in who can do it.

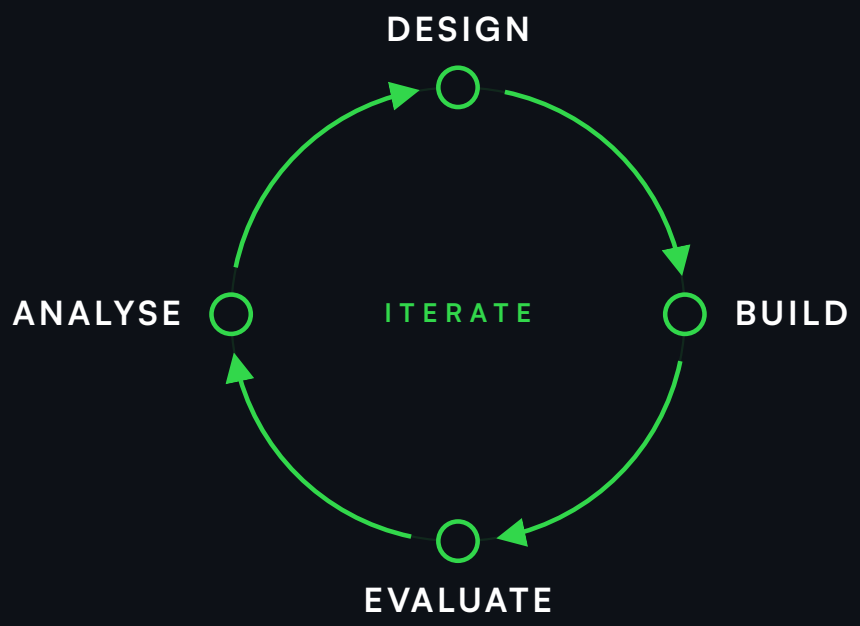
The next chapter takes these findings and translates them into concrete requirements.

CHAPTER 03

03

METHODOLOGY

How the system was designed, built, and evaluated: the Design Science Research approach, the user-validation strategy, and the methods behind each phase.



This thesis designs, builds, and evaluates a system. The research contribution is the system itself, the tools and documentation about it, and the evidence that independent makers can reproduce and use it. This is the Design Science Research (DSR) approach: the primary output is an artefact that addresses a real-world problem [49].

This chapter describes the research approach, explains why DSR is appropriate for this work, maps the methodology to the thesis structure, and details the specific methods used for requirements analysis, design, and evaluation.

3.1 RESEARCH PARADIGM

Design Science Research

Design Science Research is a problem-solving framework that produces knowledge through the design and evaluation of artefacts [49], [50]. It isn't about explaining why things happen, the way behavioural research is. It's about building things that solve problems. Hevner et al. [49] distinguish four types of artefact: constructs, models, methods, and instantiations. This thesis produces an *instantiation*: the Rep5x system, comprising hardware designs, software tools, documentation, and community infrastructure.

Peffers et al. [51] formalised the DSR process into six activities:

1. **Problem identification and motivation.** Define the problem and justify why a solution is worthwhile.
2. **Define the objectives for a solution.** Infer the requirements from the problem definition and existing knowledge.
3. **Design and development.** Create the artefact.
4. **Demonstration.** Use the artefact to solve the problem in a suitable context.
5. **Evaluation.** Observe and measure how well the artefact supports a solution.
6. **Communication.** Report the results to relevant audiences.

These six activities aren't strictly sequential. Peffers et al. [51] describe them as "nominally sequential," but note that researchers may start at almost any step, and that the evaluation activity can lead to iteration back to design and development. This matches the actual development process of Rep5x, which went through multiple hardware revisions and tool iterations based on builder feedback.

Why DSR fits this thesis

The research questions posed in the introduction are design questions, not empirical ones. They ask *how* to redesign a system, *what* tools are needed, and *how* community feedback can validate the result. These can't be answered through observation alone; they require building something and seeing if it works. That's what DSR is for.

The seven guidelines Hevner et al. [49] propose for rigorous DSR are all satisfied here: the Rep5x system is a concrete artefact, the problem is grounded in the adoption analysis from the introduction, the design is informed by adoption theory and OSHW best practices from the literature review, multiple iterations were explored during development, and the results are evaluated by independent builders in the evaluation chapter and communicated through this thesis, the project website, and the open-source repository.

Mapping DSRM to thesis structure

The six DSRM activities map directly to the thesis chapters, as shown in Fig. 3.1.

- **Problem identification:** the introduction: Andersons' prototype works but isn't adopted. The barriers are identified using Rogers' framework.
- **Objectives:** the requirements chapter: requirements are derived from the literature review and the adoption analysis.
- **Design and development:** the design and development chapter: the hardware, software tools, calibration system, documentation, and community infrastructure are designed and built.
- **Demonstration and evaluation:** the evaluation chapter: independent builders take on the build; their progress, the difficulties they hit, and their feedback are analysed.
- **Communication:** this thesis, the Rep5x website (rep5x.com), and the GitHub repository.

3.2 RESEARCH PROCESS

The research didn't follow a straight line. It started with a question, why isn't Andersons' working prototype being replicated?, which led to the literature review on adoption theory and OSHW. That produced a set of requirements. Those requirements drove the design. And as the system

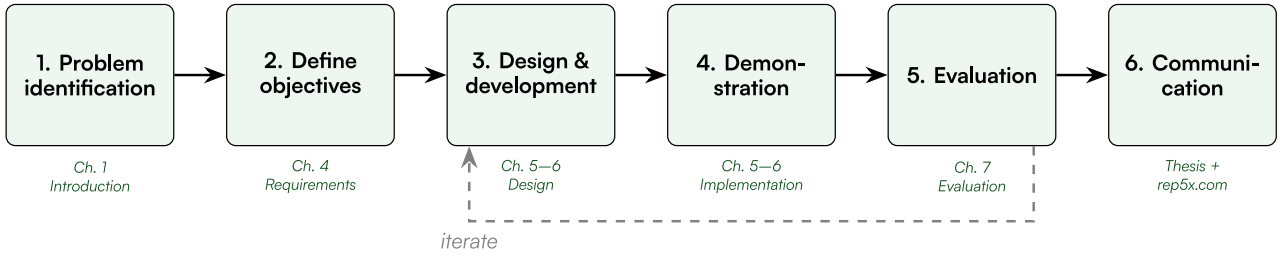


FIG. 3.1 The DSR Methodology process [51], mapped to the thesis structure.

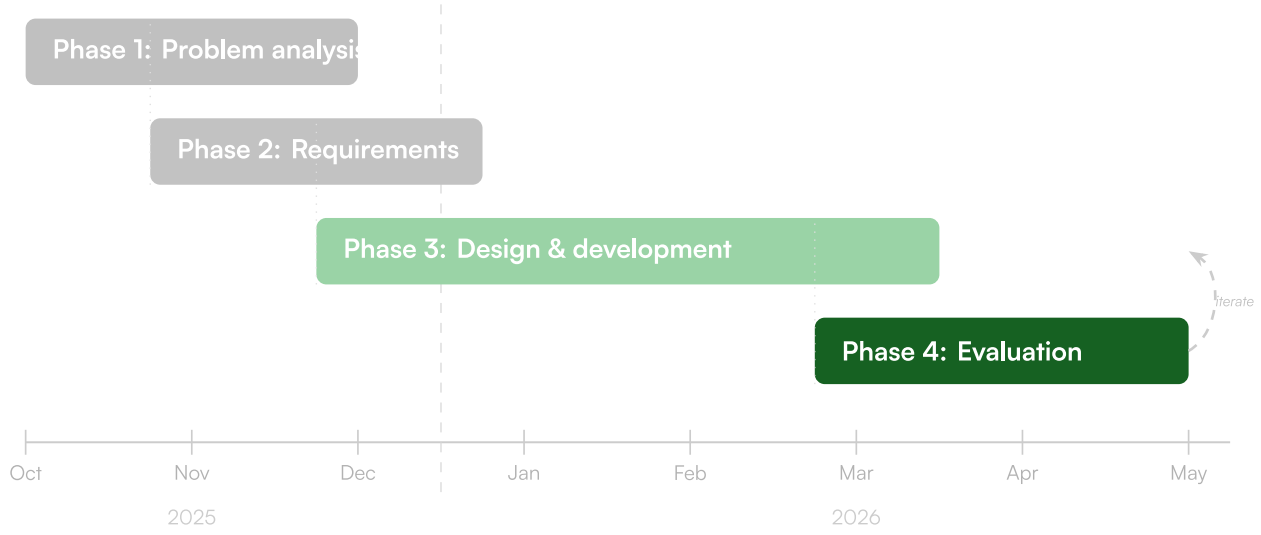


FIG. 3.2 Research phases and approximate timeline. Phases overlap due to the iterative nature of the DSR approach.

was built and tested, builder feedback prompted revisions, which fed back into the design. This cycle of design, test, revise is characteristic of DSR [51] and is summarised in Fig. 3.2.

Phase 1: Problem analysis

The starting point was Andersons [18], whose thesis produced a working 5-axis retrofit prototype. The question was: why hasn't this been replicated? To answer this, the existing system was analysed against Rogers' five adoption attributes [19] and the OSHW success criteria identified by Antoniou et al. [20] and Bonvoisin et al. [21]. This analysis identified specific barriers: hardware tied to a single printer, software requiring commercial tools (Rhino/Grasshopper), a complex setup process, fragmented documentation, and no community infrastructure.

Phase 2: Requirements definition

The barriers identified in Phase 1 were translated into design requirements using the framework developed in the literature review, covering hardware adaptability, software accessibility, documentation, community infrastructure, and open-source release. To check the barrier analysis against what makers actually perceive, two sources of user input were drawn on. First, a short online community survey was distributed during this phase through 3D printing Discord servers and subreddits (see the community survey below). Second, informal conversations on Discord, Reddit, and LinkedIn surfaced specific build concerns and tool wishlist items that the survey's closed-form questions didn't capture. No formal structured interviews were conducted; the survey included an opt-in field for follow-ups, but these were handled as ongoing informal exchanges rather than scheduled sessions. These inputs together fed into the requirements documented in the requirements chapter.

Phase 3: Design and development

The design and development phase produced the Rep5x system across four areas:

Hardware. The 5-axis retrofit was redesigned for reproducibility across multiple printer platforms (Ender 5 Pro, Ender 3 V3 SE, Ender 3 Pro).

Software tools. Eight browser-based tools were developed using vanilla JavaScript and the Web Serial API, eliminating the need for installation, licences, or command-line interaction. These tools cover the full workflow from firmware configuration to calibration to G-code generation.

Calibration system. A two-stage calibration process was designed: physical measurement of the geometric parameters (LC and LB), followed by Fourier series error mapping across the rotation range. The process is guided through the browser-based calibration tool, which walks the user through each step.

Documentation and community. A build guide was written with step-by-step instructions, wiring diagrams, and print settings. A website (rep5x.com) was built to serve as the project's public face. A Discord server was created as the primary community platform.

Each of these areas is detailed in the design and development chapter.

Phase 4: Evaluation

The evaluation phase tested whether independent makers could reproduce the system using only the provided documentation and tools. This is the critical test in DSR: does the artefact solve the problem it was designed to address? The evaluation methods are described below and the results are presented in the evaluation chapter.

Community survey

The barrier analysis in the introduction and the requirements chapter draws heavily on Rogers' adoption framework applied to Andersons' prototype. That's a theoretical position. To check it against what makers actually think, a short online survey was run early in the research.

Purpose. The survey had three goals: validate which adoption barriers the maker community perceives as most important, gather baseline data on their familiarity with

multi-axis printing and related skills, and find out what resources and tools they'd want from a project like this. The results informed the requirements in the requirements chapter.

Design. The survey had 16 questions grouped into five short sections: 3D printing background (experience, printer models, mod and firmware experience), multi-axis awareness and interest (including motivations), perceived barriers and price sensitivity, software and calibration preferences, and open-source and community attitudes. The question types were mostly multiple choice and checkbox, with two 5-point scales and one optional open-ended question at the end. An optional field was included for respondents willing to be contacted for a follow-up interview. The full question list is included in the survey appendix.

Distribution. The survey was distributed through Discord and Reddit, targeting active 3D printing communities. Google Forms was used as the platform, with no login required, to lower the participation barrier.

Sample. The target was roughly 50 responses; 89 were collected. This isn't a statistically representative sample of the broader maker community, and it wasn't meant to be. The sample size is enough to surface patterns in perceived barriers and resource preferences, which is what the survey was designed for.

Limitations. Three limitations are worth noting. First, self-selection: people who fill out a survey about multi-axis printing are already somewhat interested, so the results skew toward engaged makers rather than the general consumer population. Second, recruitment channels: Discord and niche 3D printing subreddits reach a technically engaged audience, not casual hobbyists. Third, the sample is small, so the results should be read as directional rather than definitive. None of this undermines the survey's purpose, which was to check the literature-based barrier analysis against real community input, not to produce generalisable statistics.

3.3 EVALUATION METHODS

Independent builder validation

The primary evaluation method is straightforward: can someone who wasn't involved in the development build the system using only the publicly available documentation, tools, and community support? This directly tests the thesis's central claim. If independent builders can reproduce the system, the documentation and tools are doing their job. If they can't, the gaps tell exactly what needs fixing.

Builders were recruited through online 3D printing communities, including Reddit, 3D printing forums, and the Rep5x Discord server. Each builder was provided with:

- Access to the build guide and all documentation
- Access to the browser-based tools
- Access to the Discord community for support
- A BOM with sourcing links

No direct assistance from the developer was provided beyond what any community member would receive through the Discord server.[2] This ensures the evaluation tests the documentation and tools, not the developer's personal knowledge.

Data collection

Data was collected through multiple channels:

Build logs. Builders were asked to document their build process, noting where they encountered difficulties, where the documentation was unclear, and how long each stage took.

Community interactions. Questions and problems posted in the Discord server were logged and categorised by topic (hardware assembly, electronics, firmware, calibration, software tools).

Conversation tracking. Beyond the Discord server, adoption-relevant exchanges across Reddit, Discord, and GitHub were logged in a self-hosted instance of Twenty, an open-source CRM [52]. The logging isn't message by message but one record per person, with a dated note per interaction. Each contact is classified by first-contact platform, build stage (research, planning, development, active, completed), build likelihood, engagement level, and skill level, and tagged by contribution type (building, testing, development, collaboration, advisory, interested). This turns otherwise scattered conversations into a structured record of who engaged with the project and how far they got, which the evaluation chapter reads as an adoption funnel.

Print results. Builders who completed their systems were asked to share photos and observations from their first prints. Given the complexity of the build process and the early stage of the project, the number of completed systems during the research period is small. The evaluation focuses on whether builders can get to a working print at all, rather than on detailed quality comparisons.

Builder feedback. Throughout the build process, builders provided feedback on documentation clarity, tool usability, and points of confusion. This feedback was collected informally through Discord conversations and direct messages rather than through structured surveys.

Analysis approach

The evaluation data is analysed qualitatively rather than statistically. The sample size (limited by the number of independent builders during the research period) doesn't support statistical analysis. Instead, the focus is on identifying patterns: which steps caused the most difficulty, which documentation gaps were there, and which tool features were most useful. This is consistent with the iterative nature of DSR, where evaluation can lead back to further design refinement rather than serving as a final verdict [51].

Builder feedback was directly incorporated into design revisions. When a builder identified a documentation gap, the documentation was updated. When a tool was con-

[2] In practice, this boundary is difficult to maintain perfectly. The developer's presence in the community means that answers to general questions may indirectly help specific builders. This is acknowledged as a limitation.

fusing, the interface was revised. This iterative refinement is a core feature of the DSR methodology.

3.4 LIMITATIONS

Sample size. The number of independent builders is small, constrained by the project's timeline and the effort required for each build. The evaluation provides qualitative insights rather than statistically generalisable results.

Selection bias. Builders who attempt a multi-axis printer retrofit are self-selecting. They're likely more technically skilled and more motivated than the average maker. The evaluation therefore tests whether the system is reproducible by engaged enthusiasts, not by the general population.

Developer proximity. Despite the intention to test only public documentation, the developer was active in the Discord community during the evaluation period. Builders may have received indirect assistance through community interactions that wouldn't be available to future builders without the developer's presence.

Single evaluator. The analysis of builder feedback and documentation gaps was performed by the developer, who has an inherent bias toward finding the system successful. Independent evaluation would strengthen the validity of the findings.

But these limitations are typical of DSR evaluations in their first cycle. And the open-source release of the system enables future evaluation by independent researchers with larger and more diverse builder populations.

3.5 ETHICAL CONSIDERATIONS

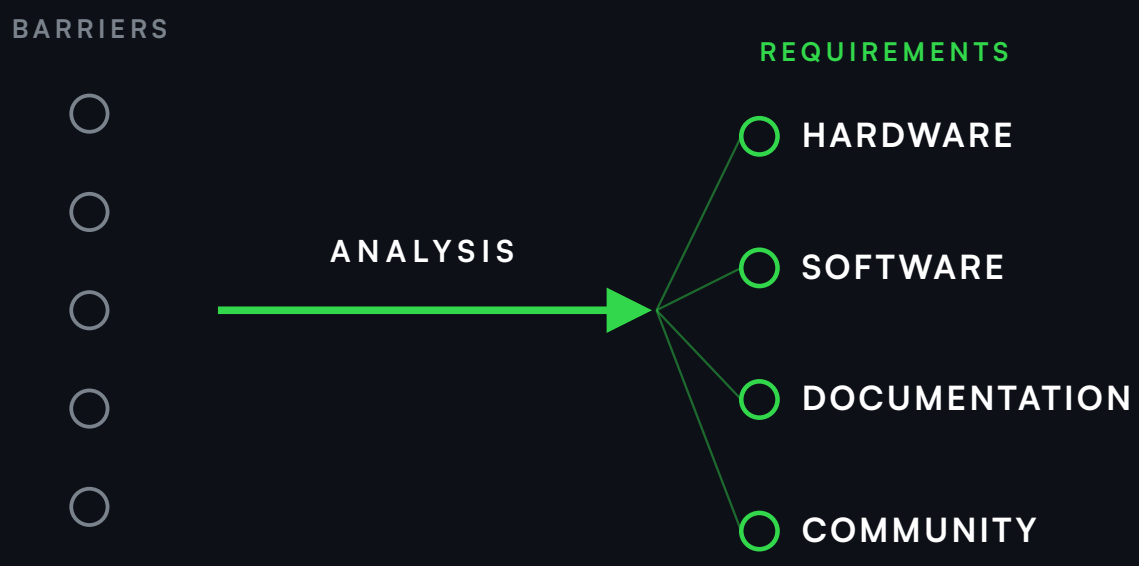
All builder participation in the evaluation was voluntary. Builders were informed that their feedback would be used in the thesis and consented to this use. No personal data beyond Discord usernames and build logs was collected. The project is open-source under GPLv3, and all documentation, tools, and designs are freely available. No commercial interests are involved in the research.

CHAPTER 04

04

ANALYSIS AND REQUIREMENTS

Turning the adoption barriers into concrete requirements: the gaps in the existing system, what makers actually need, and the specification the redesign must meet.



The literature review showed that multi-axis FDM isn't adopted because it isn't accessible, not because the technology doesn't work. This chapter pins those barriers down and turns them into requirements. It starts with Andersons' system measured against Rogers' framework, then looks at what other multi-axis projects do well and where they fall short, and ends with the requirements that the rest of the thesis implements.

4.1 CONCRETE GAPS IN THE EXISTING SYSTEM

The introduction introduced Andersons' prototype and the five adoption barriers it falls short on. This section drills into each barrier in technical detail, what specifically a maker has to deal with, so that the requirements that follow address real problems rather than generic complaints.

Gap 1: Single-printer dependency

The Robot Wrist 2 itself doesn't really depend on which printer it's bolted to. The rotary mechanism, gearing, and motor mounts are the same regardless of the printer model. Only the bracket that fastens the assembly to the X-carriage changes per printer. In principle that means adapting it to another printer is a matter of redesigning a single part. In practice, nobody knows that. The system was only ever demonstrated on a Creality Ender 5 Pro, the Fusion source files weren't structured to make the printer-specific part easy to swap, and there's no adaptation guide. The result is the same as a true single-printer dependency: a maker with a different printer has no obvious starting point. The fix is to make the modularity explicit (universal core, separate printer-specific bracket), demonstrate it on more than one printer, and document the adaptation process.

Gap 2: Commercial software dependency

The toolpath pipeline runs in Grasshopper, the visual scripting plugin for Rhinoceros 3D. Rhino costs roughly €995 for a commercial licence [53]. Grasshopper itself is free but ships only with Rhino, and the conformal slicing scripts assume the user can read and edit a node graph of dozens of components. After slicing, the G-code passes through a Python script that handles the inverse kinematics and post-processing, which has to be run from the command line with NumPy installed. Three separate tools, two paid software ecosystems, no GUI for non-programmers. The fix has to be a single browser-based toolchain with no installation, no licences, and no command line.

Gap 3: Setup complexity

The setup is all manual. The extra axes are enabled in modified Marlin source [18], so the builder has to set up a C++ toolchain, edit configuration headers, compile, and flash the binary by hand, and the inverse kinematics run as a separate Python pre-processing step between the slicer and the firmware. There's no routine to check the firmware is working once it's on the board. The geometric offsets the inverse kinematics depends on (Andersons calls these l_a and l_b) have to be measured physically on each build. And the calibration itself is dial gauge, reference cone, and sinusoidal fitting by hand in a spreadsheet, as illustrated in Fig. 4.1. Four steps, no tooling, no visual feedback, no path through it for a non-expert. The fix has to be a guided setup workflow with a tool for each step: firmware flashing, validation, offset measurement, and calibration.

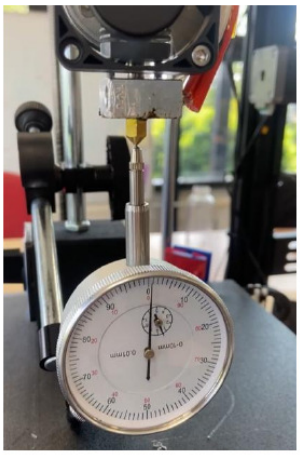


FIG. 4.1 Dial gauge used by Andersons to record Z-axis deviation across A-axis positions [18].

Gap 4: Documentation

The only documentation is the thesis itself. There's no BOM with supplier links, no wiring diagram, no assembly photos, no troubleshooting section. A maker who wants to build the system has to reverse-engineer the parts list from CAD screenshots and work out which pin goes where from the firmware code. The fix has to be a build-guide-style documentation: BOM, wiring, photos, printable files with settings, in a format that walks someone through the build step by step.

Gap 5: No community or ecosystem

The project has no community. No GitHub repo with issues enabled, no Discord, no forum thread, no mailing list. A builder who hits a problem has nobody to ask, and the project has no mechanism for absorbing improvements or platform adaptations from people who do figure things out. As Antoniou et al. [20] found, an active community is one of the strongest predictors of OSHW project success. The fix has to be community infrastructure that exists from day one, not as an afterthought.

4.2 ANALYSIS OF EXISTING SOLUTIONS

Andersons' system isn't the only attempt at accessible multi-axis printing. Appendix B surveys all existing projects; the two most relevant are summarised here. Each addresses part of the problem, but not all of it.

	Multi-platform	Accessible software	Build documentation	Community support	Open-source
Andersons (2023)	~	×	~	×	~
Open5x (2022)	×	×	~	×	✓
RotBot (2021)	×	~	~	×	~
Pentaxis (2021)	×	~	~	×	✓
3Z (2025)	×	~	~	×	✓
Fractal 5 Pro (2025)	×	~	~	×	✓
Voron 5-axis (2023)	×	×	~	×	✓
FullControl (2021)	×	~	~	×	✓

✓ = addressed × = not addressed ~ = partially

TABLE 4.1 Gap analysis of existing multi-axis FDM projects against the five adoption barriers. See Appendix B for full system descriptions.

Open5x

Open5x [22] is an open-source 5-axis retrofit for the Prusa i3 MK3s. It provides 3D-printable hardware designs and a GUI-based conformal slicer implemented as a Grasshopper plugin for Rhino. Open5x scores well on hardware openness (designs are freely available, most parts are printable) but has significant limitations:

- The rotary table mounts on the Y-axis bed, adding mass to the most active linear axis. This limits speed and accuracy.
- The slicer runs inside Rhino/Grasshopper, inheriting the same commercial software dependency as Andersons' system. The authors acknowledge this: "This dependency on a commercial product means that our solution is not completely open source" [22].
- Documentation is limited to the research paper. No build guide, no community.
- Designed for a single printer (Prusa MK3s) with no adaptation framework for other models.

RotBot

The RotBot is a 4-axis conical printer that uses only a single rotational axis (continuous rotation around Z) with a tilted nozzle. It trades the full flexibility of 5-axis printing for simplicity: fewer axes means simpler kinematics, simpler calibration, and simpler firmware. The design has gained visibility through CNC Kitchen [18].

RotBot's strength is its simplicity, but its limitation is fundamental: with only 4 axes and a fixed tilt angle, it can't achieve the same range of non-planar orientations as a full 5-axis system. It's a different trade-off, not a solution to the same problem.

Summary

Table 4.1 maps the existing projects against the key adoption barriers. The pattern is clear: most projects are open-source, but the other barriers remain largely unaddressed. No project works across multiple printer platforms. Software accessibility is mixed: Open5x depends on commercial Rhino, while RotBot uses free Python scripts. Documentation rarely goes beyond a research paper. And none has an active community for builder support. These gaps define what the requirements in this chapter must address.

4.3 STAKEHOLDERS

Rep5x isn't built in a vacuum. Several groups have a stake in whether it succeeds, and their interests shaped the requirements as much as the technical gaps did (Table 4.2).

Hobbyist makers are the primary stakeholders: the technically engaged builders the system is for, profiled in detail in the next section. Their willingness to attempt and finish a build is the measure of success, and the accessibility requirements exist for them.

Community contributors are the makers who give back, reporting issues, testing parts, sharing printer adaptations, and occasionally contributing code. They carry outsized influence: the feedback loop described later turned their problems directly into design changes, and a healthy contributor base is what lets the project outlive a single developer.

The maintainer. For now the project is largely the work of one developer, which is both its main source of momentum and its main risk. Much of the community and governance requirements exist to reduce that dependency over time.

Janis Andersons, supervisor and author of the prototype this work builds on, has a stake in seeing the approach validated and carried further. The Robot Wrist 2 mechanism is inherited largely unchanged, so the hardware credit is shared and the contribution here is deliberately about access rather than mechanism.

Stakeholder	Stake	Influence
Hobbyist makers	An accessible multi-axis system to build and use	High
Community contributors	Improving, extending, and adapting the system	High
Maintainer (developer)	Building and sustaining the project	High
Janis Andersons (supervisor)	Validating and continuing the prototype	Medium
University of Twente (DPM)	Academic rigour and a documented contribution	Medium
Upstream OSS (Marlin, RepRap, Klipper)	Downstream use and contributed fixes	Low—medium
Peer projects, makerspaces, fab labs	The wider ecosystem and future reach	Low (for now)

TABLE 4.2 Project stakeholders, their stake in Rep5x, and their influence on its direction.

The University of Twente, specifically the Department of Design, Production and Management, is the academic stakeholder: the work is assessed as a master's thesis, so it has to be properly argued and documented, not just built.

Upstream open-source projects, Marlin, RepRap, Klipper, and the wider RepRap lineage, are both dependencies and beneficiaries. Rep5x is a Marlin fork that leans on the maker ecosystem's conventions, and in return feeds fixes and a documented five-axis configuration back into it.

The wider ecosystem, peer projects like Open5x and RotBot and settings like makerspaces and fab labs, sits further out: little direct involvement now, but the most likely route to broader adoption if the project gains traction.

4.4 TARGET USER PROFILE

The requirements must be grounded in a clear understanding of who the system is for. As defined in the introduction, the target audience is technically engaged hobbyists: makers who own a working FDM printer, are comfortable with basic electronics and wiring, and enjoy the build process. This is the demographic that builds Voron printers from self-sourced kits or assembles Prusa kits from scratch (Fig. 4.2).

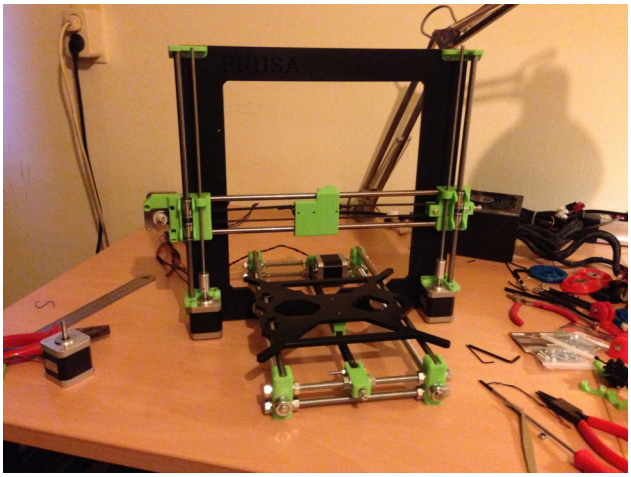


FIG. 4.2 A partially assembled Prusa i3 kit on a builder's workbench [54].

This audience has specific characteristics that affect the requirements:

- **They own consumer FDM printers.** The system must work with popular, affordable printers, not custom or premium machines. Creality's Ender series is the most common platform in this segment.
- **They can follow detailed instructions but aren't engineers.** Assembly instructions need to be explicit and visual. "Attach the motor to the frame" isn't enough; "secure the NEMA 17 motor to the bottom plate using four M3×8 screws, with the shaft pointing upward as shown in Figure X" is.
- **They don't have specialised tools.** Beyond a basic electronics toolkit (soldering iron, multimeter, hex keys), no specialised tools should be required. No CNC machining, no precision measuring instruments, no oscilloscopes.
- **They're comfortable with 3D printing.** They know how to slice and print parts, adjust settings, and troubleshoot common print issues. This means the 3D-printed components of the retrofit can be printed by the builder on their own machine.
- **They're budget-conscious.** A total retrofit cost under €200 (excluding the base printer) keeps the project accessible. Going above this threshold risks losing the hobbyist audience to "why not just buy a better printer" reasoning.
- **They're online.** They use Discord, Reddit, YouTube, and GitHub. Community support through these channels is natural for them.

4.5 REQUIREMENTS SPECIFICATION

The requirements are organised into four categories: hardware, software, documentation, and community. Each requirement is traceable to one or more of the adoption barriers identified in the analysis above and the six high-level requirements from the literature review.

Hardware requirements

HW-1 Modular architecture. The design must separate universal components (rotary mechanism, gearing, motor mounts) from printer-specific components (mounting adapter). Adding support for a new printer should require designing only a single adapter part, not redesigning the entire system.

HW-2 Printable on consumer printers. All structural components must be printable on a standard FDM printer with a 180×180×180 mm build volume, using PLA or PETG, without support structures where possible. Parts

that approach build volume limits undermine self-replication.

HW-3 COTS components only. All non-printed components (motors, bearings, fasteners, controller board, wiring) must be commercially available from global suppliers. No custom-machined parts, no region-specific suppliers, no obsolete components.

HW-4 Total cost under €200. The complete BOM, excluding the base printer and standard tools, must stay under €200. This includes all printed parts (material cost), electronics, motors, fasteners, and wiring.

HW-5 CB axis configuration. The system inherits Andersons' head-head configuration with both rotational axes at the print head, re-labelled in Rep5x as C (continuous yaw rotation) and B (tilt) to match the LinuxCNC and Siemens conventions [55], [56]. The C-axis must support continuous rotation (no cable wrap limitations). The B-axis must provide sufficient tilt range for non-planar printing while maintaining clearance.

HW-6 Multi-platform support. The system must be demonstrated on at least two different printer platforms to validate the modular architecture. Initial targets: Creality Ender 5 Pro and Ender 3 V3 SE.

Software requirements

SW-1 Browser-based. All tools must run in a standard web browser (Chrome, Edge) with no installation, no plugins, and no command-line interaction. The Web Serial API is used for hardware communication.

SW-2 No commercial dependencies. No tool may require a paid licence, a proprietary software package, or a specific operating system.

SW-3 Complete workflow coverage. The tool suite must cover the full workflow from firmware configuration through calibration to G-code generation. No step may require editing source code, running a command-line tool, or installing an external application.

SW-4 Direct printer communication. A tool for controlling the modified printer directly from the browser via serial connection, enabling real-time interaction during setup and calibration.

SW-5 Shared inverse kinematics. The IK calculations (coordinate transformations using LC and LB parameters) must be implemented as a shared library used consistently across all tools, not reimplemented in each tool separately.

Documentation requirements

DOC-1 Step-by-step build guide. A guide covering the assembly process from printing parts through mechanical assembly, wiring, firmware installation, and first calibration. Each step must include photographs or renders showing the expected result.

DOC-2 Complete BOM with sourcing. A bill of materials listing every component with quantity, specification, and links to suppliers. Multiple supplier options should be provided where possible to avoid single-supplier dependencies.

DOC-3 Wiring diagrams. Clear diagrams showing all electrical connections between the controller board, motors, endstops, and power supply. Pin assignments must be explicit.

DOC-4 Print files with settings. All 3D-printable parts provided in both manufacturing format (3MF/STL) and editable format (STEP), with recommended print settings (layer height, infill, material, orientation).

DOC-5 Project website. A public-facing website that serves as the entry point for the project: what it is, how to get started, links to documentation and tools. The website must be accessible to non-technical visitors.

DOC-6 Adaptation guidelines. Documentation explaining how to design a printer-specific adapter for an unsupported printer model. This enables community-driven expansion to new platforms.

DOC-7 Semantic versioning. Hardware and software revisions must be tracked with semantic versioning (MAJOR.MINOR.PATCH) so that builders know which parts are compatible and which require reprinting or reflashing.

Community requirements

COM-1 Real-time support channel. A Discord server with organised channels for build support, troubleshooting, and general discussion. This is the primary community platform.

COM-2 Version-controlled repository. All hardware designs, software source code, and documentation hosted on GitHub, with contribution guidelines for outside changes.

COM-3 Open-source licence. All components released under GPLv3, enabling modification, redistribution, and self-replication.

COM-4 Feedback mechanism. A structured way for builders to report problems, suggest improvements, and contribute modifications. The Discord server is the channel for this, with GitHub used for code contributions and pull requests.

4.6 REQUIREMENTS TRACEABILITY

The traceability map (Fig. 4.3) gives a visual path from barrier to requirement to deliverable, and Table 4.3 maps each requirement category to the adoption barriers it addresses and the Rogers attribute it improves. Every barrier maps to at least one requirement, and every requirement traces back to a barrier from the analysis.

The requirements span hardware, software, documentation, and community because the literature review showed that adoption depends on all four working together, not any single one in isolation. The next chapter describes how all four are implemented.

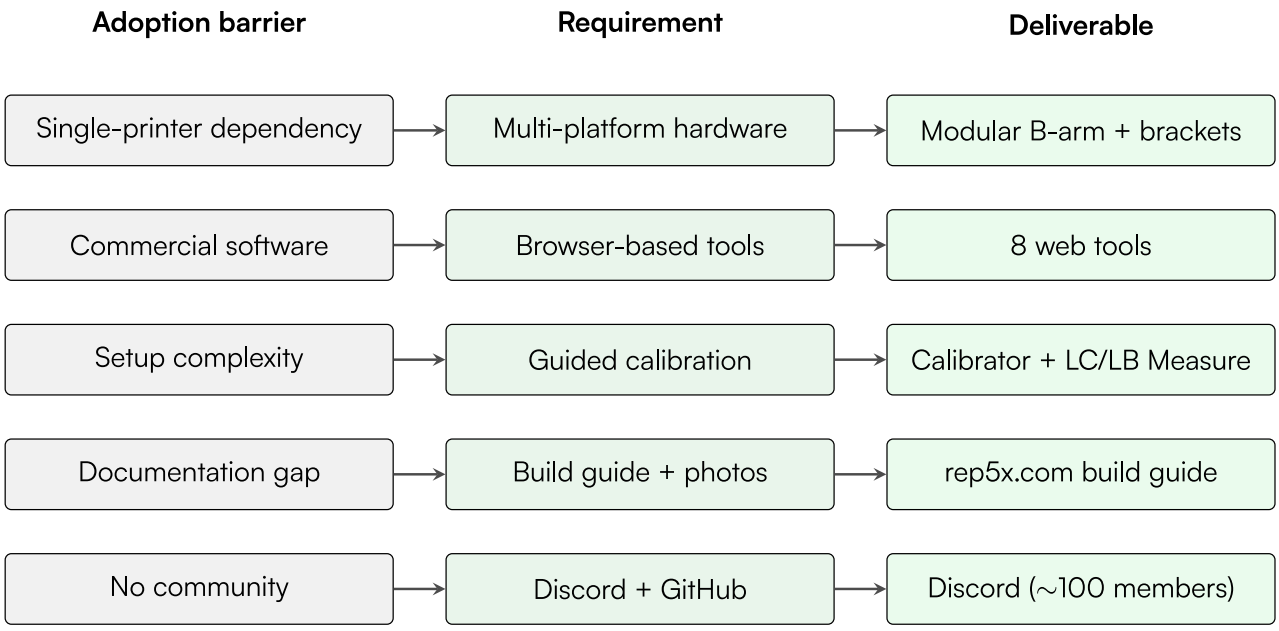


FIG. 4.3 Traceability from adoption barriers through requirements to delivered components.

Requirement	Barrier addressed	Rogers attribute
HW-1 to HW-3	Single-printer dependency, hardware cost	Compatibility
HW-4	Financial barrier	Trialability
HW-5, HW-6	Technical feasibility across platforms	Compatibility
SW-1, SW-2	Commercial software dependency	Complexity, Trialability
SW-3, SW-4	Fragmented toolchain, specialist knowledge	Complexity
SW-5	Software reliability and consistency	Complexity
DOC-1 to DOC-4	Missing build documentation	Complexity, Trialability
DOC-5	Low project visibility	Observability
DOC-6	Single-printer dependency	Compatibility
DOC-7	Reproducibility across versions	Compatibility
COM-1, COM-4	No community support	Observability, Complexity
COM-2, COM-3	No contribution mechanism	Observability

TABLE 4.3 Requirement categories mapped to the adoption barriers they address and the Rogers attribute each improves.

CHAPTER 05

05

DESIGN AND DEVELOPMENT

Where the effort actually went: small changes to Andersons' mechanism, a Marlin fork that hides the kinematics, the browser-based toolchain that replaces commercial software and the command line, and the open-source release that ties it together.

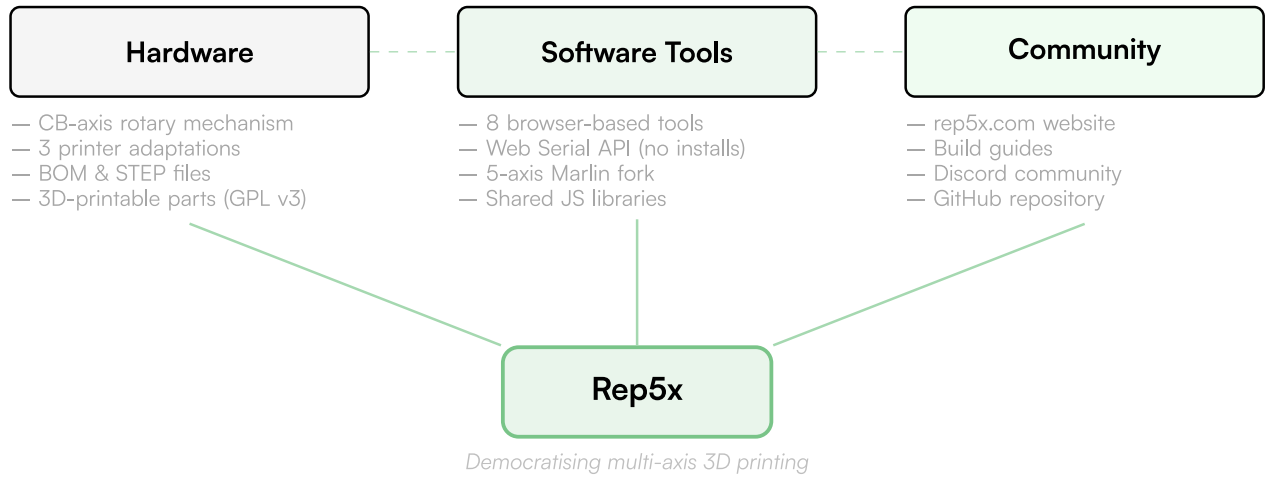


FIG. 5.1 Overview of the Rep5x system: hardware, software tools, and community, each aimed at an adoption barrier from the analysis.

This chapter describes the Rep5x system: the hardware, the firmware, and the eight browser-based software tools. Every design choice traces back to the requirements in the analysis chapter and the target user profile, so the focus here is less on what was built and more on *why* it was built that way.

The chapter is organised in three parts. First, the hardware: how the design prioritises buildability and adaptability over just performance. Second, the firmware: how Marlin was modified to hide kinematic complexity from the user. Third, the software tools: how eight browser-based applications replace what would otherwise require commercial software, command-line scripts, and specialist knowledge. Fig. 5.1 lays out the complete system, and the photographs (Fig. 5.2) show what the finished retrofit looks like on a stock printer.

5.1 HARDWARE DESIGN

Mechanical architecture

The rotary mechanism in Rep5x is Andersons' Robot Wrist 2 [18], extended with the continuously rotating yaw axis later developed at the University of Twente by Meijerman

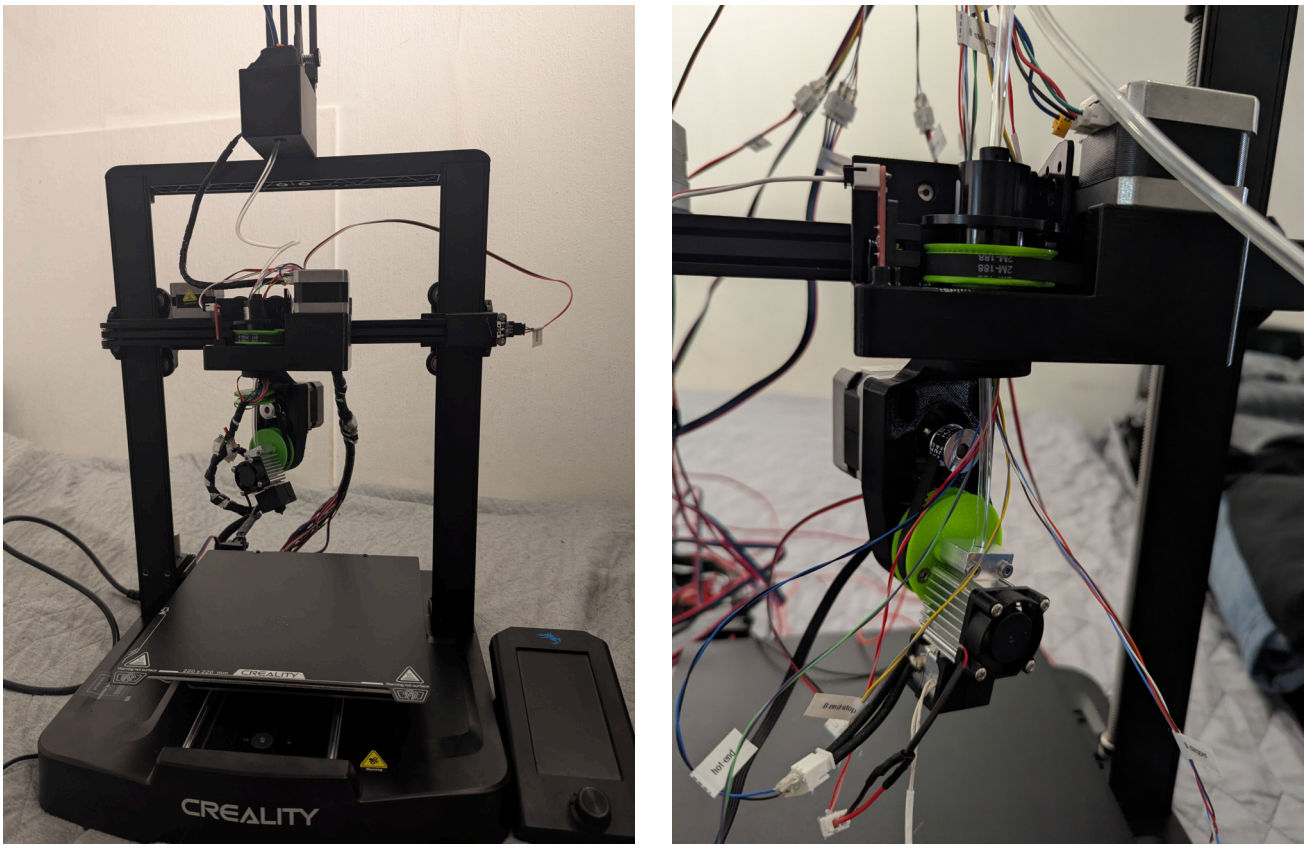


FIG. 5.2 The Rep5x 5-axis retrofit on a consumer printer. Left: complete system on a Creality Ender 3 V3 SE. Right: close-up of the RW2 head in the printing position.

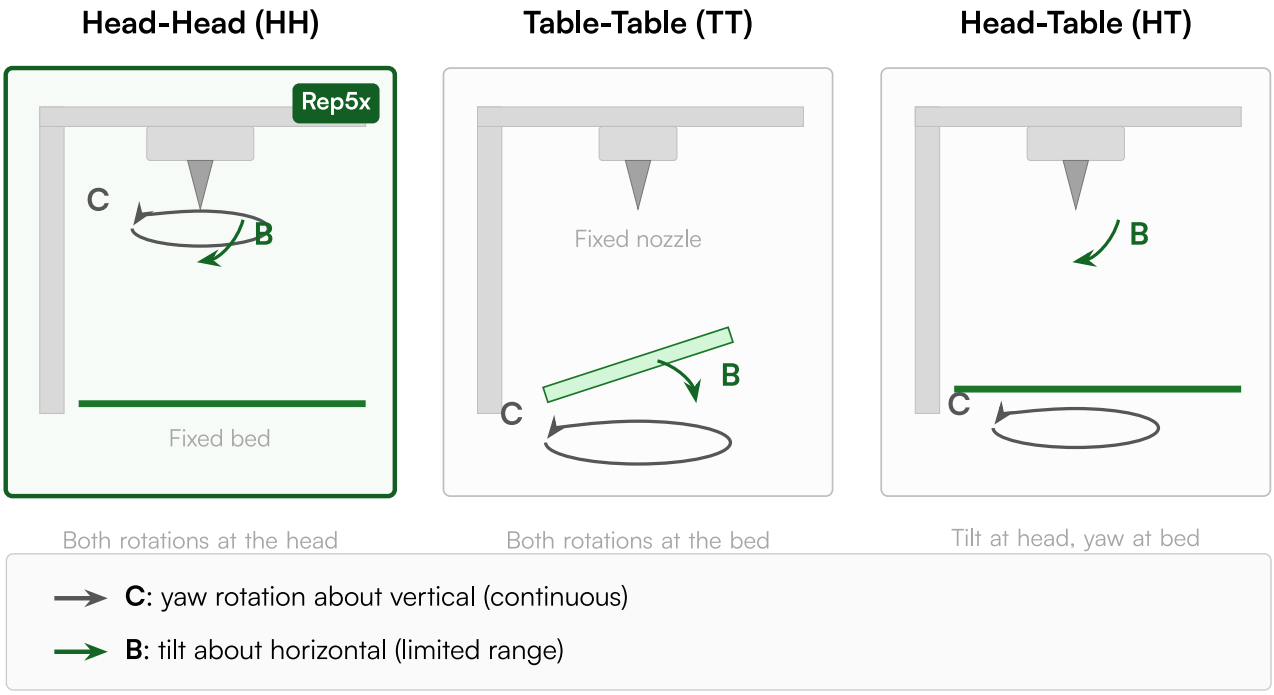


FIG. 5.3 Three kinematic configurations for adding two rotational axes: head-head (HH), table-table (TT), and head-table (HT).

and Veldman [57], [58]. It's a head-head (HH) configuration with both rotational axes placed at the print head: a C-axis for continuous yaw rotation, and a B-axis for tilt. Fig. 5.3 compares this to the alternative TT and HT layouts. The mechanism bolts to the printer's X-carriage in place of the stock hot-end assembly, and the part is printed onto a stationary bed while the nozzle tilts and rotates above it.

Andersons' thesis labels the yaw axis A and the tilt axis B. Rep5x re-labels the yaw axis as C, following the LinuxCNC and Siemens conventions for 5-axis machining centres, where C is the rotation around Z and A and B are rotations around X and Y respectively [55], [56]. The change was made after community feedback to align Rep5x's G-code with what other multi-axis CAM users expect.

Two practical adaptations were made on top of this inherited design. First, the C-axis optical homing sensor replaces the hall-effect sensor used previously, because optical sensors are more common in consumer 3D printing and easier to source. Second, two additional X-carriage brackets were designed for the Creality Ender 3 V3 SE and Ender 3 Pro, alongside the existing Ender 5 Pro bracket, so the system could be demonstrated on more than one printer. The rotary mechanism itself, the slip ring, belts, pulleys, bearings, motors, and the structural parts of the wrist, is otherwise unchanged from the inherited design.

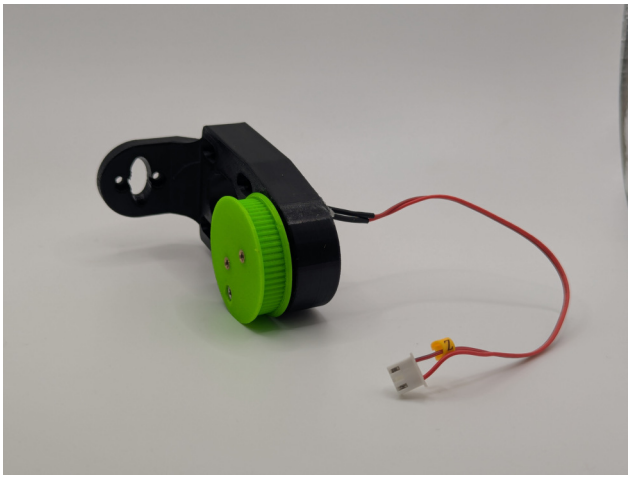


FIG. 5.4 The B-arm, the tilting member of the RW2 mechanism.

The value of the hardware contribution isn't in the mechanism itself. It's in making the design easier for someone else to build: Andersons' bracket-and-universal split was carried forward and validated by designing brackets for two additional printers, and the whole BOM was documented with global supplier links.

Modular design

Andersons' design was demonstrated on the Ender 5 Pro, with the printer-specific part already isolated to a single X-carriage bracket. Rep5x preserves that split and validates it on two additional printers: the universal parts (the rotary mechanism itself) sit unchanged across all three, and only the bracket changes per machine.

Universal components make up the bulk of the mechanism:

- **C-driven pulley**: receives the C-axis drive belt, houses two 61804 thin-section bearings for smooth rotation.

- **B-arm** (Fig. 5.4): the tilting arm that holds the hot-end assembly. A cable routing slit (added in v1.1.0) runs along the arm so the wiring can be lifted out sideways, which means the B-arm can be swapped for a replacement (after a break) or a newer revision without having to rethread the cables through a closed channel.
- **B-driven pulley**: receives the B-axis drive belt, supported by two 608 bearings (standard skateboard bearings) in the B-arm.
- **Slip ring holder**: mounts the MST-005-12A slip ring that passes electrical connections through the continuously rotating C-axis. All 12 channels (2 A each) are used: the hot-end heater, thermistor, part-cooling fan, B-axis motor phases, and B-axis microswitch signal.
- **Spacers**: position the hot-end at the correct height relative to the B-axis pivot.

Printer-specific components consist of a single carriage mount that bolts to the printer's X-carriage and holds the entire universal assembly. Adding support for a new printer requires designing only this one part. Three carriage mounts have been designed:

- **Ender 5 Pro**: the simplest adaptation, requiring only the carriage mount and a Bowden tube extension.
- **Ender 3 V3 SE**: requires additional parts (relocated X and Z endstops, external Bowden extruder mount) because the stock printer uses a direct-drive extruder that isn't compatible with the rotating mechanism.
- **Ender 3 Pro**: requires a relocated Z endstop (moved from Z-min to Z-max) but otherwise minimal changes. This bracket was designed for a community member who is building the system on their own Ender 3 Pro.

This modular split, illustrated in Fig. 5.5, directly addresses the modularity requirement.

The core mechanism is identical regardless of the base printer; only the interface changes (Fig. 5.6).

Drive system

Both axes use GT2 timing belts with 20-tooth pulleys driven by NEMA 17 stepper motors, all unchanged from the RW2. The C-axis uses a 188 mm belt and the B-axis a 158 mm belt, both 6 mm wide.

The C-axis requires continuous rotation, which introduces a wiring challenge. Conventional wire connections would wrap around the rotating axis and break. The MST-005-12A slip ring solves this: it provides 12 electrical channels through a rotating joint with a 5 mm bore. All connections between the stationary carriage and the rotating hot-end assembly pass through this slip ring (Fig. 5.7).

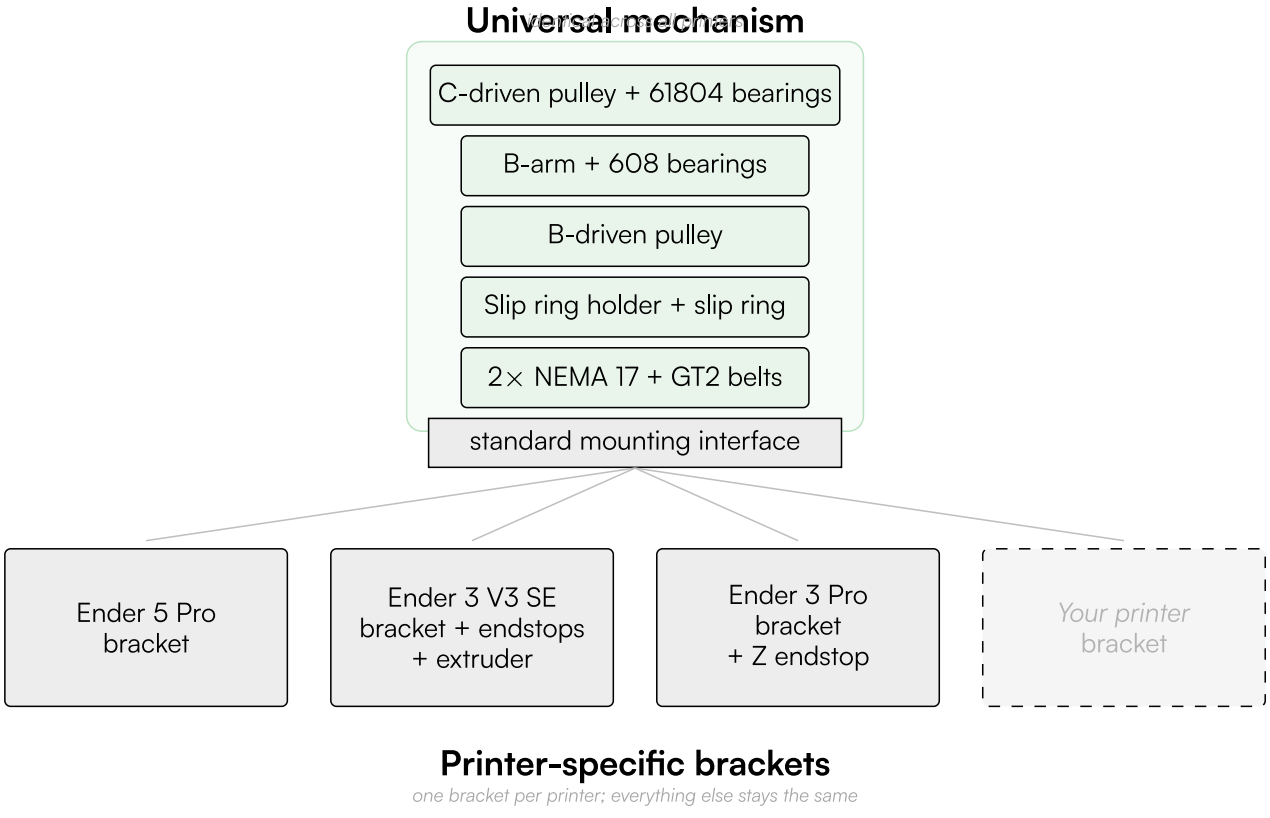


FIG. 5.5 Modular hardware architecture: the universal mechanism (top) is shared across printers; only the X-carriage bracket changes per model.

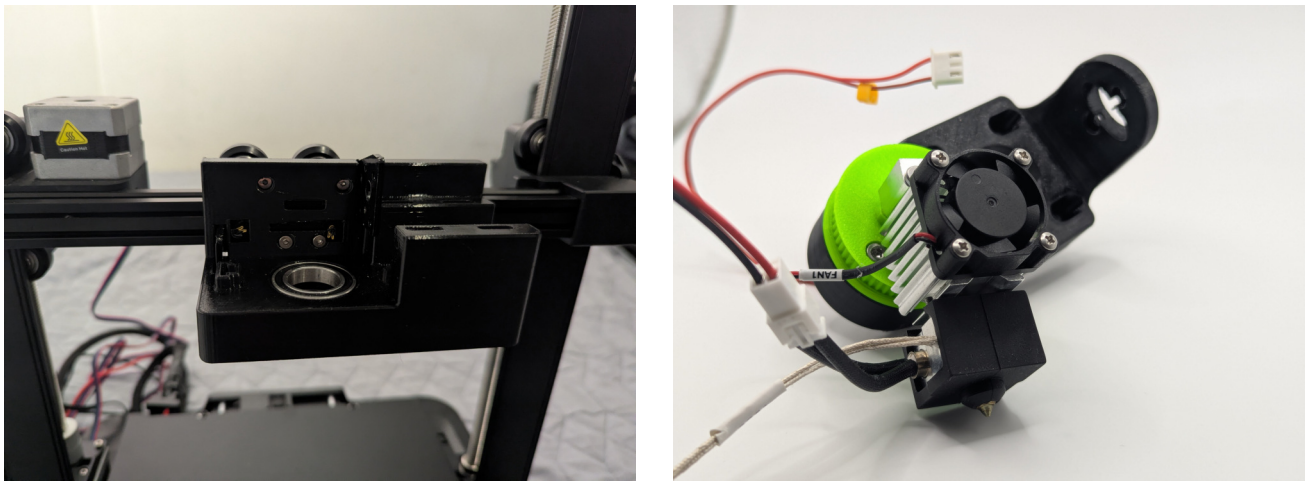


FIG. 5.6 Mechanical assembly of the RW2 mechanism on a Creality Ender 3 V3 SE. The universal rotary mechanism (right, hot-end mounted on the B-arm) bolts onto a printer-specific carriage bracket (left, installed on the X-gantry).

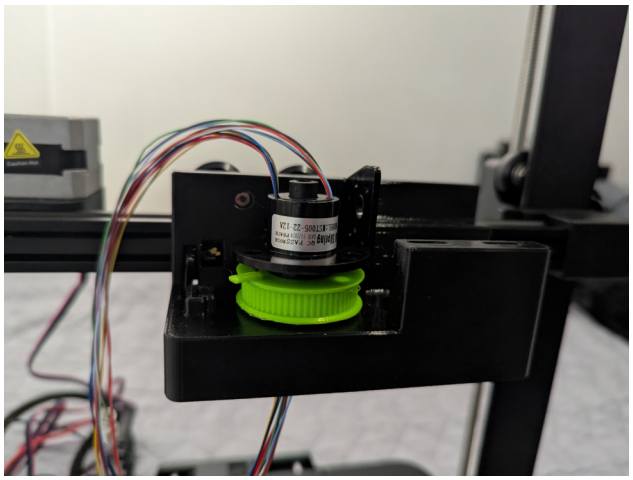


FIG. 5.7 The MST-005-12A slip ring seated in its 3D-printed holder.

Homing and endstops

The C-axis uses an optical sensor for homing. The sensor provides a clean digital signal when the reference tab on the C-driven pulley passes through it, establishing a repeatable zero position for the continuous rotation axis.

The B-axis uses a microswitch triggered by a set screw on the B-driven pulley. A heat-set insert in the 3D-printed pulley provides a durable thread for the set screw, allowing fine adjustment of the homing position. Because the microswitch sits on the rotating C-axis assembly, its signal runs through the slip ring alongside the rotating-side power and motor wiring.

Electronics

The control electronics are based on the BTT Octopus V1.1 controller board (Fig. 5.8). The Octopus was chosen over the stock Creality boards for two reasons: it has eight stepper driver slots (the 5-axis build needs six, X, Y, Z, extruder, C, B, with the spare slots leaving room for a dual-Z upgrade if the extra mass of the rotary head makes the gantry sag), and it accepts TMC2208 drivers in UART mode, which provide the silent operation and mid-move current control that the rotary axes benefit from. Stock 8-bit Creality boards don't offer either. On the Ender 3 V3 SE the Octopus drops straight into the original Creality electronics housing, so externally the printer looks stock; on the other supported printers it sits in an enclosure mounted to the frame, with the original Creality board left disconnected.

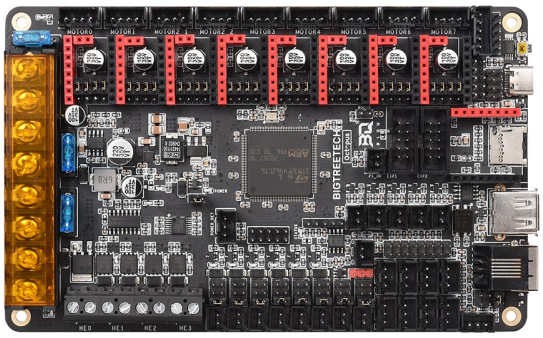


FIG. 5.8 The BIGTREETECH Octopus V1.1 32-bit controller board used in Rep5x [59].

All five motor-driven axes use TMC2208 drivers, configured from the Firmware Builder rather than via jumper settings on the board. UART mode lets the per-axis current and microstep settings be written from software, so changing them later only requires reflashing rather than physically resoldering jumpers.

Everything on the rotating C-axis side runs through the slip ring, and the full 12 channels are in use: 2 for the hot-end heater, 2 for the thermistor, 2 for the part-cooling fan, 4 for the B-axis stepper's phase wires, and 2 for the B-axis homing microswitch (which is mounted on the rotating assembly so it can sit alongside the B-driven pulley). Grounding is shared with the stationary side. The slip ring's 2 A-per-channel rating is comfortable for all of these: the 40 W hot-end pulls roughly 1.7 A, the thermistor, fan, and microswitch signals all draw well under an

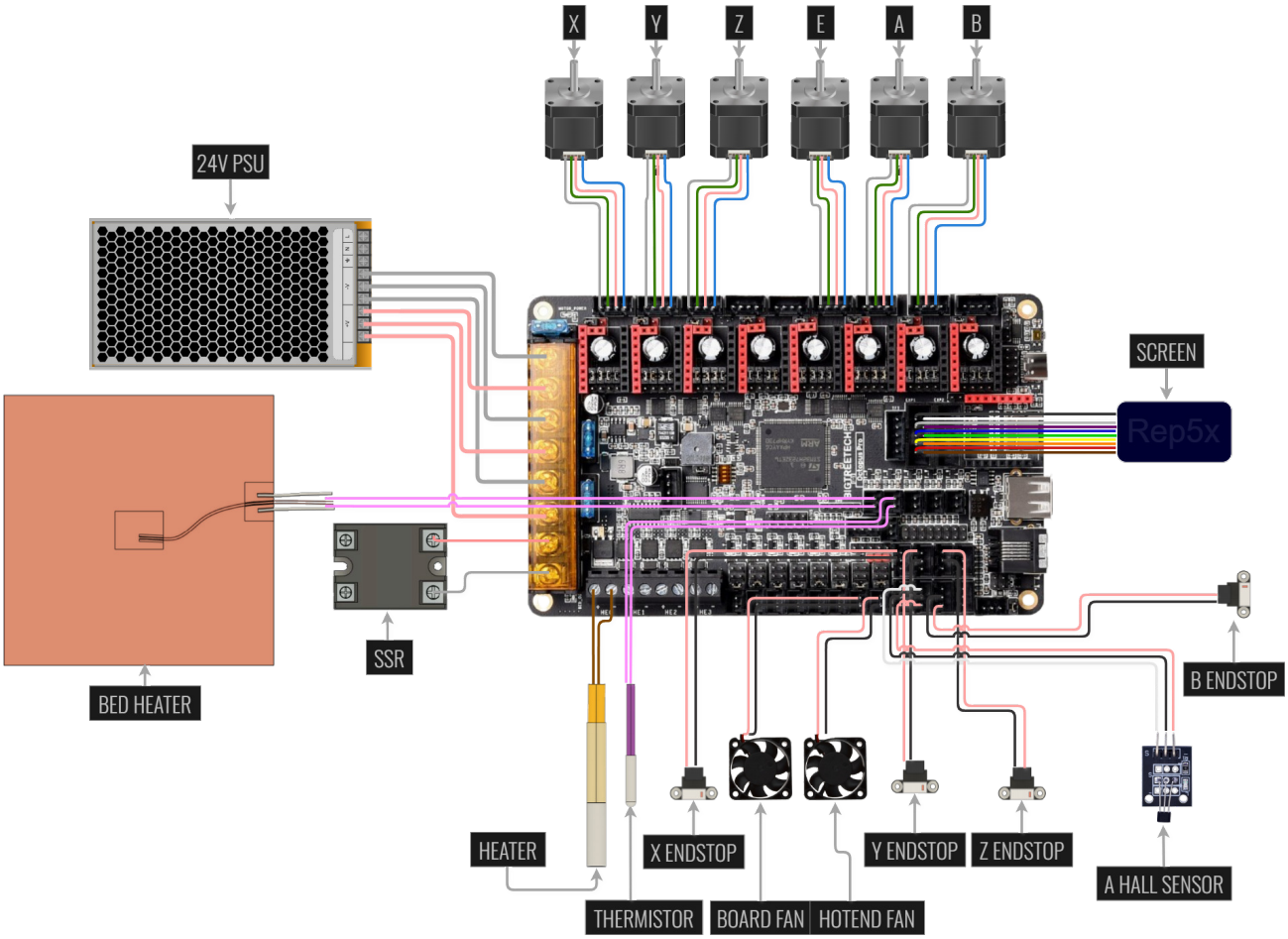


FIG. 5.9 Wiring diagram for the BTT Octopus V1.1 controller board, showing connections for all six axes (X, Y, Z, extruder, C, B), endstops, and hot-end components. Source: `build-guide/universal-parts/electronics/`.

amp, and the B-axis stepper sees at most 1.2 A RMS at typical UART current settings. The C-axis optical sensor is the one homing signal that doesn't go through the slip ring, because the sensor body is fixed to the stationary carriage and only reads a reference tab on the rotating pulley as it passes by.

The wiring layout and the cable routing on the physical build are shown in Fig. 5.9 and Fig. 5.10.

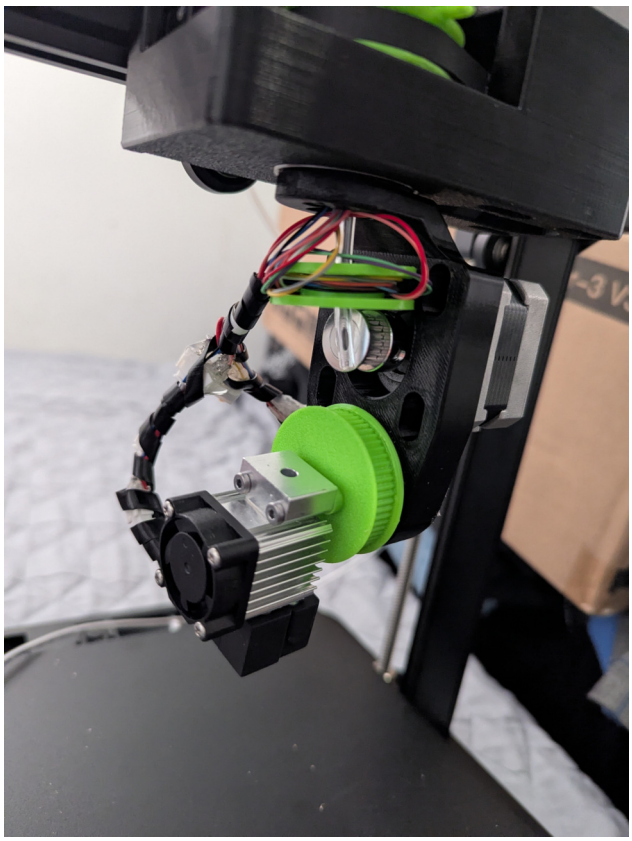


FIG. 5.10 Cable routing through the B-arm slit (introduced in v1.1.0).

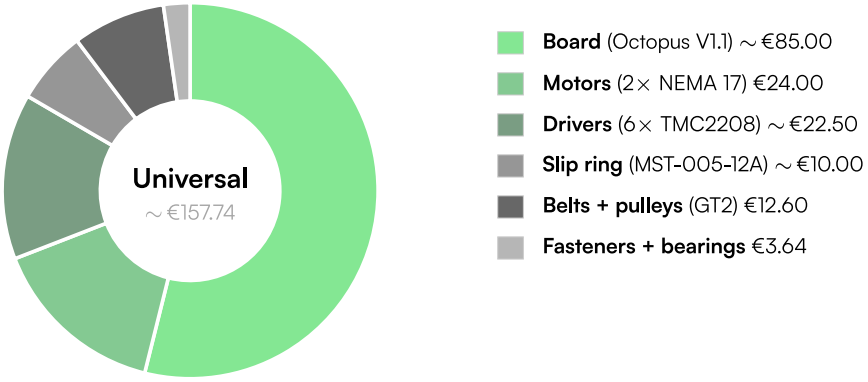


FIG. 5.11 Approximate cost composition of the universal BOM, estimated from AliExpress listings.

Bill of materials and cost

The universal BOM consists of two NEMA 17 stepper motors, two GT2 belts and pulleys, four bearings (two 608, two 61804), one slip ring, two endstop sensors, fasteners, and wiring. All components are globally available COTS parts, e.g. standard NEMA 17 steppers, 608 skateboard bearings, and M3 fasteners. The total cost of the universal components is approximately €150, with printer-specific parts adding €10—30 depending on the model.[3] This keeps the total under the €200 target. The cost split is shown in Fig. 5.11.

Design for printability

All structural components are designed to print on a standard FDM printer with a 180×180×180 mm build volume. PETG is recommended for its combination of strength, temperature resistance, and ease of printing. No support structures are needed for the parts, all overhangs are printable using wave overhangs. On a Bambu Lab A1 mini, the universal parts print in about 2 hours; a complete set including the printer-specific parts takes about 4 hours.

Parts are provided in both 3MF (for printing) and STEP (for modification) formats. The STEP files allow builders to modify the design in any CAD software, which is essential for community-driven adaptation to new printer platforms.

Rep5x Camera

Calibration is easier to get right when you can see what the nozzle is doing. The Rep5x Camera is a small, magnetically mounted USB camera that gives the calibration tools a live view of the tool centre point, so axis offsets can be read by computer vision rather than by eye and dial gauge. It's deliberately cheap and printable: the whole module costs about €10.50.

The core is an OV9726 USB camera module (1 MP, 70° field of view) that the host recognises as a plug-and-play webcam with no drivers. Four 5 mm straw-hat white LEDs are soldered straight to the camera board's VCC and GND pads to light the nozzle evenly, and the lens twists in its thread for manual focus. The housing is three printed parts in PLA or PETG: a base, a top cover, and a separate sideways mount for Z-axis calibration. 6×2 mm neodymium magnets press-fit into the base with no glue, so the camera snaps onto the bed or the Z-axis mount and can be repositioned freely; the hole pattern is Gridfinity-compatible.

Assembly is quick (Fig. 5.12): print the housing, solder the four LEDs to the board, drop the camera in, press the magnets into place, and plug in the USB cable. The calibrator at `calibrator.rep5x.com` reads the feed straight through the browser, and focus is set by twisting the lens while watching the live image.

[3] All components were sourced from AliExpress. European distributors generally carry the same parts at noticeably higher prices, with faster shipping as the trade-off.

FIG. 5.12 The Rep5x Camera, a roughly €10.50 USB camera calibrator. Left: four white LEDs soldered directly to the OV9726 board for even nozzle lighting. Right: the finished, magnet-mounted module, read live by the browser calibrator.

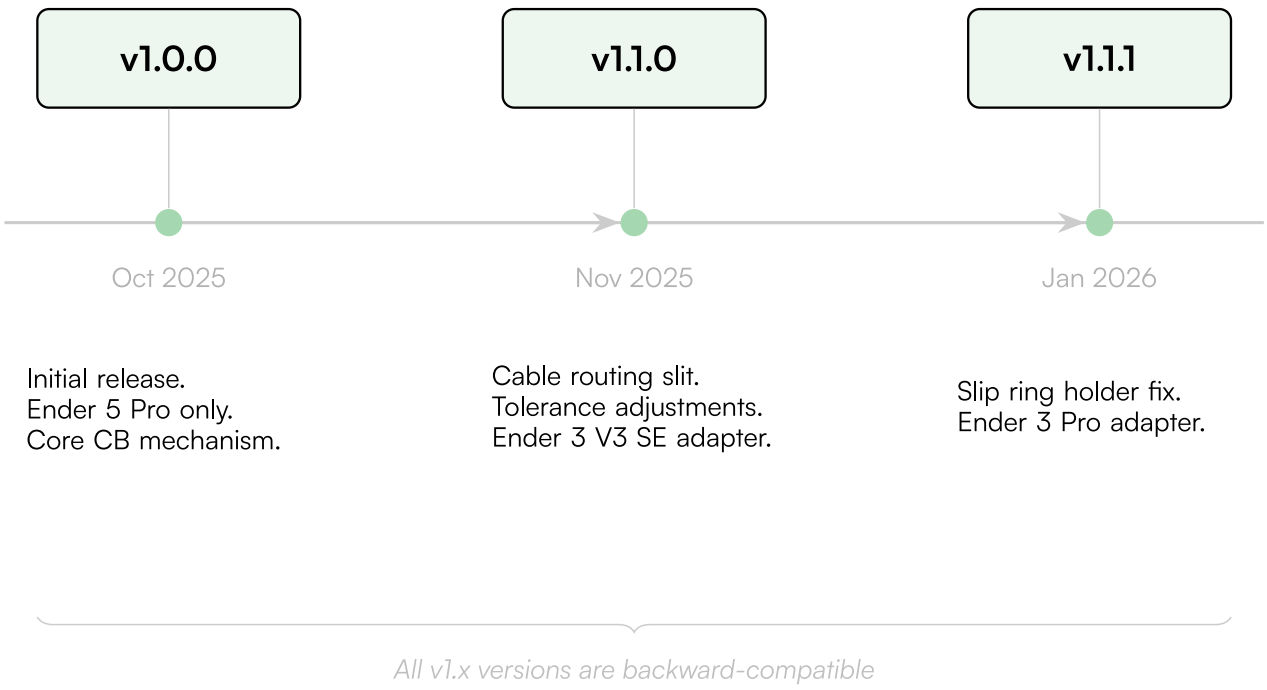


FIG. 5.13 Hardware version history. All v1.x versions are backward-compatible.

Version control

Hardware revisions follow semantic versioning: MAJOR.MINOR.PATCH, following the policy formalised in the repository's `VERSIONING.md`. MAJOR is for breaking changes (new parts required), MINOR for compatible improvements (optional upgrades), PATCH for fixes (direct replacements). The system launched at v1.0.0 (October 2025) and has progressed through v1.1.0 (November 2025, adding cable routing improvements and tolerance adjustments) to v1.1.1 (January 2026, slip ring holder clearance fix), as shown in Fig. 5.13. v1.x parts are largely backward-compatible: a builder can mix v1.0.0 and v1.1.0 parts freely, the exception being the v1.1.x slip ring holder, which needs the matching v1.1.x carriage mount.

The versioning is visible in two places. Filenames embed the version so it's obvious in a print queue (e.g. `B_arm_v1.1.0.3mf`), and the 3D-printed-parts directory is organised into per-version subdirectories (`v1.0.0/`, `v1.1.0/`, `v1.1.1/`) alongside a `current/` directory that always points to the latest recommended set. A `CHANGELOG.md` in the same folder records what changed and why. Builders who want the latest can grab `current/`; builders who want to reproduce a specific build or test a regression can grab a pinned version.

5.2 FIRMWARE

Marlin modifications

As discussed in the case studies, Marlin's configuration model is one of the accessibility barriers in FDM printing: hundreds of parameters in C++ header files, requiring recompilation for any change. For a 5-axis system, the firmware challenge is even steeper. Andersons [18] handled inverse kinematics as a separate Python script that pre-processed G-code before sending it to the printer. This works, but it means the maker needs Python installed, must understand the command line, and has to regenerate the G-code every time the kinematic parameters change. That's a significant barrier for someone who isn't a software developer.

Rep5x integrates the IK calculations directly into the Marlin firmware. The printer accepts standard 5-axis G-code (with C and B axis values) and internally computes the required linear axis positions. The maker never sees the IK equations. If the LC or LB values change after recalibration, the firmware uses the updated values automatically, and the G-code doesn't need to be regenerated. This also means G-code files are portable: the same file works on any Rep5x printer regardless of its specific kinematic parameters.

The specifics of the Marlin fork, the G-code commands that were added, and how the IK routine evolved from on-the-fly transformation to pre-computed transformation are covered below.

Inverse kinematics

The inverse kinematics transform tool-tip coordinates (where the nozzle should be) to machine coordinates (where the motors need to move). For the head-head configuration, tilting and rotating the nozzle displaces the tool tip relative to the machine's reference frame. The firmware compensates for this displacement using two geometric parameters (Fig. 5.14):

- **LC**: the horizontal distance between the C-axis rotation centreline and the nozzle tip; visible only from above and sets the swing radius the nozzle traces as C rotates.
- **LB**: the vertical distance from the B-axis pivot down to the nozzle tip; sets the lever arm by which a B-tilt drops the nozzle and shifts it sideways.

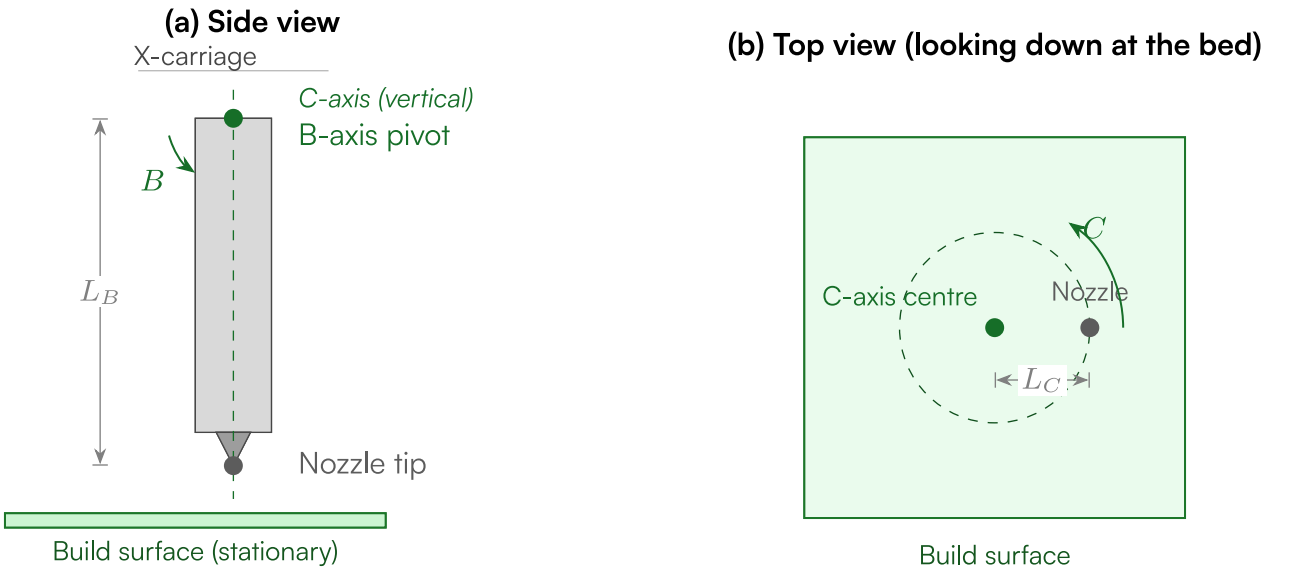


FIG. 5.14 Geometric parameters of the Rep5x inverse kinematics. (a) Side view showing L_B. (b) Top view showing L_C.

The transformation from tool-tip to machine coordinates is:

$$X' = X - \sin(C) \cdot LC + \cos(C) \cdot \sin(B) \cdot LB$$
$$Y' = Y + (\cos(C) - 1) \cdot LC + \sin(C) \cdot \sin(B) \cdot LB$$
$$Z' = Z + (\cos(B) - 1) \cdot LB$$

where (X, Y, Z) are the desired tool-tip coordinates, (C, B) are the rotation angles in radians, and (X', Y', Z') are the resulting machine coordinates sent to the stepper motors. These equations are implemented both in the Marlin firmware (for real-time computation) and in the shared JavaScript library `inverse-kinematics.js` (for use by the browser-based tools).

Marlin fork and custom G-code

The Rep5x firmware lives in a separate fork of Marlin (`dennisklappe/rep5x-marlin`). The starting point for the 5-axis integration was earlier work by DerAndere [60], a contributor in the Marlin community, who had already built a head-head penta-axis implementation with native kinematics and a few new G-code commands. His contribution deserves to be called out directly: the Rep5x firmware wouldn't exist in its current form without it. The `PENTA_AXIS_HH` kinematics mode, the `G43.4`/`G49` tool-length-offset commands (used to tell the firmware where the nozzle sits relative to the B-axis pivot), and the baseline structure for how an HH machine accepts 5-axis G-code were all his.

The Rep5x modifications build on that base in three places.

Rep5x geometry parameters. `LC` and `LB` were added as first-class configuration values. The IK routine reads them from EEPROM at runtime, so recalibrating the hardware doesn't require recompiling and reflashing firmware, just sending an EEPROM update from the Printer Setup tool.

Calibration correction via M667. A new M-code, `M667`, was added to accept the Fourier-fitted correction coefficients produced by the browser-based Calibrator. The firmware stores the coefficients in EEPROM and applies them as a final correction layer inside the IK routine. A builder who recalibrates sends the new coefficients once and every subsequent print uses the updated map. Without this, the correction data would have to be baked into every G-code file at generation time, which defeats the portability argument for firmware-based IK.

Pre-computed IK via M668. The bigger modification is `M668`, a new command that preprocesses an entire G-code file through the IK transform before the print starts. The original implementation performed the IK transform on the fly, one move at a time, as the G-code came in from the SD card. This works for simple geometry but starts to fail on dense 5-axis paths such as vase-mode spirals with continuous C-axis rotation: the planner can't keep up with the IK computation, starves, and the resulting motion stutters. The fix was to move the computation out of the real-time path. `M668` reads the source G-code file, passes every move through the IK transform plus `M667` correction, and writes the result to a new file on the SD card. When the print starts, the planner only sees already-transformed linear moves and can plan them at full speed (Fig. 5.15).

The firmware targets the Octopus V1.1 hardware described above, which provides at least the six stepper driver slots needed for the 5-axis configuration.

The browser-based Firmware Builder tool that wraps this configuration in a wizard interface is described alongside the rest of the toolkit below.

5.3 SOFTWARE TOOLS

Andersons' workflow uses Rhino/Grasshopper (commercial, €995 licence) to generate G-code for the specific demo geometries he tested, plus Python scripts run from the command line for IK and post-processing. Open5x's slicer also runs as a Grasshopper plugin on top of the same Rhino stack. As discussed in the literature review, this creates a fragmented toolchain that demands specialist software knowledge on top of the mechanical and elec-

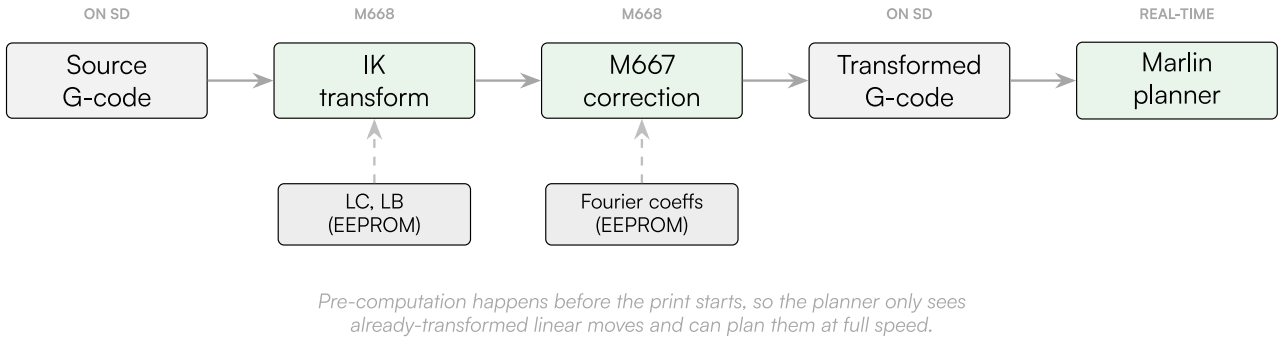


FIG. 5.15 The M668 pre-computation pipeline.

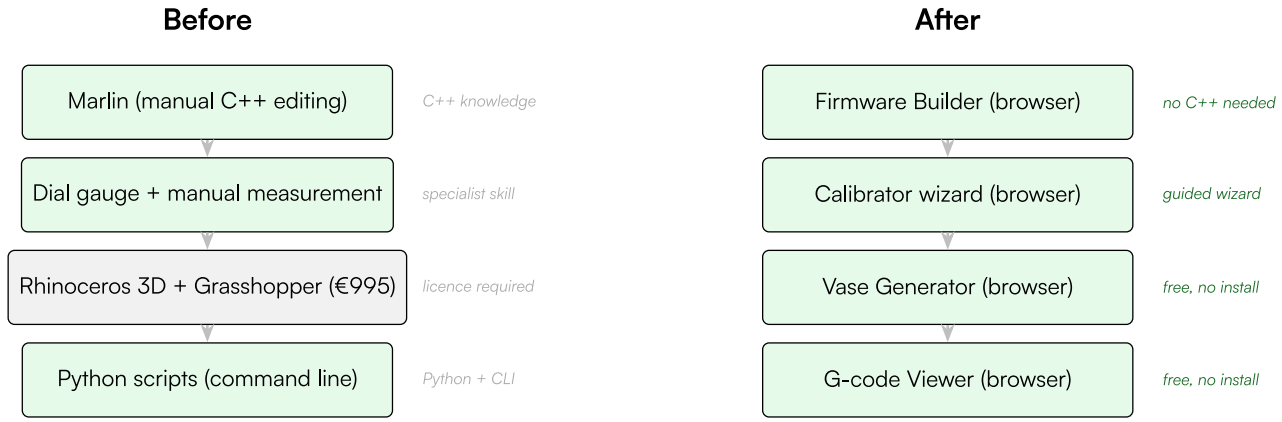


FIG. 5.16 Software toolchain before and after Rep5x.

tronics work the builder has already done. Rep5x doesn't try to replace the slicer itself; full 5-axis slicing of arbitrary models is still done with external tools. What it replaces is everything around the slicer: firmware flashing, calibration, printer control, G-code preview, and corrections, all moved to browser-based tools, plus a parametric Vase Generator for first-print demos. Fig. 5.16 compares the two stacks side by side.

The community survey backs part of this up and complicates another part. When respondents were asked to pick their top three concerns about attempting a multi-axis printer mod, specialist or expensive software was named by 28% of respondents, one of a cluster of toolchain concerns sitting behind the single biggest worry, the cost of parts (46%). That confirms the software barrier is perceived as real. But when respondents were asked where they'd want tools to run, the preference split was less convenient: 35% preferred desktop applications, against 28% for browser-based, with the rest split between "don't care" and command line. The resource side complicates it further. When asked what they'd need for a build like this, interactive browser-based setup and calibration tools came last of the options offered, named by 35% of respondents, well behind downloadable 3D models with recommended print settings (54%).

The design decision to go browser-based anyway, despite the desktop preference, was driven by the installation and onboarding barrier rather than stated preference. Desktop applications win once installed and configured, but the friction of installing PlatformIO, Python, or a CAD package keeps casual visitors from ever reaching the point of having an opinion on the UI. The target audience for Rep5x includes makers who haven't yet committed to building a 5-axis printer; the tools have to work for them on the first try. A browser tab that loads in two seconds reaches them; a desktop app that requires an installer, possibly a Python environment, and possibly a commercial licence doesn't.

Rep5x replaces the Rhino-plus-Python stack with a suite of browser-based tools that's grown to eight applications over the course of development. No installation, no licences, no command line. A builder opens a URL in Chrome or Edge and the tool is ready to use. The tools are hosted at `tools.rep5x.com` and also available offline from the GitHub repository. The choice of vanilla JavaScript (no frameworks, no build tools, no npm dependencies) is deliberate: it keeps the tools simple to maintain, easy for community contributors to understand, and free of dependency chains that could break over time.[4]

The rest of this section covers how the toolkit got to its current state: the iterative path from two standalone demos to an integrated studio, the UI and UX decisions that shaped it, and the individual tools themselves.

[4] JavaScript framework ecosystems are notorious for breaking changes. A tool built on React 17 in 2023 might not build without modifications in 2025. Vanilla JavaScript avoids this problem entirely.

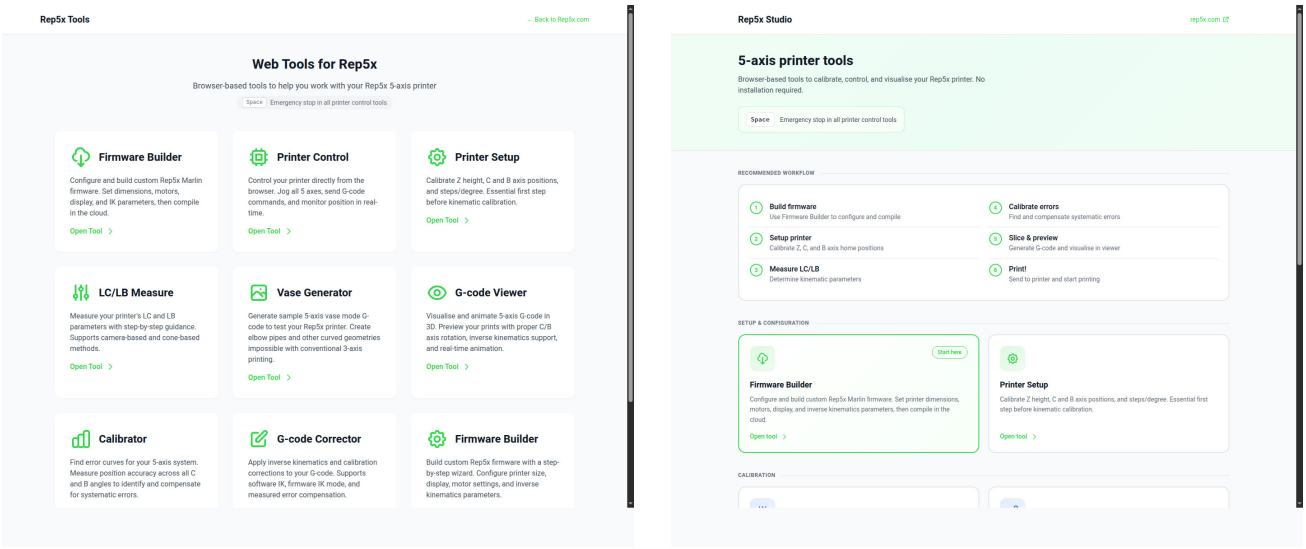


FIG. 5.17 Redesign of the tools landing page. Before (left): flat grid of eight tool cards with no workflow guidance. After (right): Rep5x Studio with a six-step workflow at the top and tools grouped by category.

Evolution of the toolkit

The toolkit is eight browser-based tools sharing a small set of JavaScript libraries, with a landing page (*Rep5x Studio*) that walks visitors through the build order. None of this was designed upfront. The hardware was already in place, so the software grew out of conversations with potential builders: someone hit a barrier, and a tool got written for it. Vase Generator and G-code Viewer for first prints, Printer Control for jogging, Printer Setup for first-time configuration, LC/LB Measurement and Calibrator for the kinematic and error sweeps. The set of tools is therefore the set of failure modes builders actually ran into, not a top-down catalogue.

The architecture and the landing page came from problems that surfaced once enough tools existed. Each tool was first written as a self-contained HTML page; by the fifth, the duplicated printer-connection flow and UI scaffolding had become a maintenance liability, and a refactor extracted them into shared libraries: a wizard framework for multi-step navigation, a connection-step base class used identically in four tools, a consolidated stylesheet, and utilities for toasts, drag-and-drop, and file downloads. The landing page was created the same way. The original flat grid of cards confused every early tester, who couldn't tell which tool to use when. The redesign, Rep5x Studio, puts a one-to-six workflow at the top (build firmware, set up printer, measure LC/LB, calibrate errors, slice & preview, print) with tools grouped by category below (Fig. 5.17).

That shared architecture is what allowed the later additions, EEPROM backup and restore in Printer Control, opt-in G-code sharing that lets builders contribute files to a research dataset, and arc support and tube rendering in the Viewer, to be small bits of new logic on top of an existing scaffold rather than fresh tools built from nothing.

UI and UX decisions

The way a tool feels to use matters as much as what it does, especially when the goal is to bring in people who wouldn't otherwise touch a multi-axis printer. A few design principles ran through the refactor and the Studio rebuild.

Wizards for first-time tasks. Four of the eight tools (Firmware Builder, Printer Setup, LC/LB Measure, Calibrator) use a step-by-step wizard layout (Fig. 5.18). The wizard pattern was chosen for tasks a builder only does once or twice, where the main problem isn't efficiency but not getting lost. A visible progress indicator at the top tells the user where they are, step-level validation prevents moving on with invalid state, and each screen is a single focused decision rather than a dense form.

Consistent visual language. On the Studio landing page the tools are grouped by category and colour-coded for navigation: green for setup and configuration, blue for calibration and measurement, purple for preview and printing, and orange for G-code generation. Inside the tools themselves the palette is the Rep5x green throughout, and the same typography, spacing, and card styling are reused everywhere, so once a builder knows one tool the rest feel familiar.

Tools-as-workflow, not tools-as-catalogue. The old landing page treated each tool as an equal item in a list. That's fine if the user already knows what they need. But the survey results and early Discord feedback made it clear that new builders didn't know. The numbered-workflow block at the top of Rep5x Studio was a direct response: by the time a visitor finishes reading the first screen, they have a mental model of the whole process.

Hiding complexity, not capability. The IK equations, Fourier fitting, Gaussian elimination, Marlin configuration parameters, all of it is still there under the hood. None of it is exposed in the UI. The Firmware Builder doesn't ask about C++ header flags, it asks about printer dimensions. The Calibrator doesn't ask for Fourier harmonics, it asks the user to click "next". Advanced users can still load JSON configurations or inspect the generated G-code, but by default the mathematical complexity stays out of the way.

Visual feedback at every step. Toast notifications for short messages, live position read-outs during jogging, real-time 3D previews in the Vase Generator, on-the-fly

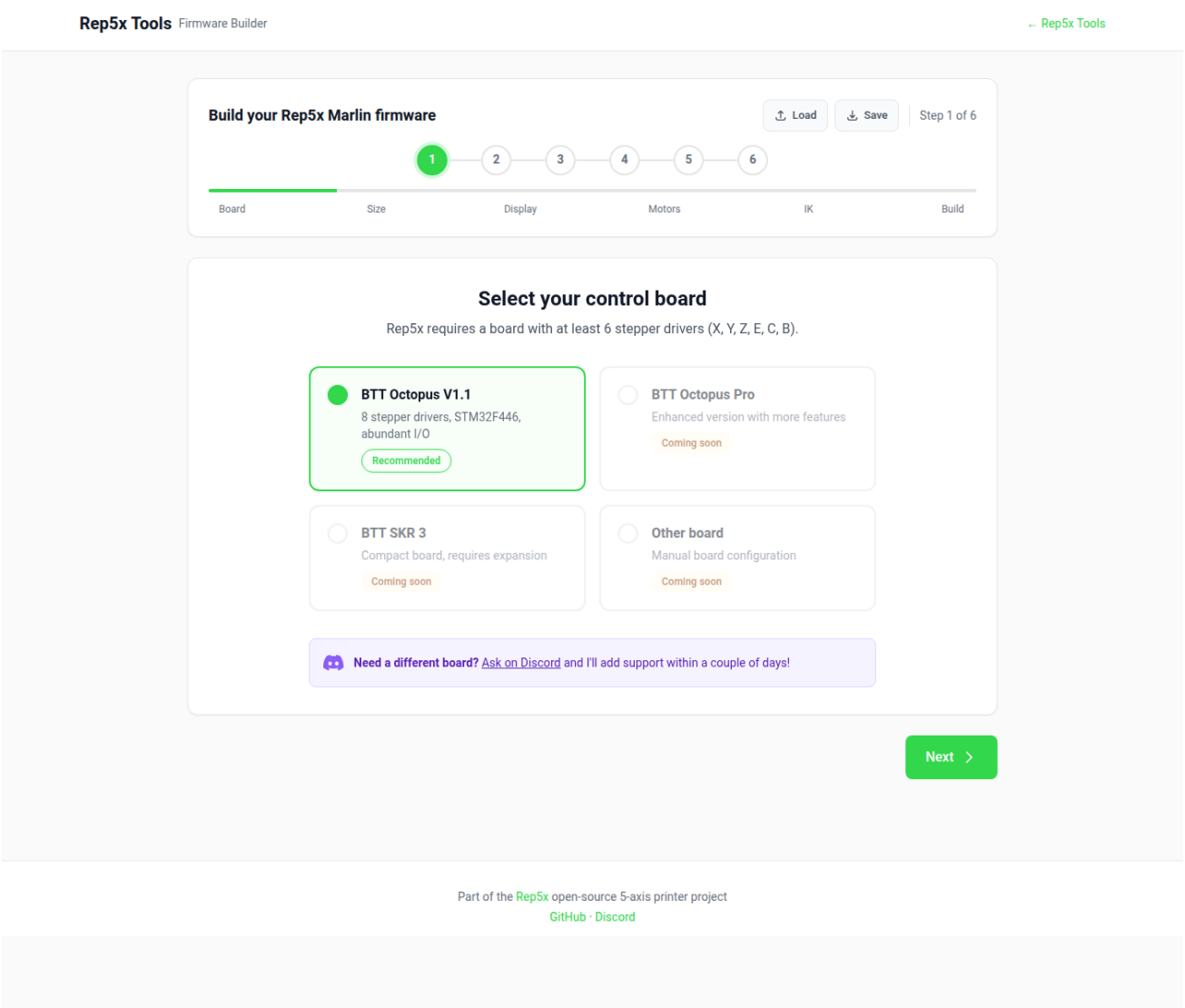


FIG. 5.18 Firmware Builder tool, step 1 of 6.

curve overlays during calibration. Builders who aren't sure what's supposed to happen can see the tool respond to every action, which shortens the debug loop when something goes wrong.

Keyboard controls throughout. Arrow keys for jogging, number keys for step size, spacebar as an emergency stop (M112) in every tool that talks to the printer. Voron and Klipper users are used to keyboard-first interfaces, and the survey showed a technically engaged audience. Full mouse-only operation still works, but keyboard users don't have to hunt for buttons.

Shared architecture

The tools are organised into three phases of use and share data through the browser's `localStorage` (Fig. 5.19).

Underneath, they build on a small set of shared JavaScript libraries:

- **Wizard Framework** (`wizard-framework.js`): a reusable multi-step wizard component that handles navigation, progress indicators, step validation, and state management. Used by the Firmware Builder, Printer Setup, LC/LB Measure, and Calibrator.
- **StepConnectBase**: a base class for the "connect to printer" step, used by the Printer Setup, Calibrator, and LC/LB Measure wizards. Handles serial connection, test-mode selection, and error recovery in one place.
- **Printer Interface** (`printer-interface.js`): a Web Serial API wrapper that manages serial communication. It provides command queuing, response parsing (posi-

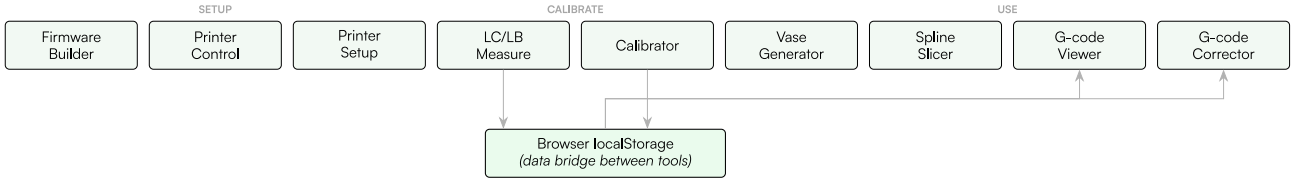


FIG. 5.19 Rep5x software tools and data flow, grouped into three phases of use.

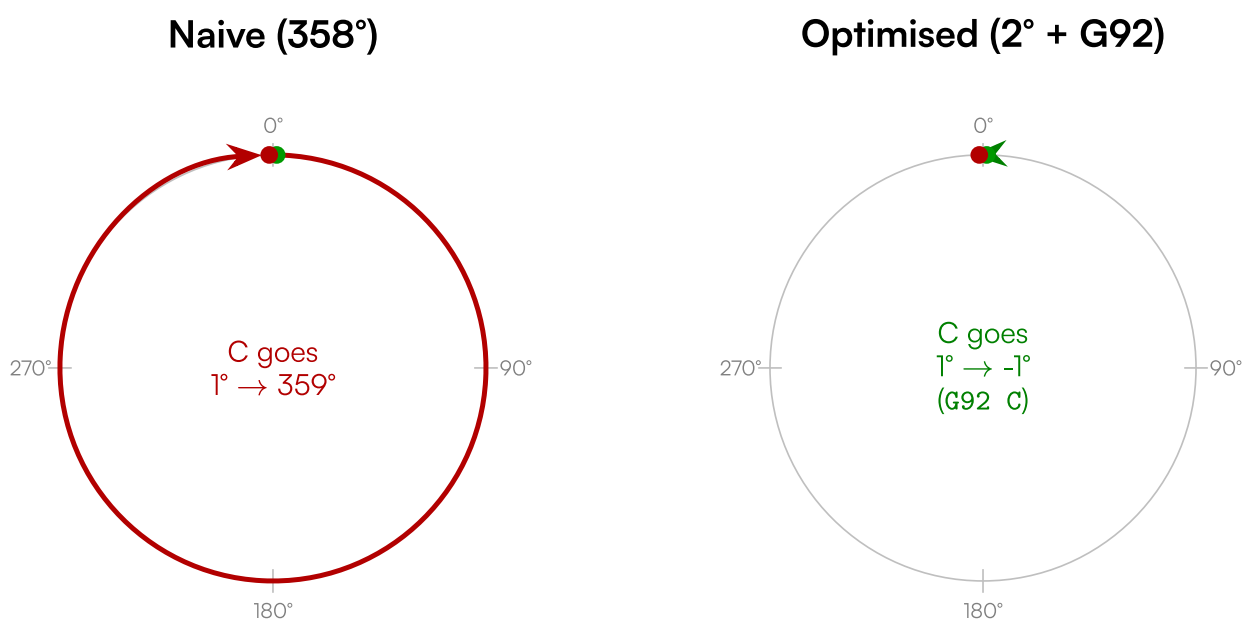


FIG. 5.20 C-axis shortest-path optimisation: naive long-way rotation (left) vs. wrap-aware short-path rotation (right).

tion, temperature, configuration queries), and a test mode for development without hardware.

- **Inverse Kinematics** (`inverse-kinematics.js`): the IK equations described above, implemented as both a class and standalone functions.
- **Calibration Corrector** (`calibration-corrector.js`): evaluation of the Fourier error maps, with the underlying curve fitting factored into a separate `fourier-fitter.js` module. Detailed in the Calibrator description below.
- **C-axis Optimiser** (`c-axis-optimizer.js`): takes the shortest rotational route between consecutive C-axis moves. When a move would naively rotate 358°, it inserts a `G92` reset and rotates 2° the other way instead (Fig. 5.20). Used by the Vase Generator and G-code Corrector.
- **Storage Manager**: persistent storage via the browser's `localStorage` API. Lets tools share data (LC/LB values, calibration results, tool-specific settings) without requiring file transfers between them.
- **Camera Manager**: wraps the browser's Media Devices API for the webcam-based measurement methods in LC/LB Measure and Calibrator.
- **Shared UI utilities**: toast notifications, drag-and-drop file handlers, file-download helpers, a standard header and footer, and a consolidated stylesheet.

This architecture is what made the later expansion possible. Adding a new tool starts from a working wizard shell, a working connection flow, and a consistent visual design. The new tool's code is mostly its own logic.

The tools

The current toolkit covers the full workflow from firmware compilation to printing.

Firmware Builder. Configuring Marlin normally means editing two C++ header files (`Configuration.h` and `Configuration_adv.h`) containing hundreds of parameters, then compiling and flashing. The first version of Rep5x's firmware documentation followed that standard approach: pre-built config files in the build-guide repo with an `INSTALLATION.md` walking through the install process.

Some builders edited the files without trouble; others stalled at "touching firmware" even when the change was one line. Questions came up on Discord, how to flash, how to adjust LC, how to change the B-axis limit, how to invert a motor direction, and writing a tool turned out to be easier than answering each one individually. Every build also has a slightly different setup (driver type, motor direction, B-axis travel, IK parameters), so the pre-built configuration is a good starting point but never the final one.

The Firmware Builder is a wizard that removes the C++-editing framing entirely. Six steps cover board selection, printer dimensions, display and neopixel options, motor configuration, kinematics parameters (LC, LB, B-axis travel, segments per second), and a final review. The generated configuration is compiled by a GitHub Actions workflow and the resulting binary is returned to the browser, which the builder flashes to the Octopus over USB. No local toolchain, no PlatformIO, no ARM compiler. Configurations can be saved as JSON and re-imported. The pre-built `Configuration.h` files are still shipped for advanced users who prefer to edit headers directly.

Printer Control. The Printer Control tool is a Pronterface-like [61] interface for direct interaction with the printer. It shows real-time axis positions (X, Y, Z, C, B) and temperatures (hot-end, bed), provides jog controls for all six axes with selectable step sizes (0.1/1/10/100 mm for the linear axes, 1/5/15/45/90° for the rotary), and includes a G-code console with command history.

Printer Setup. The Printer Setup tool is a seven-step wizard that establishes the printer's baseline after the hardware is assembled:

1. Connect to the printer via serial.

2. Home all axes and verify positions.
3. Verify axis movement directions (X, Y, Z).
4. Calibrate the C-axis: set the home position, verify 360° rotation.
5. Calibrate the B-axis: verify tilt range, establish zero.
6. Set the Z-axis offset (nozzle height above the bed).
7. Save all values to the printer's EEPROM.

The tool writes Marlin configuration commands (M206 for home offsets, M92 for steps per unit) directly to the EEPROM, so values persist across power cycles.

LC/LB Measurement. The LC/LB Measurement tool determines the two geometric parameters that define the 5-axis kinematics. Two measurement methods are supported. *Camera method*: uses a camera with a crosshair overlay for visual alignment. The nozzle position is manually tracked across rotation angles. *Cone method*: uses a 3D-printed reference cone aligned with the nozzle tip. The builder manually jogs the axes and the tool records the displacement.

Both methods follow a six-step wizard: method selection, connection, preparation, LC measurement (rotating the C-axis and measuring displacement), LB measurement (tilting the B-axis and measuring offset), and results. The measured values are saved to the printer's EEPROM via `M500` so the firmware uses them on the next print, and also cached in the browser's `localStorage` so the other tools can read them without prompting again.

Calibrator. The Rep5x Calibrator turns the calibration step into a guided wizard. Even after accurate LC/LB measurement, residual positioning errors remain due to mechanical imperfections like eccentricity and axis misalignment. These errors vary with rotation angle and follow periodic patterns. The Calibrator maps them across both axes using a two-sweep measurement process (Fig. 5.21).

C-sweep: the nozzle position error (X, Y, Z) is measured at 8 angular positions around the C-axis (0°, 45°, 90°, ..., 315°) with B held at 0°.

B-sweep: the error is measured at 13 angular positions of the B-axis (0°, ±15°, ±30°, ..., ±90°) with C held at 0°.

The measured error data is fitted with Fourier series using least-squares regression (an example fit is shown in Fig. 5.22). The C-sweep data, being periodic (0—360°), is fitted with a 3-harmonic Fourier series. The B-sweep data, which is non-periodic (±90°), is fitted with a 2-harmonic trigonometric series. The fitting runs in the browser using Gaussian elimination with partial pivoting, implemented in the `fourier-fitter.js` shared library. The fitted coefficients are then consumed by `calibration-corrector.js` to apply corrections at runtime in the G-code Viewer, the Corrector, and the firmware itself.

The fitted error model produces smooth correction functions that can be evaluated at any (C, B) angle pair. The total correction is the C-sweep correction plus the B-sweep correction, minus the value of either fitted curve at the reference position (C=0, B=0). Both sweeps pass

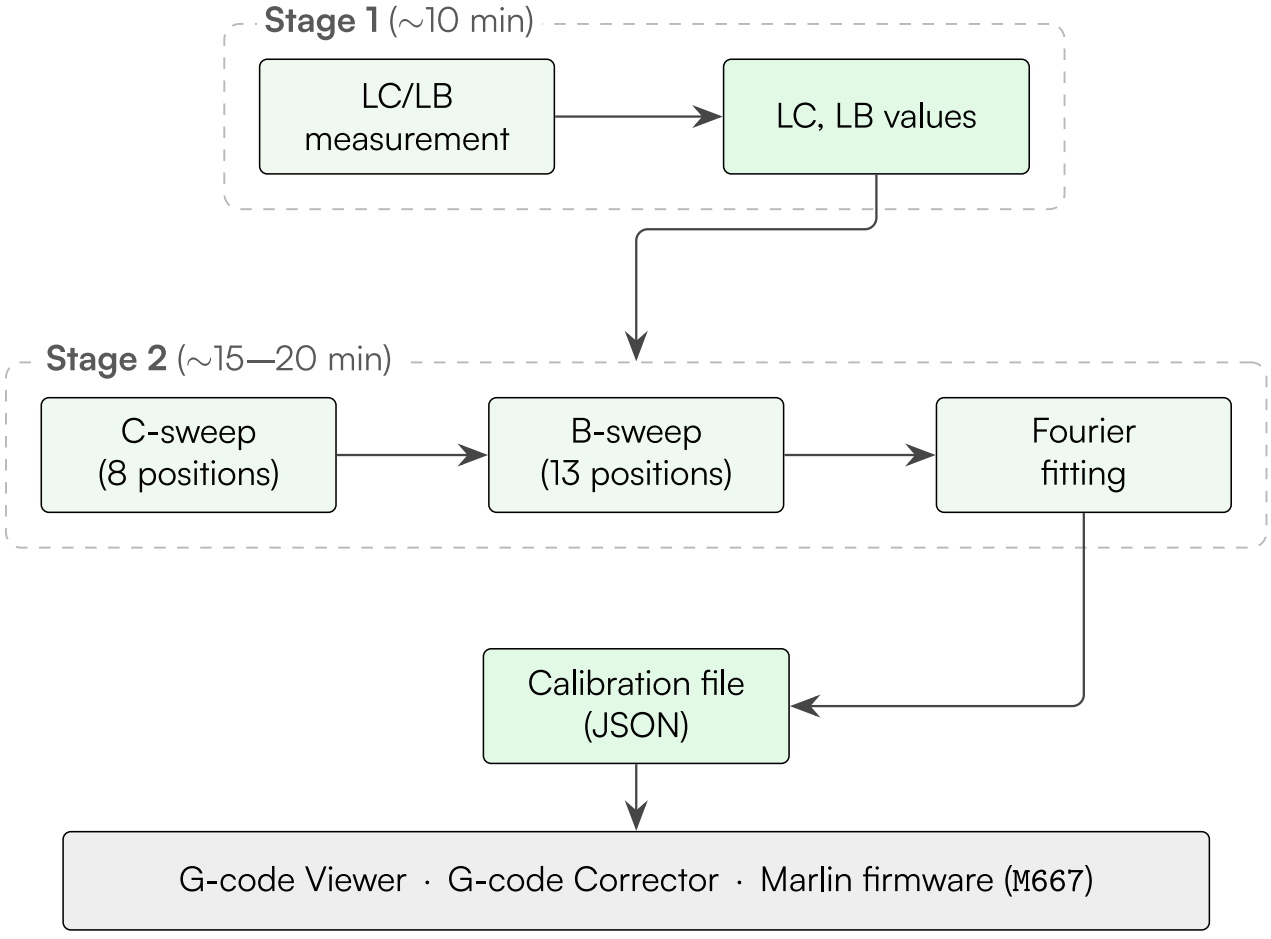


FIG. 5.21 Two-stage calibration workflow.

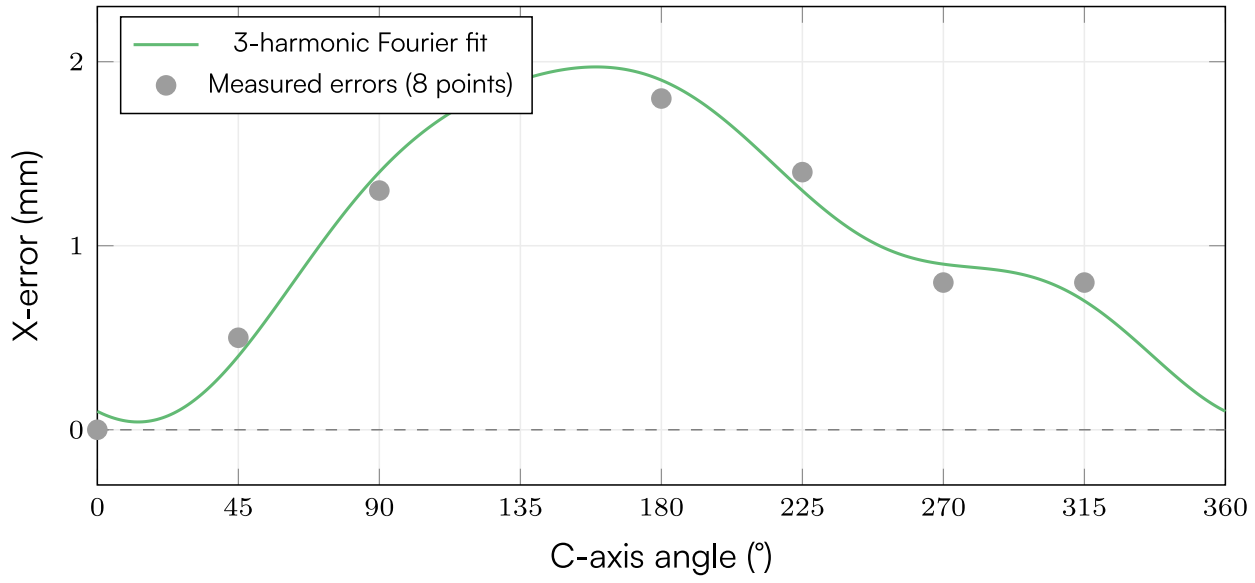


FIG. 5.22 C-sweep calibration data with 3-harmonic Fourier fit.

through that reference point (the C-sweep is measured with B=0, the B-sweep with C=0), so adding them would count the reference-position error twice; subtracting it once cancels the duplication. This additive model is a simplification, but the 21-point measurement sweep is practical for non-expert users, and the residual errors after correction are small enough for the target application.

From the builder's perspective, the process is straightforward. The wizard connects to the printer, moves the nozzle to a reference position, then steps through each measurement angle automatically. At each position the builder aligns a camera crosshair with the nozzle tip (or confirms alignment manually with the cone method) and clicks "next". The tool handles the kinematics, the error calculation, and the curve fitting. The whole sweep takes 15—20 minutes, after which the builder sees a graph of the raw measurements overlaid with the fitted correction curves. If the fit looks reasonable, the calibration is done. The fitted coefficients are sent to the printer via `M667` (stored in EEPROM, applied inside the IK routine), cached in the browser's `localStorage` for the other tools, and downloadable as a JSON file for backup or sharing.

G-code Viewer. A Three.js-based 3D visualiser that renders multi-axis toolpaths. The viewer parses G-code (G0/G1 linear moves, G2/G3 arcs) with X, Y, Z, C, B coordinates, applies the inverse kinematics transformation to show actual machine positions, and optionally applies calibration corrections so the builder can see the effect of compensation before printing. A simple collision check flags any move where the tilted head would crash into the bed or the part being printed, so the problem shows up in the preview rather than mid-print. The head is modelled as three primitives (a nozzle cylinder, a heatblock box, and a cable cylinder) anchored at the nozzle tip; at each toolpath step those primitives are tested against the bed plane and against the points already deposited, and any collision is highlighted in the 3D view.

Vase Generator. A parametric generator for 5-axis vase-mode G-code. Three shape presets are included: an elbow pipe, a mushroom, and a spiral flower. The generated G-code is passed through `c-axis-optimizer.js`, which wraps C-axis positions to the 0—360° range and inserts `G92` resets whenever consecutive moves would cross the wrap boundary, so the printer always takes the shorter rotational route. A Three.js preview shows the generated shape and toolpath in real time as parameters are adjusted.

G-code Corrector. The G-code Corrector applies calibration corrections, and optionally the inverse kinematics, to an existing G-code file, so prints generated outside the Rep5x toolchain can still be calibrated. It loads the Fourier error model exported by the Calibrator and rewrites every move with the correction applied. It runs in three modes: for raw tool-tip G-code it applies the IK transform (from the entered LC and LB) and then the correction; for a file already transformed by the firmware's IK it detects this automatically and applies the correction alone, in the matching coordinate mode; and for a file that only needs IK it adds the transform without any correction. A Three.js preview overlays the uncalibrated and calibrated nozzle paths so the effect is visible before printing. Because the Corrector shares `calibration-corrector.js` and the IK library with the rest of the toolkit, a correction comes out identical whether it's applied here, in the Viewer, or in the firmware. Since the inverse kinematics and the calibration correction now run in the firmware itself (`M668` and `M667`), the Corrector is no longer part of the standard workflow. It is kept as a fallback for applying calibration to G-code from external slicers that hasn't passed through the firmware path, and is not under active development.

Firmware Validator. The requirements chapter noted that Andersons' setup gave a builder no way to check the firmware was working once it was flashed. The Firmware Validator is that check. It was the last tool built, a ninth alongside the eight released tools, and is still in testing,

not yet released at the time of writing. It came from builder feedback: until then, verifying a flash meant working through every endstop, motor direction, and thermistor by hand, which is tedious and easy to get wrong, since a single forgotten check tends to surface as a failed print later. It connects to the printer and steps through five stages: diagnostics (`M115` for firmware identity, `M122` for the TMC drivers over UART, `M119` for endstop states, and `M105` to confirm the thermistors read room temperature), a manual endstop walk-through where the builder triggers each switch in turn while the tool polls `M119`, a stepper-direction test that jogs each of the five axes and asks whether it moved the expected way, a homing-direction check, and an extruder test with optional E-steps calibration. A motor or endstop wired backwards is the most common first-flash mistake, and a six-axis build offers more ways to get it wrong than a stock printer does. So the validator does more than report: if the builder uploaded their Firmware Builder configuration at the start, it writes out a corrected JSON with the fixes applied, a reversed motor flips the relevant invert flag, a reversed homing direction flips the home-direction sign, and a measured E-steps value is written back. Re-importing that file into the Firmware Builder and reflashing turns a debugging session into a couple of clicks. In the Studio workflow the validator sits between the Firmware Builder and Printer Setup.

The rep5x.com website

The tools live at `tools.rep5x.com`. The project's public face lives at `rep5x.com`, a separate site built with Astro 5 and deployed on Cloudflare Pages. Its job is different: where the tools are functional, the website is informational. It's the first thing a new visitor sees, and it has to make the project make sense in the first thirty seconds (Fig. 5.23).

The key architectural decision is that the website doesn't have its own copy of the documentation. Every piece of build-guide content, the universal BOM, the universal assembly instructions, the printer-specific BOMs, the printer-specific assembly steps, lives as Markdown in the `build-guide/` directory of the GitHub repository. At build time, the Astro site reads those Markdown files directly and processes them into web pages. When a contributor opens a pull request that fixes a typo in the Ender 3 Pro BOM, the same edit goes live on the website as soon as the PR is merged and the Cloudflare build runs. There's no second copy of the documentation to maintain, and no way for the website to drift out of sync with the canonical source. The same pattern extends to adding a new printer: a contributor just adds a new directory under `build-guide/printer-specific/`, and the next site build picks it up automatically, no website change needed.

The build guide page turns this directory structure into something a non-GitHub user can actually navigate. A printer selector at the top of the page lets the visitor pick their machine from the supported list (Ender 5 Pro, Ender 3 V3 SE, Ender 3 Pro, or "Universal only"). Selecting a printer swaps in that printer's BOM and assembly instructions alongside the universal ones, and the selection is remembered in `localStorage`, so the next visit opens on the right printer. The underlying data comes straight from the printer-specific Markdown files in GitHub; the website just presents it in a way that doesn't require knowing which folder to open.

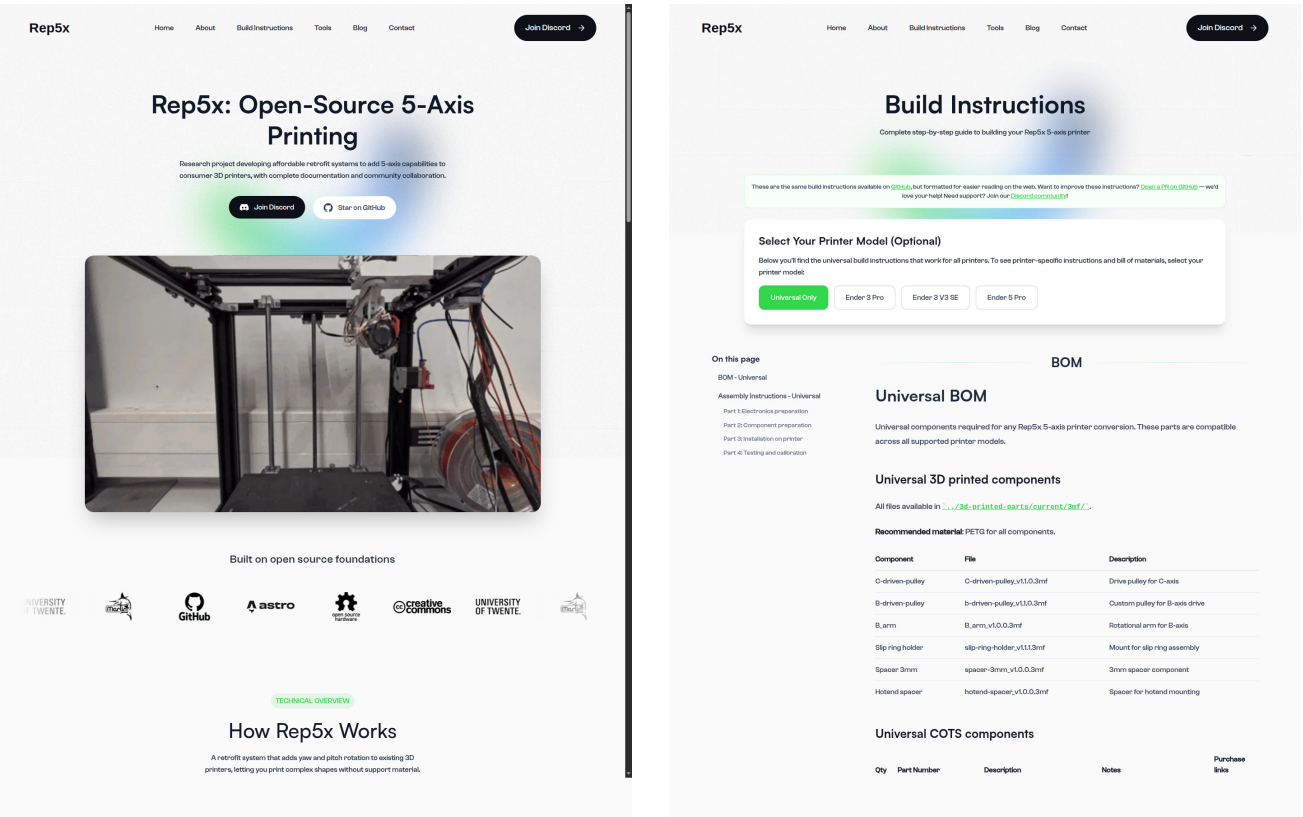


FIG. 5.23 The rep5x.com website. Left: project homepage. Right: build instructions page with printer selector and BOM.

The page adds one more piece of state that the raw Markdown doesn't have: check-off. Every BOM row is clickable, and clicking a row strikes it through and greys it out. The state is stored in the browser's `localStorage`, keyed by section and row index, so the checklist persists across visits on the same device. A builder working through the BOM sees at a glance what they've ordered and what they haven't, without having to copy the table into a spreadsheet. The Markdown source is untouched; the website layer on top of it turns a static document into a working checklist for people who aren't used to reading GitHub directly.

5.4 OPEN-SOURCE RELEASE AND COMMUNITY

The Rep5x project is released under GPLv3. The main GitHub repository, `github.com/dennisklappe/rep5x`, holds the hardware design files (STEP and 3MF), the BOMs and supplier links, the build guide as Markdown, all eight browser-based tools, and the source of the rep5x.com website; the modified Marlin fork lives in a companion repository, `rep5x-marlin`. Issues and pull requests are open. There's no separate "commercial" track or paid version.

The hardware documentation is also being consolidated into a standalone open-hardware paper prepared for submission to HardwareX, carrying the complete bill of materials, wiring, assembly, and calibration procedure in a citable form, complementing the living documentation in the repository.

GPLv3 was chosen for the same reason RepRap and Voron use it: anything derived from a Rep5x part inherits the licence, so improvements stay open. Permissive licences (MIT, BSD) would let downstream forks close up; GPL keeps the project's contributions returning to the same pool.

Discord community

The Rep5x Discord has around 200 members at the time of writing. It's the project's main support channel and its main feedback channel, both at once. Builders ask questions when something doesn't work the way the build guide describes; the answers feed back into the build guide and, often enough, into a tool change.

Most of the questions cluster in the same areas: firmware (how to flash, which driver type, how to invert a motor), calibration (what numbers should I see, why is the curve ugly, how do I redo this), and adapting the design to printers other than the three supported ones. The pattern is consistent enough that a question asked once is likely to be asked again, which is what makes Discord answers worth turning into tools or guide additions.

Iteration from community feedback

The system kept changing in response to what builders actually ran into. Four cases stand out.

The Firmware Builder. The first version of the firmware documentation shipped pre-built `Configuration.h` files and an `INSTALLATION.md`. Some builders edited the files without trouble; others stalled at "touching firmware" even when the change was one line. Questions on Discord about flashing, LC adjustment, B-axis limits, and motor direction kept coming back, and writing a wizard that removes the C++ framing turned out to be easier than answering each one individually. The Firmware Builder exists for that reason.

Rep5x Studio. The original tools landing page was a flat grid of cards in no particular order. Early testers all reported the same problem: they didn't know which tool to use when. The Studio redesign puts a recommended one-to-six workflow at the top of the page, so the order of use is the first thing a visitor sees. The redesign came directly from tester confusion, not from a planned UX overhaul.

The C-axis re-labelling. Andersons labelled the yaw axis A and the tilt axis B. Builders coming from multi-axis CNC pointed out that this clashed with the LinuxCNC and Siemens convention, where the continuous rotation around Z is the C axis. Re-labelling the yaw axis from A to C made the generated G-code match what other multi-axis CAM users expect, at the cost of diverging from Andersons' notation. The change was a direct response to that feedback.

The Firmware Validator. Builders flashing the firmware for the first time were checking every motor direction, endstop, and thermistor by hand, and missing steps. The Firmware Validator (described above) replaces that with a single sequence of checks and writes back a corrected configuration. It was the most recent of these changes.

None of these were predicted at the start; they came up because the system was being used by people who weren't the designer.

5.5 ADOPTION IMPACT

The design decisions in this chapter were driven by the adoption barriers identified in the analysis chapter. Mapping the implemented system back to Rogers' five attributes [19] shows how each component contributes:

- **Relative advantage** was already high for multi-axis printing (better surface quality, less waste, controlled anisotropy). The system preserves this advantage while making it accessible. The Vase Generator provides immediate proof: after calibration, the builder can produce a 5-axis print within minutes, demonstrating the advantage firsthand.
- **Compatibility** is addressed by the modular hardware (works with existing printers, doesn't modify the print head), the Marlin firmware (builds on the most widely used FDM firmware), and the browser-based tools (works on any OS with Chrome).
- **Complexity** is reduced by the guided wizards (replacing command-line scripts and manual measurement with step-by-step processes), the firmware builder (replacing C++ header editing with a web form), and the firmware-based IK (hiding the kinematic equations from the user entirely).

- **Trialability** is improved by the low cost (under €200), the self-replicating parts (printable on the builder's own machine), and the Vase Generator (which produces a working 5-axis print immediately after calibration, providing quick feedback that the system works).
- **Observability** is addressed by the project website, the 3D G-code viewer (which makes 5-axis toolpaths visually understandable), and the community infrastructure described above.

5.6 DESIGN TRADE-OFFS

Several design decisions involved explicit trade-offs between capability and accessibility.

Browser-based tools vs desktop applications. Browser-based tools have real limitations: the Web Serial API only works in Chromium-based browsers, localStorage is limited to about 5 MB, and JavaScript isn't the fastest language for computation. But the accessibility advantage, no installation, no dependencies, works on any OS, outweighs these limitations for the target audience. The computational demands of IK and Fourier fitting are well within what a browser can handle.

Firmware-based IK vs pre-processing. Implementing IK in the firmware adds complexity to the Marlin codebase and requires a custom firmware build. Pre-processing (Andersons' approach) keeps the firmware standard. But firmware-based IK means the G-code is portable: the same file works on any Rep5x printer regardless of its specific LC/LB values. This decoupling is important for sharing G-code within the community.

Cloud firmware compilation vs local toolchain. The firmware builder sends the configuration to a cloud service for compilation rather than building locally in the browser. This requires internet access, which is a dependency. But it eliminates the need for the builder to install PlatformIO, Python, and the ARM compilation toolchain, which would be a significant barrier for non-developers.

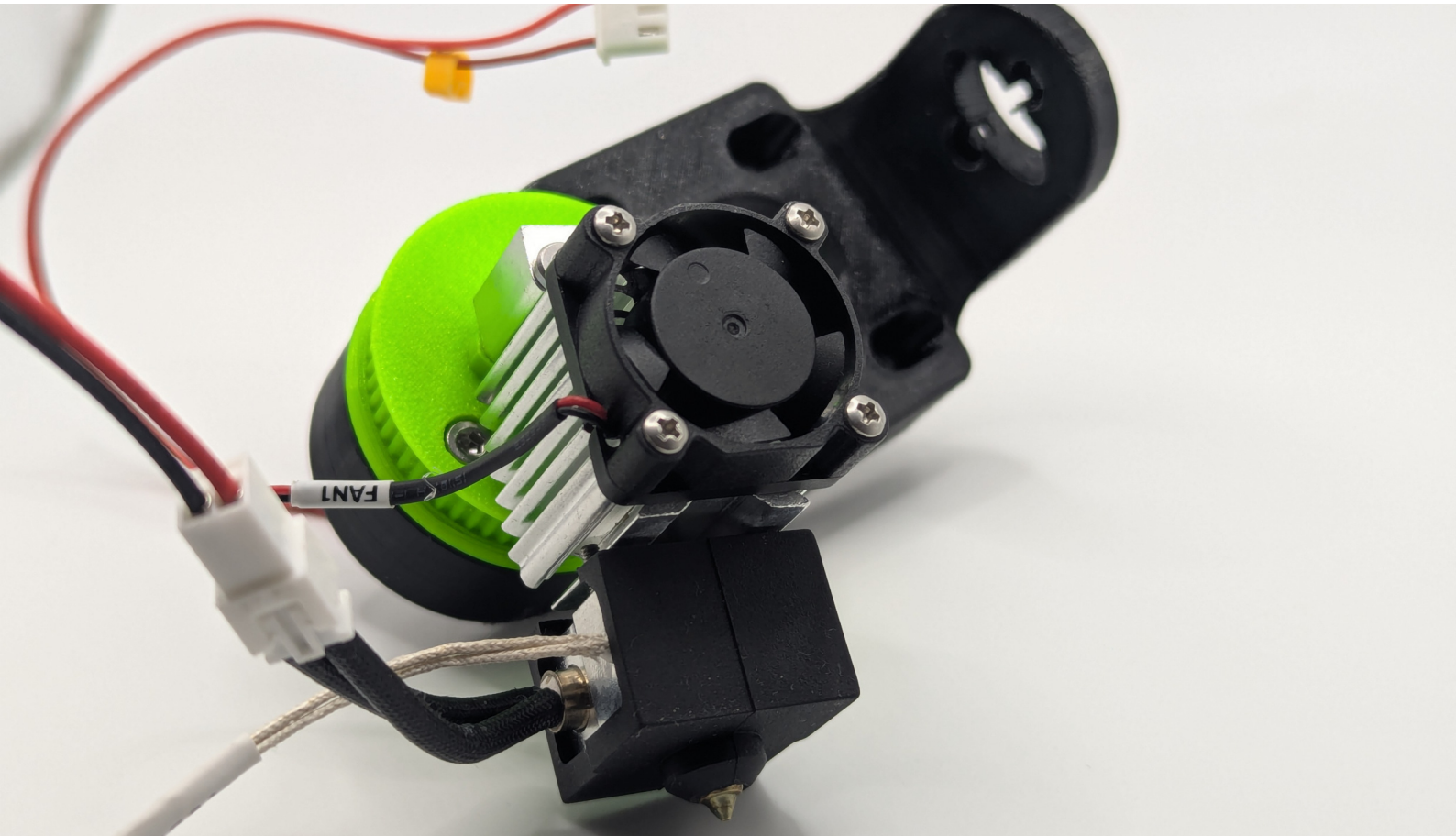
Each of these trade-offs follows the same principle: where capability and accessibility conflict, accessibility wins. The system doesn't need to be the most capable 5-axis printer. It needs to be the most buildable one.

CHAPTER 06

06

EVALUATION

Evaluating the system: a working build and its calibration, a pre-build survey, the Discord feedback loop, the adoption funnel, and the first independent build attempts.



The design and development chapter described the Rep5x system and how it's released. This chapter looks at whether the system works and whether it gets used. The evidence comes in five forms: a functional check of the build and its calibration, a pre-build interest survey, the Discord activity and the iteration it produced, the adoption funnel logged in the project CRM, and the outcomes of independent build attempts.

6.1 FUNCTION AND CALIBRATION

The system itself works. The project's reference build, an Ender 3 V3 SE put together from parts salvaged off two retired printers plus a few AliExpress-ordered components like the controller board and the slip ring, runs the Rep5x Marlin firmware, homes all five axes, and prints. The build guide on rep5x.com is based on it, and the vase-generator shapes were made to exercise the C and B axes on it. The same add-on has also been fitted to two other base printers, Janis' Ender 5 Pro and another student's Elegoo Neptune 4 Max, which shows the mechanism adapts across platforms. Neither is fully there yet: on the Ender 5 Pro the Rep5x firmware ran but with some issues still to sort out, and the Elegoo build is still in progress.

Calibration brings the geometric error into range. Before correction, the nozzle's positional error across the rotation range is large: on the reference build the deviation grows with tilt to almost 5 mm at B = 90° and averages 2.85 mm RMS over the full B and C range, the worst component being the Y direction at up to 4.9 mm. That's far more than any print could tolerate. But the error is systematic: smooth and monotonic in the B tilt, sinusoidal in the C rotation. That's exactly what a Fourier model is good at. Because the error repeats with rotation, a few harmonics stay smooth across the 0°—360° wrap, where a polynomial would diverge at the ends and joining the measured points would just chase noise. X and Y are fit this way, three harmonics across the C rotation and two across the B tilt (Fig. 6.22); the Z offset, which only appears once the B axis tilts, is calibrated separately rather than with a Fourier series. Applied in real time in the firmware, the correction collapses the modelled residual to about 0.15 mm RMS, 0.4 mm worst case (Fig. 6.1). That 0.15 mm is a best case, though. It's the model's own fit at the calibration poses, not an independent re-measurement, and the error isn't perfectly repeatable from one calibration run to the next, so in practice a calibrated machine still drifts a little more than the figure shows. The correction clearly works in principle; pinning down the real-world residual across runs is left for future work.

6.2 PRE-BUILD INTEREST SURVEY

A community survey collected 89 responses from active 3D-printer makers (the full instrument and the complete results are in Appendix D). The sample skews experienced and hands-on: 84% had already heard of multi-axis FDM, a third had built a printer from a kit or scratch, and 42% had edited firmware configuration files. Interest in the idea was high, 60% rated it 4 or 5 out of 5, and the motivations they cited mirror the advantages the literature attributes to multi-axis FDM, with support reduction and otherwise-impossible overhangs leading (Fig. 6.3). That's exactly what a self-selected sample would say, so the value is in the barriers and preferences, not the enthusiasm.

Cost is the single biggest concern, but the toolchain is the bigger story. Asked to name their top three concerns about attempting a multi-axis mod, the clearest single answer was the cost of parts and electronics (46%). After that came a tight cluster of non-hardware barriers, none dominant on its own: calibrating the extra axes (29%), specialist or expensive software (28%), too little community or support (26%), and missing documentation (25%). Taken together, at least one of these four toolchain-and-knowledge barriers was named by **70% of respondents** (Fig. 6.2), far more than any physical or mechanical worry. The mechanism isn't what makes people hesitate. Everything around it is.

The free-text answers point the same way, and more sharply. Unprompted, several respondents named slicing as the real wall: *"a good non-planar slicer... that's the main*

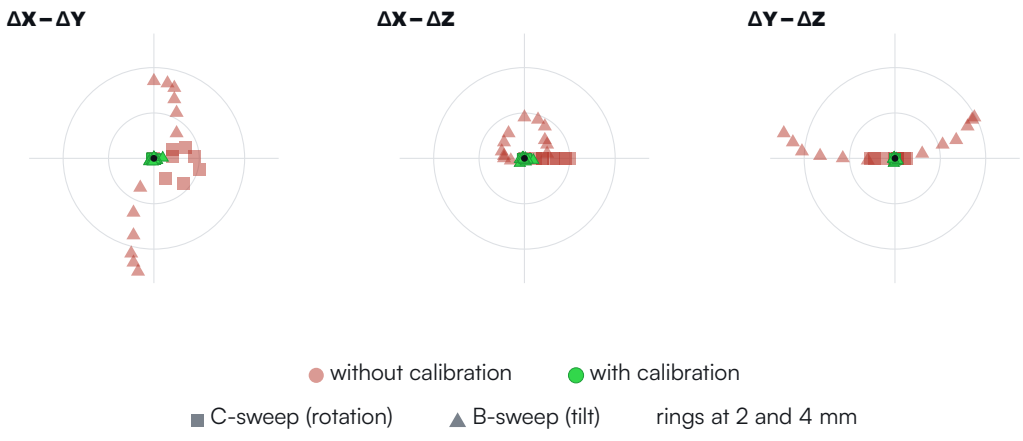


FIG. 6.1 Nozzle drift around the target point, projected onto the three axis planes. Each point is one calibration pose, pooling the C-sweep (B = 0°, C stepped around) and the B-sweep (C = 0°, B tilted). Without calibration the position deviates by up to 5 mm (2.85 mm RMS); with the Fourier correction applied the modelled residual collapses to the centre (0.15 mm RMS, 0.4 mm worst case).

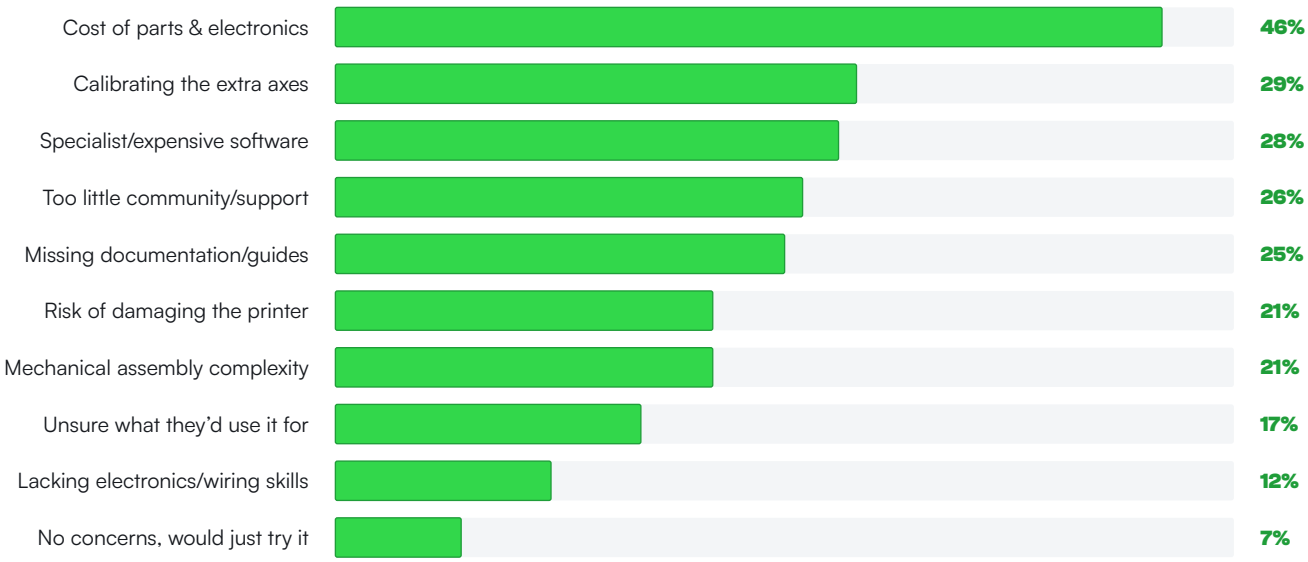


FIG. 6.2 Biggest concerns about attempting a multi-axis mod (top three per respondent, N = 89). Cost leads, but the cluster of software, calibration, documentation and community, the toolchain rather than the mechanism, is named far more often in total.

problem"; *"the biggest hurdle is slicing"*; *"custom slicer development"*. One experienced maker wrote that what would make the difference is *"(excellent) slicing software and at least 'good' firmware."* This is the software gate the literature review predicted, stated by makers in their own words.

The audience says it prefers desktop software, and didn't ask for browser tools. Two of the results run against Rep5x's central design choice. Asked how they'd want setup, calibration and G-code tools delivered, desktop applications were the plurality preference (35%), ahead of the browser (28%), with another 27% saying they didn't care as long as it worked. And when asked what resources a build like this would need, interactive browser-based tools came **last** of seven options (35%), well behind a written guide (70%) and downloadable models (54%). The browser-based decision was made anyway, on a trialability argument: the survey reaches makers already committed enough to install and configure software, so their stated preference understates how much an install step deters someone still deciding whether to attempt the build at all. Removing it lowers the barrier to that first attempt, which is where adoption is won or lost.

The sample is small (N = 89) and self-selected from people interested enough to find and fill in the form, so it can't speak to the wider population of makers who've never heard of multi-axis FDM. They hint at a direction, but don't prove it. What they do is corroborate the framing the rest of the thesis is built on: cost matters, but the binding constraint is the toolchain and the knowledge around it.

6.3 DISCORD AND THE ITERATION LOOP

The Rep5x Discord has around 200 members at the time of writing. It's the primary feedback channel and the source of most design changes since v1.0.0.

The design and development chapter described four cases where community feedback drove specific changes to the system: the Firmware Builder, the Rep5x Studio redesign, the C-axis re-labelling, and the Firmware Validator. Each is direct evidence that the design reached changes that wouldn't have come from the designer alone. The pattern is consistent: a builder hits a problem, the problem surfaces in Discord or testing, and the system changes in the next iteration.

What the iteration loop doesn't measure is whether the changes worked. Whether the Firmware Builder got more builders to flash firmware successfully, whether Studio's workflow header reduced confusion, whether the Firmware Validator caught wiring mistakes that would

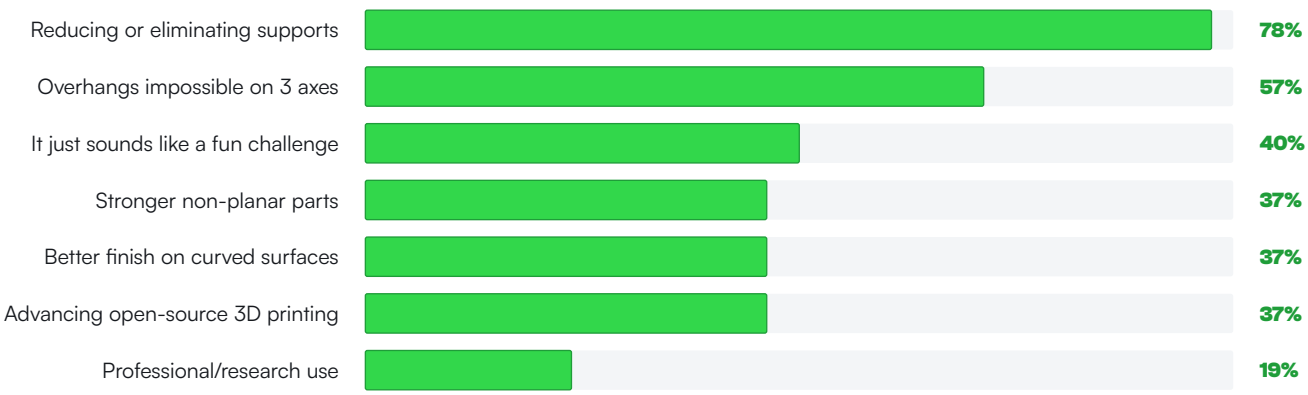


FIG. 6.3 What would motivate makers to try multi-axis printing (N = 89). Support reduction and otherwise-impossible overhangs dominate, the same advantages the literature attributes to the technology.

FIG. 6.4 The adoption funnel from the project CRM (220 contacts), drawn as a flow. 155 joined the community, but most went no further; 15 worked towards a build (9 at planning, 6 in development or active building).

otherwise have surfaced as failed prints, those are downstream effects, measurable only after several full builds. The pattern of feedback-driven change is documented; the effect on the adoption rate isn't.

6.4 THE ADOPTION FUNNEL

The project kept a running record of everyone who engaged with it. Adoption-relevant conversations on Reddit, Discord, and GitHub were logged in a self-hosted Twenty CRM [52], one record per person and a dated note per interaction, with each contact classified by build stage, build likelihood, engagement, and skill (the instrument is described in the methodology). Read from top to bottom, the build-stage field is an adoption funnel: from people who only researched the idea, through those planning and actively developing a build, down to the few who reached a completed one.

At the time of writing the CRM holds 220 tracked contacts, most reached through Discord (160 of 220). The funnel in Fig. 6.4 shows how far they got. 155 joined the Discord, a healthy community by the standards of a niche project, but the step that matters here is the next one: only 15 moved past lurking towards an actual build, six reached development or active building, and none completed an independent Rep5x build. Interest and community aren't the bottleneck; turning either into a finished build is.

The classification carries the same message. Build likelihood, the running estimate of how likely each contact is to follow through, is low for 196 of the 220, with only around twenty rated medium or higher. The base printers people mention are almost all from the Ender 3 and Ender 5 families, the platforms Rep5x targets, so the audience is the right one. It just doesn't convert.

Two things are worth keeping in mind. The CRM is a hand-maintained record, so the stage and likelihood ratings are a single classifier's judgement rather than an independent measure, and some fields are only sparsely filled. And it's a living record, so these figures are a snapshot at the time of writing. But the broad shape, a wide top and almost nothing reaching the bottom, isn't a marginal effect, and it's the same story the individual build attempts tell in detail.

6.5 INDEPENDENT BUILD ATTEMPTS

The harder question is whether someone else can reproduce it, and there the evidence is still early. The build-phase effort went into the toolkit rather than into recruiting builders, so independent attempts are few and self-selected: some were approached directly, others found the repository or the website on their own and documented their progress in the Discord build logs. None reached a first print within the assignment window, but with numbers this small the useful question isn't how many finished, it's where the ones who stalled got stuck.

The independent builds stall at the toolchain, not the mechanism. The mechanical side rarely blocked anyone: the builders who got that far printed the parts, assembled them, and wired the motors without much trouble. Where progress stopped was firmware and configuration. One reached a fully assembled, wired, homing machine with heaters and fans working, then stalled on the C and B axis motion setup. Another never reached assembly, held up first by parts trickling in from overseas and then by time.

More telling is how builders routed around the firmware rather than through it. One abandoned Marlin entirely for a Duet board and RepRap firmware, explaining that without a software background the compiling and code-hunting Marlin demanded had driven an earlier attempt into the ground. Another started on Klipper, hit the limits of their own knowledge, and switched to Marlin *"so I can have more support from you all,"* picking the platform with the most community behind it. Each reached for whatever asked the least of them at the firmware step, which is the toolchain barrier from the survey, observed in the wild.

A separate project shows the same wall from the other side. Alongside these builds, a graduation student building their own non-orthogonal five-axis printer on an Ender 5 Pro drew on parts of Rep5x but ran a project of their own. They found embedding into the Rep5x firmware *"too daunting for now,"* solved the inverse kinematics offline, and printed from a stock Klipper setup, reaching a first successful five-axis print and then continuous-rotation printing onto a previously printed mandrel, solving a bowden-tube twist with a makeshift slip coupler along the way. It isn't a Rep5x reproduction and isn't counted as one.

But it's the clearest case of the pattern: someone capable enough to write their own inverse-kinematics solver still routed around the native-firmware path, which is exactly the gap this thesis leaves open.

The community did real work along the way. The build logs are where the iteration loop is visible. The C-axis re-labelling described in the design and development chapter came directly out of one of these logs, when the author of Marlin's multi-axis code joined the discussion and pointed out that the axis names should follow the NIST standard. Recurring hardware problems, bowden-tube stiffness against low-torque pancake steppers, heat creep needing a larger fan, the bowden twist under continuous rotation, were diagnosed in the open with input from other builders and the developer.

The honest summary is that the system itself builds and prints, on the developer's machine and the supervisor's, but within the assignment window none of the independent builds reached a first print. Most are in progress rather than abandoned, several of them stalled at the firmware step where help is currently focused, and the one nearby project that did print ran around the Rep5x firmware rather than on it. None of this proves an adoption rate. But a stall this consistent is itself the finding: reproduction catches not on the mechanism, which builders assemble and home without much help, but on the firmware, calibration, and toolchain that stand between a homing machine and a first print. That is the barrier this thesis set out to locate, and the independent builds locate it precisely.

6.6 WHAT THE EVALUATION SUPPORTS

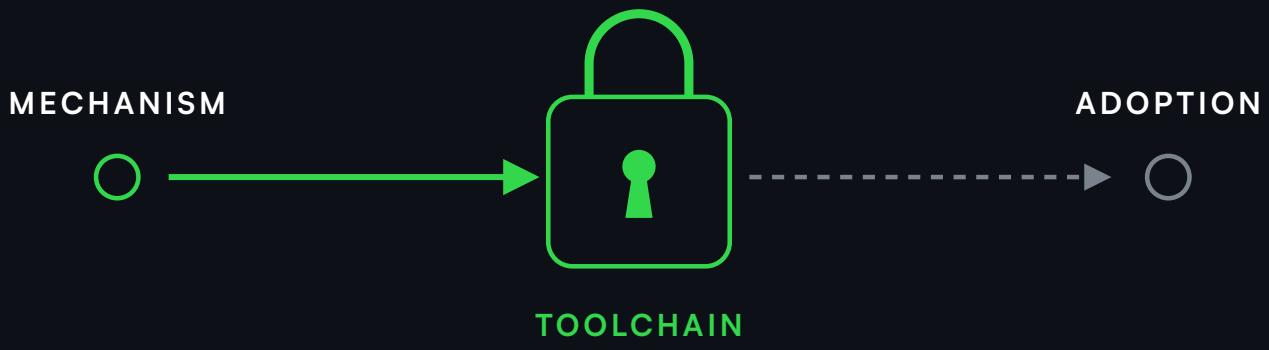
The pre-build survey confirms the software-barrier framing in the literature review and identifies a tension between desktop preference and installation friction. The Discord and iteration cases show that community feedback produces design changes; the changes themselves are described in the design and development chapter. The build attempts add a third strand: the system builds and prints on the developer's and supervisor's machines, while the independent builds, most still in progress rather than abandoned, stall at the firmware and toolchain, the same place the survey points; even a nearby project that drew on Rep5x printed only by routing around its firmware. None of this measures an adoption rate, which would need a longer and larger trial than this thesis runs for. What it does is locate the barrier precisely, and a barrier this consistent is itself a result.

CHAPTER 07

07

DISCUSSION

What the results mean: the software gate, the role of community, and what this says for open-source hardware beyond multi-axis printing.



This chapter looks at the findings together, answers the research questions, and reflects on what the results mean, both for multi-axis FDM specifically and for open-source hardware. It ends with a remark of how the project actually went, and the work that remains.

7.1 KEY FINDINGS

The clearest finding is that the barrier to multi-axis FDM lies in the toolchain more than the mechanism. Every part of the evaluation points the same way: the survey's concerns are about software, calibration, documentation and community, not the hardware; the independent builds assemble and home without much trouble, then stall at firmware and configuration; and the makers who got furthest reached for whatever firmware asked least of them. The mechanism is mature. What keeps it out of reach is the software and knowledge around it.

The hardware contribution of this thesis is, by design, small. Andersons' Robot Wrist 2, with the continuous yaw axis later added by Meijerman and Veldman [57], [58], was already working and was carried over almost unchanged. The effort went into everything around the mechanism, and the evaluation is what justifies that choice.

The development was shaped by feedback more than by a plan. The Firmware Builder, landing page, and the C-axis re-labelling as an example all came out of problems community members ran into, not a roadmap set in advance. The community didn't write code or design parts, but its questions set the direction, which is a different way of working from a single developer deciding everything alone.

7.2 ANSWERING THE RESEARCH QUESTIONS

The primary research question asked how a five-axis retrofit could be redesigned and documented so that independent makers can reproduce it. The redesign keeps Andersons' mechanism and its core-plus-bracket layout, and extends it beyond the original Ender 5 Pro with per-printer brackets and a global, sourceable bill of materials. The documentation lives in the repository and on rep5x.com. On reproduction itself the evidence is only partial: builds are in progress and the iteration cases prove that the feedback loop works, but firm adoption-rate evidence isn't there yet.

Factors that determine successful reproduction

On the first sub-question, the literature and this project point to the same set of factors: documentation quality (Antoniou et al. found that 25 of 30 practitioners named it a success factor), software accessibility (commercial dependencies, scripts for IK, etc.), an active community for support and improvement (von Hippel; Bonvoisin et al.), a multi-platform and modular design, and cost. The pattern is that any one of these missing is enough to block adoption, regardless of how strong the others are.

Infrastructure needed for replication

On the second sub-question, the infrastructure comes down to a few things: tools that replace the command line (the Firmware Builder and Calibrator), documentation kept with the code so it doesn't go out of date, a single entry point that shows the order of use (Rep5x Studio), and a community channel where recurring questions turn into new guides or tools.

Gathering and analysing community feedback

On the third sub-question, two channels did the work: a pre-build survey for framing (small N, directional only), and Discord for everything ongoing. Centralising feedback in Discord was deliberate, rather than splitting it across GitHub issues, so discussion stayed in one place and open to the whole community, not only those who file issues. The iteration cases serve as empirical proof that the loop closes. The open question is how to measure the effect of a change on the adoption rate, rather than simply the volume of feedback it generates.

7.3 IMPLICATIONS

For multi-axis FDM specifically, the implication is that the technology is mature and the toolchain is the bottleneck. Removing the Rhino dependency, even just around the slicer rather than replacing slicing itself, widens the potential audience by an order of magnitude.

For open-source hardware more generally, community-driven iteration produces changes that a solo developer cannot predict, which argues for investing in a feedback channel early rather than late. For academic open-source hardware in particular, a published paper is not an open-source release: reproduction needs the CAD, the BOM, the source, and a build guide as first-class artefacts. Andersons' thesis was open-access, but a paper isn't a release: without the CAD, BOM, source, and a build guide as first-class artefacts, even good work stays hard to reproduce, which is a large part of why this one wasn't picked up.

There is a methodological implication too. The Design Science Research cycle iterates faster when community feedback is inside the loop: several substantial design changes happened within months, none of them foreseen at the outset. And there is a policy implication: research-funded open-source hardware should be required to release the full artefact set, not just the paper.

7.4 LIMITATIONS

Several limitations qualify these findings. The survey is small (N = 89) and self-selected from people already interested in the topic. The independent build trial is incomplete, so the reproduction-rate evidence is partial. The Discord community, at around 200 members, is small enough that the strength of the iteration loop could shift in either direction at scale. No formal per-tool usability test-

ing was carried out; the iteration cases are reactive, drawn from what builders happened to report, not from structured studies. And the project remains largely the work of a single developer: the community produces feedback and, in one case, helped test a printer bracket, but not yet substantial independent code or design contributions.

7.5 REFLECTION ON DIRECTION

The project took a different direction than first planned. The original intent was closer to a brand-building and community-growth project: a steady stream of social posts, an active push to recruit builders, and the community itself as the primary evidence that the system worked. In practice it pivoted. Far more time went into the tool studio than into outreach. The project didn't feel ready for a big outreach push yet. A push like that needs a base to bring people into, a stable toolkit and a community already going, and that was still being built.

The consequence was fewer completed independent builds during the assignment than originally hoped. Some people started a build, and a few got as far as a working, homing machine, but none reached a first print within the window. Part of this was a deliberate choice to hold back from advertising a project still full of open doors and rough edges. The end result is a smaller community than planned but a stronger base than planned, which leaves mixed feelings: disappointment that the community side didn't grow more, set against satisfaction with what was actually built.

There is also a defensible reframing. Building the system first and advertising second is a reasonable order of operations. Recruiting builders into a half-finished toolkit would have produced a different and probably worse set of problems. With the toolkit now stable enough to recruit on, an outreach push, videos, tutorials, demo content, and more channels, is the natural next step rather than a missed one.

7.6 FUTURE WORK

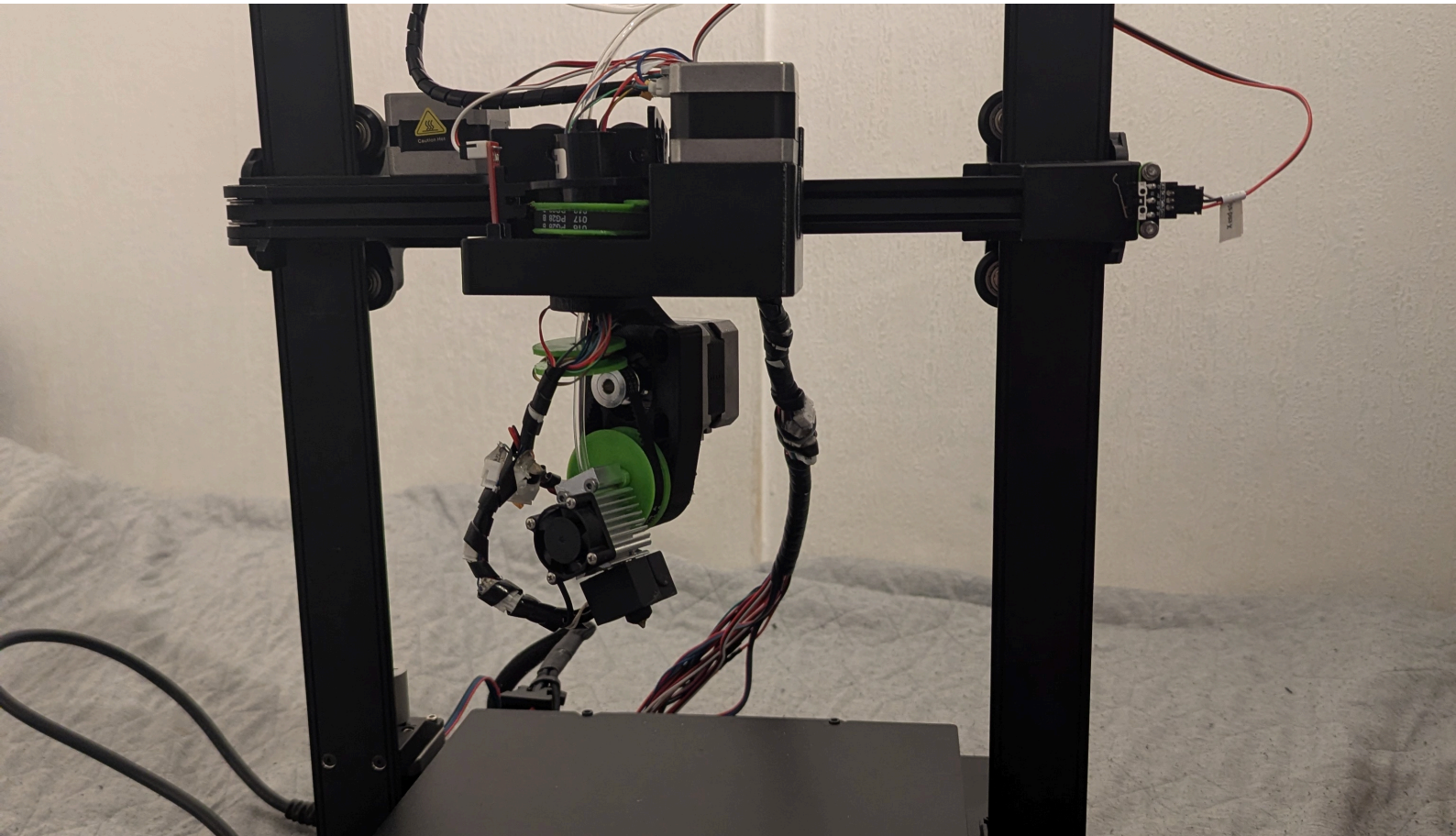
Several directions follow directly. A formal build trial would follow makers from ordering parts to a first print, including the ones already mid-build, documenting where each one gets stuck. The most significant technical gap the project leaves open is a native five-axis slicer for arbitrary geometry, whether built on the emerging OrcaSlicer forks or as a standalone browser tool drawing on the conformal-slicing literature. A Klipper port is worth pursuing, since the survey hints at a Klipper-leaning audience while Rep5x is currently Marlin-only. Calibration could be automated further, using the camera to detect the nozzle and remove the manual alignment step. An outreach push, held back during the build phase, now makes sense. Community-driven printer adaptations, contributed by builders rather than the original developer, would both broaden platform support and test the project's community-readiness. Longer term, the project needs governance, pull-request review and version cuts, for when a single developer is no longer the bottleneck, and an adoption-rate metric that measures builds completed rather than feedback volume.

CHAPTER 08

08

CONCLUSION

Conclusions and the road ahead.



8.1 SUMMARY

This thesis began from a contradiction: multi-axis FDM printing demonstrably works, yet almost nobody outside a research lab uses it. Andersons had already built an affordable five-axis retrofit of a consumer printer and shown that it produces support-free overhangs, smoother surfaces, and large reductions in material and time. The question this work set out to answer was therefore not whether the technology works, but why a working, open, affordable design had not been reproduced by anyone else, and what it would take to change that.

The answer, developed through Design Science Research and grounded in Rogers' diffusion of innovations and the open-source-hardware reproducibility literature, is that the obstacle is accessibility rather than capability. The response is Rep5x: a five-axis retrofit system designed for several common desktop printers, a Marlin fork with native five-axis kinematics, a suite of browser-based tools that remove the command line and the commercial-software dependency, and the documentation, website, and community needed to actually rebuild it. Evaluation through a pre-build survey, an active community feedback loop, and early independent build attempts supports the central claim, while leaving firm adoption-rate evidence for future work.

Each research question is answered. Reproduction, the central question, is answered in the design and tooling; independent builders completing their own builds is what will confirm it in practice. The factors that decide whether a build succeeds are documentation quality, software accessibility, an active community, multi-platform modularity, and cost, and any one of them missing is enough to stall the rest. And the infrastructure that closes the gap is browser-based tooling, version-controlled documentation that stays with the design, a guided workflow, and a community feedback loop.

8.2 CONTRIBUTIONS

The contributions of this thesis are practical, not theoretical.

Hardware for more than one printer. The mechanism is Andersons' Robot Wrist 2, kept as it was. Adding a printer just means a new mounting bracket, a few hours of work, so the same system now runs on the Ender 5 Pro and the Ender 3 V3 SE instead of one machine, with an Ender 3 Pro bracket designed for a community member building it. The bill of materials is put together so the parts can be bought anywhere.

Browser-based tools. The old workflow needed paid software, Rhino and Grasshopper, plus Python scripts run from the command line. Rep5x replaces that with tools that run in the browser with nothing to install, covering firmware, setup, printer control, calibration, and G-code. This is the part that takes away the paid software and the command line, which the survey and the literature both pointed to as the main thing holding people back.

On-machine five-axis firmware. A Marlin fork runs the inverse kinematics and error correction on the printer itself, so the same G-code works on any Rep5x machine and there's no separate post-processing step like earlier setups needed.

Documentation and community. A build guide, the rep5x.com website, and an active Discord give the project a home and a place to ask questions, all under GPLv3 and open to contributions.

All four point at the same thing: multi-axis FDM is an adoption problem, not a technical one. The mechanism already works. What was missing is the accessibility around it that lets someone else rebuild it.

8.3 RECOMMENDATIONS

For makers, multi-axis printing is no longer just for specialists. The parts are cheap, the tools run in the browser, and the build guide assumes no prior firmware or five-axis experience. What it still costs is time, mostly in sourcing the parts and building the printer. Firmware and calibration used to be the hard parts, but the tools handle those now; what's still genuinely hard is slicing for arbitrary geometry, which the project leaves open for now. And since no independent build has finished yet, there aren't yet community results to point to, so some of the payoff has to be taken on trust for now. It suits the maker who finds the build itself worthwhile; the smoothest route is a supported printer, an Ender 3 or Ender 5, with the Discord to fall back on when something breaks.

For researchers and funders of open-source hardware, the recommendation is clearer: a paper is not a release. Work published as open hardware should come with the CAD, the BOM, the source, and a build guide, and the documentation and a way to get feedback should be there from the start, not added at the end. If only its author can rebuild it, it's not really open. Andersons' retrofit is a good example: it was open-access, cheap, and worked well, but no one else rebuilt it, not because anything was wrong with it, but because a thesis doesn't come with the build guide and tools a maker needs. Closing that gap is what this work set out to do.

For the project itself, the priority now is reach rather than more features. The toolkit is stable enough to recruit on, so the next phase is outreach and a longer study that follows builders from start to finish, both held off during the build phase and set out in the future work. It will also need some governance, reviewing pull requests and cutting versions, once it grows past a single developer. The one technical gap still worth closing first is a native five-axis slicer, the last specialist dependency the toolchain leaves in place.

8.4 CLOSING REMARKS

Andersons proved that a maker could build a five-axis printer. This thesis set out to make sure they actually can. The gap between those two statements, between a working prototype in a lab and a system someone else can reproduce at home, is small in engineering terms and large

in every other way. Closing it takes documentation, tools, and a community more than it takes new mechanics, and that's the work that decides whether a good idea stays in a thesis or actually gets built.

REFERENCES

[1] I. Gibson, D. Rosen, B. Stucker, and M. Khorasani, *Additive Manufacturing Technologies*, 3rd edn. Cham: Springer, 2021. [Online]. Available: https://doi.org/10.1007/978-3-030-56127-7

[2] C. W. Hull, 'Apparatus for Production of Three-Dimensional Objects by Stereolithography'. [Online]. Available: https://patents.google.com/patent/US4575330A/en

[3] ISO/ASTM, 'Additive manufacturing — General principles — Fundamentals and vocabulary', Standard ISO/ASTM52900:2021, 2021. [Online]. Available: https://www.iso.org/standard/74514.html

[4] S. S. Crump, 'Apparatus and Method for Creating Three-Dimensional Objects'. [Online]. Available: https://patents.google.com/patent/US5121329A/en

[5] 'Fused Deposition Modeling'. Accessed: May 08, 2026. [Online]. Available: https://www.goengineer.com/3d-printing/fused-deposition-modeling

[6] R. Jones *et al.*, 'RepRap — the replicating rapid prototyper', *Robotica*, vol. 29, no. 1, pp. 177—191, 2011, doi: 10.1017/S026357471000069X.

[7] 'Ender-3 V3 SE 3D Printer'. Accessed: May 08, 2026. [Online]. Available: https://www.creality3dofficial.com/de/products/ender-3-v3-se-3d-printer

[8] 'Bambu Lab A1 mini'. Accessed: May 08, 2026. [Online]. Available: https://eu.store.bambulab.com/nl/products/a1-mini

[9] 'Original Prusa MK4S 3D Printer'. Accessed: May 08, 2026. [Online]. Available: https://www.prusa3d.com/product/original-prusa-mk4s-3d-printer/

[10] RepRap Community, 'RepRap'. Accessed: Mar. 16, 2026. [Online]. Available: https://reprap.org/wiki/RepRap

[11] 'Interview with Dr. Adrian Bowyer, Inventor of the RepRap'. Accessed: May 08, 2026. [Online]. Available: https://www.3dsourced.com/interviews/reprap-dr-adrian-bowyer/

[12] J. Jiang, J. Stringer, X. Xu, and R. Y. Zhong, 'Investigation of printable threshold overhang angle in extrusion-based additive manufacturing for reducing support waste', *International Journal of Computer Integrated Manufacturing*, vol. 31, no. 10, pp. 961—969, 2018, doi: 10.1080/0951192X.2018.1466398.

[13] N. Zohdi and R. C. Yang, 'Material Anisotropy in Additively Manufactured Polymers and Polymer Composites: A Review', *Polymers*, vol. 13, no. 19, p. 3368, 2021, doi: 10.3390/polym13193368.

[14] P. Tang *et al.*, 'A review of multi-axis additive manufacturing: Potential, opportunity and challenge', *Additive Manufacturing*, vol. 83, p. 104075, Mar. 2024, doi: 10.1016/j.addma.2024.104075.

[15] D. Ahlers, F. Wasserfall, N. Hendrich, and J. Zhang, '3D Printing of Nonplanar Layers for Smooth Surface Generation', in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2019, pp. 1737—1743. doi: 10.1109/COASE.2019.8843116.

[16] 'Caracol 3D Printers'. Accessed: May 08, 2026. [Online]. Available: https://3dprintingcanada.com/pages/caracol

[17] Q5D Technology, 'CY1000 Robotic 5-Axis 3D Printing Cell'. Accessed: May 08, 2026. [Online]. Available: https://q5d.com/news/press-release-q5d-launches-cy1000/

[18] J. Andersons, 'Exploring the Benefits and Limitations of Multi-Axis 3D Printing for Improved Part Quality and Reduced Waste', Master's thesis, Enschede, The Netherlands, 2023.

[19] E. M. Rogers, *Diffusion of Innovations*, 5th edn. New York: Free Press, 2003.

[20] R. Antoniou, J. Bonvoisin, P.-Y. Hsing, E. Dekoninck, and D. Defazio, 'Defining Success in Open Source Hardware Development Projects: A Survey of Practitioners', *Design Science*, vol. 8, p. e8, 2022, doi: 10.1017/dsj.2021.30.

[21] J. Bonvoisin and R. Mies, 'Measuring Openness in Open Source Hardware with the Open-o-Meter', *Procedia CIRP*, vol. 78, pp. 388—393, 2018, doi: 10.1016/j.procir.2018.08.306.

[22] F. Hong, S. Hodges, C. Myant, and D. Boyle, 'Open5x: Accessible 5-axis 3D printing and conformal slicing', in *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1—6. doi: 10.1145/3491101.3519782.

[23] C. Dai, C. C. L. Wang, C. Wu, S. Lefebvre, G. Fang, and Y.-J. Liu, 'Support-free volume printing by multi-axis motion', *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1—14, Aug. 2018, doi: 10.1145/3197517.3201342.

[24] Y. Yao, L. Cheng, and Z. Li, 'A comparative review of multi-axis 3D printing', *Journal of Manufacturing Processes*, vol. 120, pp. 1002—1022, 2024, doi: 10.1016/j.jmapro.2024.04.084.

[25] G. Cocco, E. Garner, V. Belle, C. Zanni, and X. Chermain, 'Towards Accessible Non-Planar FFF Using Triple Z-Axis Kinematics', in *Proceedings of the ACM Symposium on Computational Fabrication*, Cambridge, MA, United States, Nov. 2025, pp. 1—12. doi: 10.1145/3745778.3766652.

[26] D. Brogan, 'Fractal 5 Pro'. Accessed: Mar. 23, 2026. [Online]. Available: https://github.com/fractalrobotics/Fractal-5-Pro

[27] M. Wüthrich, M. Gubser, W. J. Elspass, and C. Jaeger, 'A Novel Slicing Strategy to Print Overhangs without Support Material', *Applied Sciences*, vol. 11, no. 18, p. 8760, 2021, doi: 10.3390/app11188760.

[28] R. K. Mueller, 'Penta Axis (PAX) 5-Axis Printing Option'. Accessed: Mar. 23, 2026. [Online]. Available: https://xyzdims.com/2021/02/08/3d-printing-penta-axis-pax-5-axis-printing-option/

[29] Buzzloopster, 'Voron Trident 5-axis'. Accessed: Mar. 23, 2026. [Online]. Available: https://github.com/Buzzloopster/Voron-Trident-5-axis

[30] A. Gleadall, 'FullControl GCode Designer: open-source software for unconstrained design in additive manufacturing', *Additive Manufacturing*, vol. 46, p. 102109, 2021, doi: 10.1016/j.addma.2021.102109.

[31] Julianh72, 'My Prusa Mendel RepRap (with RepRapped Filament Spool)'. Accessed: May 08, 2026. [Online]. Available: https://commons.wikimedia.org/wiki/File:My_Prusa_Mendel_RepRap_(with_RepRapped_Filament_Spool).jpg

[32] J. M. Pearce, 'Building Research Equipment with Free, Open-Source Hardware', *Science*, vol. 337, no. 6100, pp. 1303—1304, 2012, doi: 10.1126/science.1228183.

[33] B. T. Wittbrodt *et al.*, 'Life-Cycle Economic Analysis of Distributed Manufacturing with Open-Source 3-D Printers', *Mechatronics*, vol. 23, no. 6, pp. 713—726, 2013, doi: 10.1016/j.mechatronics.2013.06.002.

[34] J. Bonvoisin, R. Mies, J.-F. Boujut, and R. Stark, 'What is the Ṣource" of Open Source Hardware?', *Journal of Open Hardware*, vol. 1, no. 1, 2017, doi: 10.5334/joh.7.

[35] M. Oellermann *et al.*, 'Open Hardware in Science: The Benefits of Open Electronics', *Integrative and Comparative Biology*, vol. 62, no. 4, pp. 1061—1075, 2022, doi: 10.1093/icb/icac043.

[36] S. Oberloier and J. M. Pearce, 'General Design Procedure for Free and Open-Source Hardware for Scientific Equipment', *Designs*, vol. 2, no. 1, p. 2, 2018, doi: 10.3390/designs2010002.

[37] E. Sells, Z. Smith, S. Bailard, A. Bowyer, and V. Olliver, 'RepRap: The Replicating Rapid Prototyper: Maximizing Customizability by Breeding the Means of Production', *Handbook of Research in Mass Customization and Personalization*. World Scientific, 2010. doi: 10.1142/9789814280280_0028.

[38] Verashape, 'VSHAPER 5AX - Professional 3D Printing five axis gamechanger!'. Accessed: May 27, 2026. [Online]. Available: https://vshaper.com/3d-printers/vshaper-5ax-machine

[39] M. Molitch-Hou, '5axisworks Brings Low-Cost 5-Axis Milling, 3D Printing Machine to Kickstarter', *3D Printing Industry*, Sept. 2014, Accessed: May 27, 2026. [Online]. Available: https://3dprintingindustry.com/news/5axisworks-brings-low-cost-5-axis-milling-3d-printing-machine-kickstarter-33406/

[40] E. von Hippel, *Democratizing Innovation*. Cambridge, MA: MIT Press, 2005.

[41] Marlin Firmware, 'What is Marlin?'. Accessed: Mar. 11, 2026. [Online]. Available: https://marlinfw.org/docs/basics/introduction.html

[42] Prusa Research, 'About Us'. Accessed: Mar. 11, 2026. [Online]. Available: https://www.prusa3d.com/page/about-us_77/

[43] Prusa Research, 'Open-Source at Prusa Research'. Accessed: Mar. 11, 2026. [Online]. Available: https://www.prusa3d.com/page/open-source-at-prusa-research_236812/

[44] J. Prusa, 'The State of Open-Source in 3D Printing in 2023'. [Online]. Available: https://blog.prusa3d.com/the-state-of-open-source-in-3d-printing-in-2023_76659/

[45] D. Dougherty, 'The Story of Voron Design', *Make: Magazine*, Mar. 2023, [Online]. Available: https://makezine.com/article/digital-fabrication/3d-printing-workshop/the-story-of-voron-design/

[46] VORON Design, '15000 serialized VORON printers in the wild'. Accessed: June 02, 2026. [Online]. Available: https://forum.vorondesign.com/threads/15000-serialized-voron-printers-in-the-wild.2315/

[47] Voron Design, 'VORON 2.4'. Accessed: May 08, 2026. [Online]. Available: https://vorondesign.com/voron2.4

[48] Klipper3d, 'Klipper Features'. Accessed: Mar. 11, 2026. [Online]. Available: https://www.klipper3d.org/Features.html

[49] A. R. Hevner, S. T. March, J. Park, and S. Ram, 'Design Science in Information Systems Research', *MIS Quarterly*, vol. 28, no. 1, pp. 75—105, 2004, doi: 10.2307/25148625.

[50] R. J. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Berlin, Heidelberg: Springer, 2014. doi: 10.1007/978-3-662-43839-8.

[51] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, 'A Design Science Research Methodology for Information Systems Research', *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45—77, 2007, doi: 10.2753/MIS0742-1222240302.

[52] Twenty Technologies, 'Twenty: The #1 Open-Source CRM'. [Online]. Available: https://twenty.com/

[53] Robert McNeel & Associates, 'Rhino 8 — Pricing'. Accessed: June 02, 2026. [Online]. Available: https://www.rhino3d.com/store/

[54] J. Rood, 'Partially constructed Prusa i3'. Accessed: May 08, 2026. [Online]. Available: https://commons.wikimedia.org/wiki/File:Partially_constructed_Prusa_i3.jpg

[55] LinuxCNC, 'Machining Center: Rotational Axes'. Accessed: May 09, 2026. [Online]. Available: https://linuxcnc.org/docs/html/gcode/machining-center.html#_rotational_axes

[56] Siemens AG, 'Milling with Sinumerik: 5-Axis Machining Manual', technical report, 2009. Accessed: May 09, 2026. [Online]. Available: https://cache.industry.siemens.com/dl/files/454/37335454/att_110322/v1/SIN_WF5_0509_en.pdf

[57] J. L. Meijerman, H. J. J. Veldman, and J. A. Andersons, 'Implementation of a Continuous Yaw Axis in a 5-Axis FDM Printer', Bachelor's assignment, Enschede, The Netherlands, 2025.

[58] H. J. J. Veldman and J. A. Andersons, '5-Axis FDM Continuous Rotation', Bachelor's assignment, Enschede, The Netherlands, 2025.

[59] BIGTREETECH, 'Octopus V1.1 32-bit Controller Board'. Accessed: May 08, 2026. [Online]. Available: https://global.bttwiki.com/Octopus.html

[60] DerAndere, 'Multi-axis-Marlin'. Accessed: May 27, 2026. [Online]. Available: https://github.com/DerAndere1/Marlin

[61] K. Yanev and G. Seguin, 'Printrun: Pure Python 3D printing host software (Pronterface, Pronsole, Printcore)'. Accessed: May 27, 2026. [Online]. Available: https://github.com/kliment/Printrun

[62] Bold and Open, 'How Prusa became one of the most likable and fastest-growing startups in the world'. Accessed: June 02, 2026. [Online]. Available: https://www.boldandopen.com/blog/how-3d-prusa-became-one-of-the-most-likable-and-fastest-growing-startups-in-the-world

[63] Bambu Lab, 'BambuStudio'. Accessed: July 01, 2026. [Online]. Available: https://github.com/bambulab/BambuStudio

[64] FullControl, 'Open-Source 4-Axis Printer Conversion and FullControl Toolpath Generation'. Accessed: July 01, 2026. [Online]. Available: https://github.com/FullControlXYZ/multiaxis

[65] T. Zhang *et al.*, 'S^3-Slicer: A General Slicing Framework for Multi-Axis 3D Printing', *ACM Transactions on Graphics*, vol. 41, no. 6, 2022, doi: 10.1145/3550454.3555516.

[66] J. Qu *et al.*, 'INF-3DP: Implicit Neural Fields for Collision-Free Multi-Axis 3D Printing'. [Online]. Available: http://arxiv.org/abs/2509.05345

SUPPORTING MATERIAL

APPENDICES

Case studies in open-source 3D printing, existing multi-axis FDM systems, multi-axis slicing tools, the community survey instrument, and the statement on the use of AI.

APPENDIX A

A CASE STUDIES IN OPEN-SOURCE 3D PRINTING

This appendix examines five projects that have shaped the open-source 3D printing ecosystem. Each is analysed along the same dimensions: development model, documentation and community structure, and adoption characteristics using Rogers' framework [19]. The goal isn't to analyse each project in full, but to see what drove adoption and what didn't.

RepRap

Background. The RepRap project was initiated in 2005 by Adrian Bowyer at the University of Bath, with the goal of creating a 3D printer capable of manufacturing a significant proportion of its own parts [10]. The first functional machine, "Darwin," demonstrated self-replication in May 2008 when it produced a complete set of its own printable parts [6]. The premise was straightforward: a self-replicating manufacturing machine, released as open-source hardware, that anyone could build.

Development model. RepRap was released under the GNU General Public License (GPL), the same copyleft licence used for Linux. This was a deliberate choice: the GPL's requirement that derivative works also be open-source meant that improvements would flow back to the community. The self-replication aspect amplified this. When a printer can manufacture parts for copies of itself, the distribution mechanism is built into the product. Jones et al. [6] report that RepRap achieved 48% self-replication by part count, with the remainder being standard COTS components.

The development model was decentralised from the start. There was no company, no commercial entity. Contributions came from individual makers and small groups worldwide. This meant fast iteration but also fragmentation: multiple incompatible forks, inconsistent documentation, and no single authoritative version. The community coined "Fused Filament Fabrication" (FFF) as a trademark-free alternative to Stratasys' "Fused Deposition Modeling" [6].

Documentation and community. Early RepRap documentation lived on a MediaWiki instance (reprap.org), which anyone could edit. The wiki-based approach was good for collecting diverse knowledge but made it difficult to maintain consistency or find a clear build path. A newcomer looking to build a RepRap in 2010 would find dozens of printer variants, hundreds of wiki pages of varying quality, and no single authoritative build guide.

Community interaction happened primarily through forums, IRC channels, and later, social media groups. There was no formal support structure. This worked for the innovator and early-adopter demographics that RepRap attracted, but it was a barrier for anyone less technically confident.

Adoption analysis. Using Rogers' five attributes:

- **Relative advantage:** High. RepRap offered something genuinely new: a desktop manufacturing tool for under $500 when commercial equivalents cost tens of thousands.
- **Compatibility:** Low. Nothing about RepRap was compatible with existing workflows. Building one required sourcing obscure components, learning about stepper motors and firmware, and a lot of patience.
- **Complexity:** High. Building an early RepRap was a substantial project requiring mechanical assembly, electronics wiring, firmware configuration, and troubleshooting.
- **Trialability:** Low. There was no way to "try" RepRap without committing to a full build. Finished printers weren't available for purchase.
- **Observability:** Medium. The project had good online visibility through forums, blogs, and YouTube, but few people encountered a RepRap in person.

RepRap succeeded despite scoring poorly on four of five adoption attributes because the relative advantage was strong enough that it attracted a dedicated innovator community. But the project itself never crossed into mainstream adoption. That required the commercial derivatives (MakerBot, Prusa) that took the RepRap design and addressed the other four attributes.

Key lessons. Self-replication as a distribution mechanism is effective but not sufficient. Open-source licensing ensures improvements flow back to the community. But decentralised development without clear documentation standards creates fragmentation that limits adoption beyond the innovator demographic.

Marlin firmware

Background. Marlin is an open-source firmware for FDM 3D printers, originally created by Erik van der Zalm and first released on GitHub on 12 August 2011 [41]. It was derived from two earlier open-source projects: Sprinter and grbl. Marlin runs directly on the printer's microcontroller, handling G-code interpretation, stepper motor control, temperature management, and all real-time operations as a single self-contained application.

Development model. Marlin is licensed under GPLv3 and hosted on GitHub, where it has accumulated over 17,000 stars, 19,000 forks, and contributions from roughly 1,100 developers. The project's lead maintainer since 2014 is Scott Lahteine (known as "thinkyhead"), whose work is funded entirely through community donations via Patreon and GitHub Sponsors. This is a common model for critical

open-source infrastructure: a small number of dedicated maintainers supported by the community.

The project went through a major architectural overhaul with Marlin 2.0 (released 2019), which added support for 32-bit ARM-based controller boards through a hardware abstraction layer (HAL). This was essential for keeping Marlin competitive as the hardware ecosystem moved beyond 8-bit AVR microcontrollers.

Documentation and community. Marlin's documentation is hosted on its official website (marlinfw.org), with configuration guides, G-code references, and hardware compatibility lists. Configuration is done by editing two C++ header files (`Configuration.h` and `Configuration_adv.h`), which contain hundreds of parameters with inline comments. This approach is flexible but intimidating: changing a single setting requires opening a code editor, finding the correct line among thousands, editing it, recompiling the firmware, and flashing it to the printer.

Community support operates through GitHub issues, a Discord server (over 14,000 members), and various third-party forums (Reddit, Facebook groups). Major printer manufacturers (Creality, Prusa, LulzBot, Ultimaker) ship Marlin variants with their machines, which means millions of printers run some version of Marlin, even if many users don't know it [41].

Adoption analysis.

- **Relative advantage:** High. Marlin is mature, well-tested, and supports an enormous range of hardware configurations. For printer manufacturers, it's a proven foundation that saves years of firmware development.
- **Compatibility:** High. Marlin supports virtually every FDM printer architecture: cartesian, delta, CoreXY, SCARA. It works with both 8-bit and 32-bit boards. Most hardware is "Marlin-compatible" by default.
- **Complexity:** Medium-high. For manufacturers embedding Marlin, complexity is manageable. For end users wanting to modify settings, the header-file configuration model is a significant barrier. There's no graphical configuration tool for most settings.
- **Trialability:** High for printer buyers (Marlin comes pre-installed). Low for makers wanting to customise firmware (requires recompilation and flashing).
- **Observability:** High. Marlin's dominance means nearly every FDM printer tutorial, guide, and troubleshooting resource references it.

Marlin's adoption was driven primarily by manufacturers embedding it in commercial products. The GPL licence means these manufacturers must provide source code, which feeds improvements back into the project. But Marlin's configuration model (editing C++ header files, recompiling, flashing) represents a significant accessibility barrier for individual users wanting to modify their firmware, which is directly relevant to the Rep5x use case.

Key lessons. Becoming the default through manufacturer adoption creates a self-reinforcing cycle: more users means more testing, more bug reports, and more community knowledge. But a configuration model designed for developers (header files, compilation) creates a barrier when non-developers need to make changes. For Rep5x, this means the firmware configuration process needs to be abstracted behind a more accessible interface.

Prusa Research

Background. Prusa Research was founded in 2012 by Josef Průša, a Czech maker who had been active in the RepRap community. He simplified the RepRap Mendel design and shared it on GitHub, where it gained wider adoption than the original. The company started as a one-person operation shipping printer parts from a basement workshop, with no external funding or Kickstarter campaigns [42]. By 2019, Prusa Research had grown to over 250 employees and €70 million in annual revenue [62], making it one of the world's largest 3D printer manufacturers.

Development model. Prusa's model is a hybrid: open-source design with commercial distribution. Historically, all printer designs were released under the GPL, with firmware and slicer software (PrusaSlicer) also open-source. The company made money by selling assembled printers and kits, not by restricting access to the designs. Roughly 80% of customers chose the kit version, suggesting that the target audience actively wants the build experience.

This model came under pressure in the 2020s. New entrants, particularly Bambu Lab, brought competing CoreXY printers to market and built on community-developed software including forks of PrusaSlicer [63]. Bambu's hardware is its own design rather than a clone of Prusa's, but the broader effect on Prusa's market position was the same. In response, Prusa introduced the Open Community Licence (OCL) in 2024, which permits modification and personal use but restricts commercial sale of complete machines based on the designs [43]. The Core One, released in late 2024, initially shipped without open-source hardware, though Prusa later released CAD files under the OCL in December 2025. Firmware (GPLv3) and PrusaSlicer (AGPLv3) remain fully open, but the shift from GPL to the more restrictive OCL for hardware is a notable departure from the company's founding principles.

Documentation and community. Prusa's documentation is among the best in the industry. Each printer ships with an assembly manual (for kits) and a usage guide. The company maintains an extensive online knowledge base with troubleshooting guides, video tutorials, and firmware update instructions. PrusaSlicer includes printer profiles that work out of the box.

The community operates through official forums (143,000+ members), a Discord server, a Reddit community, a GitHub presence, and social media channels. A substantial portion of Prusa's staff is dedicated to customer support, handling thousands of live chats and emails monthly in multiple languages [62]. This investment in support infrastructure is a key differentiator: unlike pure community-driven projects, Prusa provides professional, reliable support.

Adoption analysis.

- **Relative advantage:** High. Prusa printers offered a reliable, well-documented 3D printing experience at a reasonable price point, with the added benefit of being open-source.
- **Compatibility:** High. PrusaSlicer works with standard file formats, the printer uses standard filaments, and the ecosystem is compatible with the broader FDM community's practices.
- **Complexity:** Low to medium. Kit assembly is well-guided. Using the printer is straightforward thanks to good defaults in PrusaSlicer and a polished LCD interface.
- **Trialability:** High. Buyers can purchase a complete kit or assembled unit with minimal risk. The strong online community and support infrastructure reduce perceived risk.
- **Observability:** High. Prusa has a large online presence with active forums, YouTube content, and widespread visibility in the maker community.

Prusa is the clearest example of how to take an open-source design from the innovator stage to mainstream adoption. The key wasn't the hardware (which was iterative, not revolutionary). It was the *packaging*: documentation with assembly photos and troubleshooting, a reliable supply chain, professional support, and a brand that signalled quality. Prusa took RepRap's open-source foundation and addressed every adoption barrier that RepRap itself had left unresolved.

Key lessons. Open-source design and commercial viability aren't mutually exclusive, but the business model needs to be grounded in something beyond the design files themselves (in Prusa's case: manufacturing quality, documentation, and support). When competitors can access the same open-source designs, differentiation comes from execution, not intellectual property. The recent shift to the OCL licence also illustrates the tension between openness and commercial sustainability.

Voron Design

Background. Voron Design was founded in 2015 by Maksim Zolin, a Russian-born maker living in the United States who wanted a faster, quieter, and better-looking printer than what was commercially available [45]. After briefly selling kits through a company called MZBOT (2016—2018), Zolin concluded he didn't want to run a commercial operation and instead released the designs as fully open-source, inviting community collaboration.

Development model. Voron designs are released under GPLv3. The project is a nonprofit organisation that develops specifications for open hardware. Unlike Prusa, there's no commercial entity selling Voron printers. Builders source their own components ("self-sourcing") using detailed BOMs provided by the project, or purchase third-party kits from vendors who bundle the parts. The project has no financial stake in kit sales.

Multiple printer designs exist for different use cases: the Voron 0 (compact, 120×120×120 mm build volume), Voron Trident (medium, three independent Z motors), and Voron 2.4 (large, flying gantry). All use CoreXY kinematics for speed and print quality. The designs are maintained on GitHub, with over 4,300 stars and 1,000 forks for the Voron 2 repository alone.

Community contributions are formalised through a modification ("mod") ecosystem. Users design improvements, share them through the community, and the best ones are integrated into the mainline specification. By 2025, over 15,000 Voron printers had been serialised (registered as completed builds) [46].

Documentation and community. Voron's documentation is community-maintained and covers hardware sourcing, mechanical assembly, electrical wiring, firmware configuration, and tuning in step-by-step depth. The recommended firmware is Klipper (discussed below), and the documentation includes complete Klipper configuration files for each printer variant.

The community operates primarily through Discord (nearly 40,000 members), which is described as "the heart of the Voron community." Discord provides real-time support channels organised by printer model, build stage, and topic. This real-time support model is particularly effective for complex builds where problems are hard to describe in a forum post but easy to diagnose through a conversation.

The serialisation system is a notable community feature. Builders who complete a Voron to specification can apply for a serial number, which serves as both a quality gate (the build must pass visual inspection) and a community milestone. This gamification element creates social proof and motivation.

Adoption analysis.

- **Relative advantage:** High. Voron printers offer performance comparable to or exceeding commercial printers costing significantly more, with full customisability.
- **Compatibility:** Medium. Voron uses Klipper firmware rather than the more common Marlin, which means a different configuration standard. But it uses standard file formats, slicers, and filaments.
- **Complexity:** High. Building a Voron is a multi-day project requiring mechanical assembly, wiring, firmware configuration, and extensive tuning. The project explicitly targets experienced builders.
- **Trialability:** Low. There's no way to try a Voron without committing to a full build. Third-party kits reduce sourcing effort but not build complexity.
- **Observability:** High. Voron has a strong presence on YouTube, Reddit, and Discord. The serialisation system and active community create high visibility.

Voron demonstrates that a purely community-driven project, with no commercial entity, can achieve significant adoption if the community infrastructure is strong enough. The Discord community effectively replaces the professional support that Prusa provides commercially. But Voron's complexity means it appeals primarily to the enthusiast segment. It hasn't crossed into the early majority, and it isn't trying to.

Key lessons. Community can substitute for commercial support infrastructure, but only for audiences that are willing to engage with the community. The serialisation

system is an effective mechanism for quality assurance and motivation. The mod ecosystem enables distributed innovation without fragmenting the project. But the self-sourcing model and high build complexity limit adoption to a specific demographic.

Klipper firmware

Background. Klipper is an open-source 3D printer firmware created by Kevin O'Connor, with the first code released on GitHub on 25 May 2016 [48]. Its defining characteristic is an architectural departure from traditional firmware: instead of running all processing on the printer's microcontroller, Klipper splits the workload between a host computer (typically a Raspberry Pi) and the microcontroller. The host handles G-code parsing, motion planning, and kinematics calculations, while the microcontroller handles only real-time stepper pulse generation.

Development model. Klipper is licensed under GPLv3 and hosted on GitHub, with over 11,000 stars, 5,800 forks, and roughly 500 contributors. Kevin O'Connor remains the primary maintainer, supported by community contributions and funding through Patreon. The project has a more centralised governance model than Marlin: O'Connor maintains tight control over what gets merged into the mainline codebase.

The host-based architecture has a significant consequence for configuration: changes don't require recompilation or flashing. Klipper's configuration is a plain-text file (`printer.cfg`) that can be edited and reloaded at runtime. This is a fundamental usability improvement over Marlin's compile-flash cycle, particularly for users who need to iterate on configuration frequently, such as those building custom or modified printers.

Documentation and community. Klipper's documentation is hosted on klipper3d.org and covers installation, configuration, calibration, and advanced features. The documentation is more developer-oriented than user-oriented: it assumes familiarity with Linux, command-line tools, and networking.

The ecosystem around Klipper includes several community-developed web interfaces: Mainsail and Fluidd provide browser-based printer control and monitoring. OctoPrint integration is also available. This web-based control paradigm aligns well with Klipper's host-based architecture and provides a significantly more modern user experience than Marlin's LCD-menu interface.

Community support happens primarily through a Discourse forum (nearly 10,000 users) and a Discord server (roughly 27,000 members). Klipper has become the dominant firmware choice in the performance-oriented and DIY segments of FDM printing, particularly among Voron builders.

Adoption analysis.

- **Relative advantage:** High. Klipper's architecture enables features that are difficult or impossible on microcontroller-only firmware: input shaping for vibration reduction (introduced 2020), pressure advance, and precise high-speed motion control. Step timing precision of 25 microseconds enables smoother, quieter operation [48].
- **Compatibility:** Medium. Klipper works with most common controller boards and printer architectures. But it requires a host computer (additional hardware) and a different configuration paradigm than Marlin.
- **Complexity:** Medium. The initial setup requires installing an operating system on the host, flashing the microcontroller firmware, and configuring the text-based config file. This is more complex than Marlin on a pre-configured printer, but ongoing configuration changes are easier.
- **Trialability:** Medium. Klipper can be installed on an existing printer without hardware modifications (beyond adding a Raspberry Pi), which lowers the barrier to experimentation.
- **Observability:** High. Klipper has strong visibility on YouTube, Reddit, and Discord. The performance benefits (faster printing, better quality through input shaping) are visually demonstrable.

Klipper's adoption was driven by a genuine technical advantage (input shaping, faster speeds) that users could see and measure. The Voron community's adoption of Klipper as its recommended firmware created a mutually reinforcing cycle: Voron brought users to Klipper, and Klipper's capabilities made Voron printers perform better.

Key lessons. Architectural decisions have direct accessibility consequences. Klipper's text-file configuration model is more accessible than Marlin's compile-flash cycle for users who need to make frequent changes. The web-based control interfaces (Mainsail, Fluidd) demonstrate how browser-based tools can improve the user experience for hardware that would otherwise require specialist knowledge. For Rep5x, this supports the rationale for building browser-based tools rather than desktop applications.

Comparative analysis

Table A.1 summarises the key characteristics of each project, and Table A.2 maps them against Rogers' adoption attributes. The same comparison appears as a radar chart in the literature review.

Several patterns emerge from this comparison.

Technical excellence alone doesn't drive adoption. All five projects offer genuine technical advantages. But adoption correlates more strongly with the other four Rogers attributes than with relative advantage. RepRap and Voron have the highest complexity and lowest trialability, and both remain confined to the enthusiast demographic. Prusa scores best across all five attributes and achieved the widest adoption.

Documentation quality separates success from failure. Prusa and Voron invest heavily in documentation. RepRap's wiki-based approach covered everything but was disorganised. Marlin's documentation is extensive but developer-oriented. The projects with the most structured, user-facing documentation have the strongest adoption.

Community can substitute for commercial support, but at a cost. Voron demonstrates that a purely commu-

	Type	Licence	Primary community	Commercial model
RepRap	Hardware	GPL	Wiki, forums	None (derivatives only)
Marlin	Firmware	GPLv3	GitHub, Discord	Donation-funded
Prusa	Hardware + software	GPL → OCL	Forums, support staff	Kit and assembled sales
Voron	Hardware	GPLv3	Discord	None (third-party kits)
Klipper	Firmware	GPLv3	Discourse, Discord	Donation-funded

TABLE A.1 Key characteristics of the five case-study open-source projects.

	Relative adv.	Compatibility	Complexity	Trialability	Observability
RepRap	High	Low	High	Low	Medium
Marlin	High	High	Med—high	High	High
Prusa	High	High	Low—med	High	High
Voron	High	Medium	High	Low	High
Klipper	High	Medium	Medium	Medium	High

TABLE A.2 The five projects scored against Rogers' five adoption attributes.

nity-driven project can provide effective support through Discord. But this only works for an audience that's willing and able to engage with the community. Prusa's investment in professional support (20% of staff) enabled it to reach a much broader audience.

Configuration accessibility matters. Klipper's text-file configuration model is more accessible than Marlin's compile-flash cycle. But both still require technical knowledge. Prusa's approach of shipping pre-configured printers with sensible defaults eliminates the configuration barrier entirely for most users.

Licence model affects sustainability. All five projects use GPL or GPL-derived licences. Prusa's recent shift to the OCL illustrates the tension between openness and commercial sustainability when competitors can clone open-source designs. For non-commercial projects like Voron and the firmware projects, GPL works well because there's no revenue to protect.

Implications for Rep5x. Rep5x combines elements of several of these projects. Like Voron, it's a community-driven hardware project with no commercial entity. Like Marlin, it involves firmware modification. Like Prusa, it aims for documentation that walks a builder through every step. And like Klipper, it uses browser-based tools to lower the configuration barrier. The case studies suggest that the combination of step-by-step documentation, accessible tooling, and active community support is more important for adoption than the hardware design itself. This informs the priorities in the following chapters.

APPENDIX B

B EXISTING MULTI-AXIS FDM SYSTEMS

This appendix looks at existing multi-axis FDM printer projects, focusing on open-source or publicly documented systems. Each is assessed on how well it addresses the adoption barriers identified in the literature review: can a maker with a different printer use it? Can a non-expert set it up and use it? Is the documentation sufficient for independent reproduction? Is there a community for support? The comparison table at the end provides a summary.

Open5x

Open5x is a 5-axis retrofit for the Prusa i3 MK3s, developed at Imperial College London [22] (Fig. B.1). It uses a table-table configuration with a 3D-printed rotary table that replaces the print bed. The design is open-source under the MIT licence, hosted on GitHub with over 1,200 stars. Most structural parts are 3D-printable, and the project includes a conformal slicer implemented as a Grasshopper plugin for Rhino.

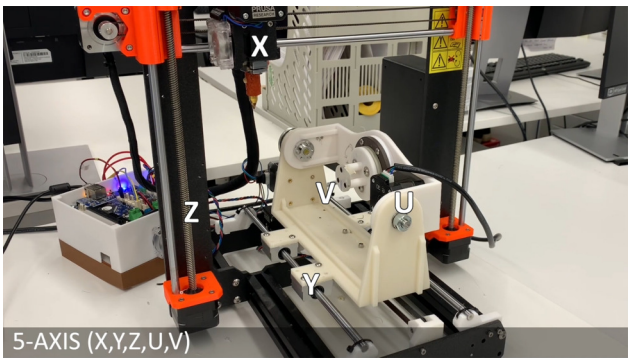


FIG. B.1 The Open5x 5-axis retrofit mounted on a Prusa i3 MK3s. Image from the Open5x GitHub repository (MIT licence) [22].

Open5x is the closest existing project to Rep5x in scope, but it has significant limitations. The rotary table mounts on the Y-axis bed of the Prusa, adding mass to the most active linear axis and limiting speed and accuracy. The slicer requires Rhino (commercial, approximately €995 [53]) and Grasshopper, which the authors themselves acknowledge: "This dependency on a commercial product means that our solution is not completely open source" [22]. Documentation is limited to the research paper; there's no build guide and no community forum. The project runs on RepRapFirmware (Duet boards), not Marlin.

A community member known as BrendonBuilds adapted Open5x for the E3D Toolchanger and Voron platforms, contributing the changes back to the main repository. This variant uses a Duet 3 controller and has been tested with engineering materials (PETG-CF, Nylon-CF). But the Rhino/Grasshopper dependency remains.

RotBot

The RotBot is a 4-axis conical printer developed at the Zurich University of Applied Sciences (ZHAW) by Wüthrich et al. [27]. Instead of adding two rotational axes, it adds one: the nozzle is tilted at 45° from vertical and rotates continuously around the Z-axis. Parts are sliced into conical layers rather than flat ones. This allows overhangs of up to 100° without support (Fig. B.2).

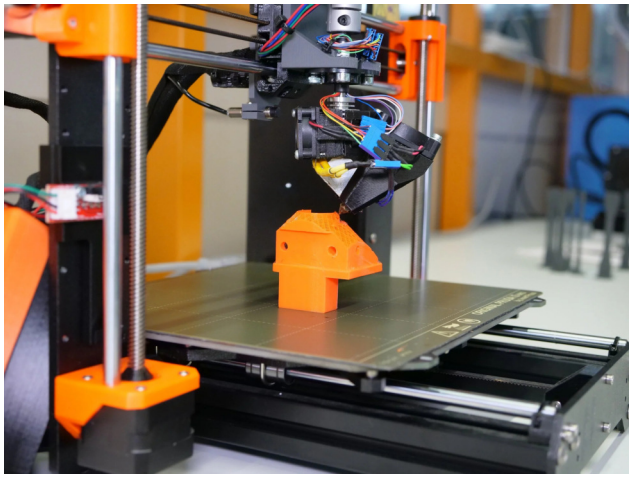


FIG. B.2 The RotBot 4-axis printer printing a part with conical layers. The tilted nozzle rotates continuously around the Z-axis, enabling overhangs up to 100° without support. Image courtesy of Stefan Hermann / CNC Kitchen.

Stefan Hermann of the CNC Kitchen YouTube channel built a replica and popularised the concept, releasing a simplified Python slicer as open-source.[5] The slicing workflow is a two-step process: the STL is mathematically transformed into a conical coordinate system, sliced with a conventional slicer (Cura, PrusaSlicer), then the G-code is back-transformed.

RotBot's strength is its simplicity. But it's fundamentally limited to 4 axes with a fixed tilt angle. It can't achieve the arbitrary nozzle orientations that a full 5-axis system provides, and the conical layer geometry constrains what parts can be printed.

Pentaxis (PAX)

Pentaxis is a head-head 5-axis design by Rene K. Mueller, documented on his XYZdims website [28]. Both rotational axes are placed at the nozzle: an A-axis for yaw rotation and a B-axis for tilt. Three variants exist (PAX 90, 135, and 180) with different tilt ranges, the number indicating the maximum tilt angle in degrees (Fig. B.3).

[5] CNC Kitchen, "Non-Planar 3D Printing with a RotBot," YouTube, October 2022.

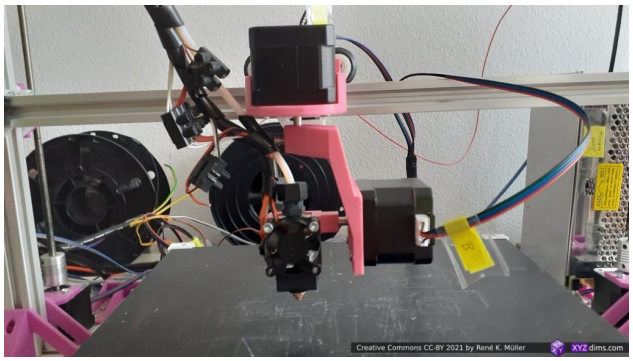


FIG. B.3 The Pentaxis PAX 90 head-head mechanism with NEMA 17 stepper motors. Image by Rene K. Mueller (CC-BY) [28].

Mueller also developed Slicer4RTN, a conic slicing tool that wraps around conventional slicers (CuraEngine, Slic3r) to add rotational axis support. The project is open-source and extensively documented on the XYZdims wiki. But it remains experimental: the hardware hasn't been widely reproduced, and full 5-axis slicing (beyond conic mode) isn't available.

The HH configuration introduces the same filament routing challenges that Andersons encountered. At extreme tilt angles, the Bowden tube bends sharply, causing extrusion inconsistencies.

Triple Z-Axis (3Z)

The 3Z system, presented by Cocco et al. at SCF 2025 [25], takes a different route to non-planar FFF. Rather than attaching a rotational gantry to the print head or a tilting plate to the build, it modifies a commercially available RatRig V-Core 3.1 by actuating its three Z-axis lead screws independently. Combined with a ball-on-rail joint that lets the bed both slide and rotate as the carriages move, the result is a bed with three controllable degrees of freedom: height plus two tilt angles.

The hardware modification is described by the authors as "minimal," involving only extensions to the rails and bed screws; no print head modification is required. The kinematic model with closed-form solutions and an open-source Python implementation that maps 3D toolpaths to machine commands are released alongside the paper.

The trade-off is range. The maximum bed tilt is limited to 30°, "constrained by the maximum outward extension of the rails" [25]. This excludes the support-free overhang printing a 5-axis HH or TT system enables, but for shallow non-planar work (smoothing curved surfaces, layer alignment) the approach reaches the maker community more directly because the host hardware is already commercially available.

Fractal 5 Pro

The Fractal 5 Pro is a custom-built 5-axis printer developed by Dan Brogan and open-sourced in 2025 under GPLv3 [26]. Unlike the retrofit approaches above, it's built from scratch using a Voron-inspired CoreXY frame with 30×30 mm aluminium extrusions. The A-axis provides continuous bed rotation (using slip rings), and the B-axis provides 90° tilt via a gear drive. It runs Klipper firmware (Fig. B.4).

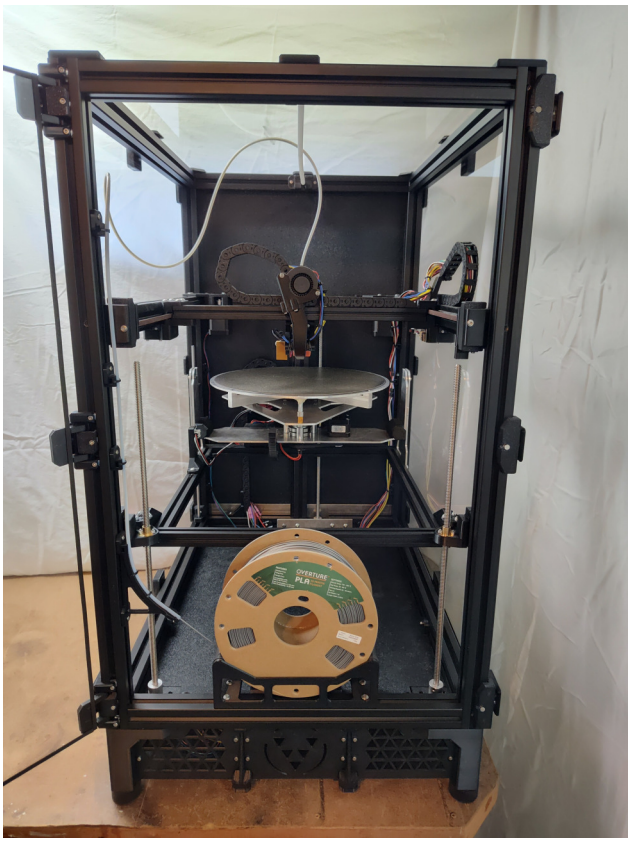


FIG. B.4 The Fractal 5 Pro, a custom-built 5-axis printer with CoreXY gantry and tilting bed. Image from the Fractal 5 Pro GitHub repository (GPLv3) [26].

Brogan also developed Fractal Cortex, a dedicated 5-axis slicer (discussed in Appendix C). The printer itself is costly at approximately $1,900 in parts, substantially more expensive than a retrofit approach. The creator has stepped back from active development, leaving the project to community stewardship.

Voron Trident 5-axis

Buzzloopster's Voron Trident 5-axis is a modification of the Voron Trident 250 that adds a C-axis (infinite rotation) and A-axis (110° swivel) in a table-table configuration [29]. It runs RepRapFirmware on a Duet board and includes both direct-drive and Bowden extruder variants (Fig. B.5).

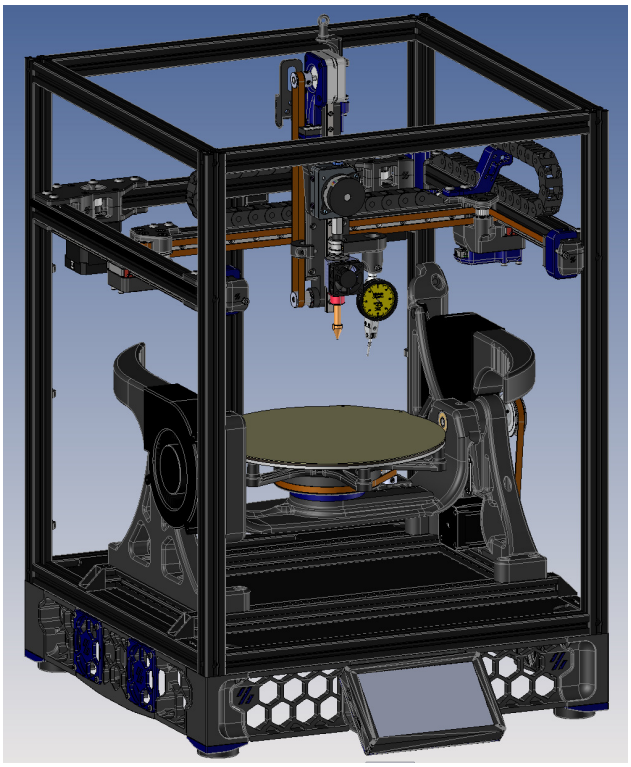


FIG. B.5 CAD render of the Voron Trident 5-axis modification showing the tilting bed mechanism. Image from the project GitHub repository (GPLv3) [29].

The project has good documentation (assembly manuals, calibration guides, BOMs) and is released under GPLv3. But it requires a Voron Trident as the base printer, which is itself a complex self-sourced build. There's no dedicated slicer, no community beyond the GitHub repository, and limited evidence of independent reproduction.

FullControl 4-axis

Andrew Gleadall's FullControl project at Loughborough University developed a minimal 4-axis conversion for the Prusa MK3S, requiring only three purchased parts and three to six 3D-printed parts [64]. It adds a single tilting axis at the print head using a Duet 3 Mini controller running RepRapFirmware (Fig. B.6). Toolpaths are generated using the FullControl Python library (see Appendix C).

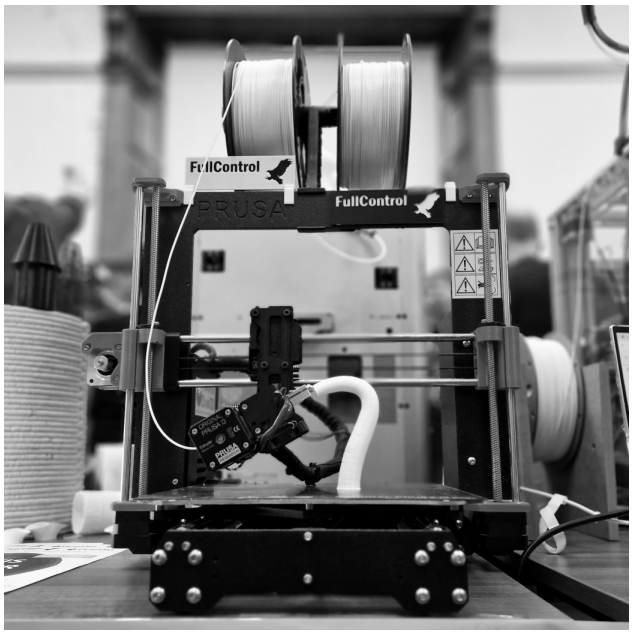


FIG. B.6 The FullControl 4-axis conversion for the Prusa i3 MK3S. Image from the FullControl GitHub repository [64].

The hardware is deliberately minimal, making it one of the most accessible multi-axis conversions available. But it's currently limited to 4 axes, with a 5-axis version announced but not yet released.

Research-only systems

Several multi-axis FDM systems exist only as academic prototypes without public hardware releases:

Shen et al. (2018) built a 5-axis system on a delta printer with an interference-free cone nozzle. The paper demonstrates tangential, normal, and combined printing modes, but no hardware files or firmware were published.

Gen5X (2023) is a 5-axis printer whose frame was designed using Autodesk Fusion 360's generative design tools, originally released under MIT on GitHub as an open-source project. By 2025, the project pivoted to a commercial model: a partnership with AiBuild produced "Generation One," a pre-order product with proprietary slicing software and yearly licence fees. The original open-source repository remains on GitHub but hasn't been updated since January 2023.

Pentarod (2015) was an early 5-axis modification of a RepRap Ormerod by a University of Oslo master's student. It demonstrated the concept but predates modern firmware support for multi-axis FDM and was never widely replicated.

Comparison

Table B.1 compares the multi-axis FDM systems against the key adoption barriers (✓ addressed, ~ partially addressed, ✗ not addressed).

The pattern is consistent. Most projects release their designs as open-source, and some provide reasonable documentation. But the barriers that matter most for adoption remain unaddressed: no project supports multiple printer platforms, software accessibility is mixed (some depend on commercial Rhino/Grasshopper, others require Python scripting), documentation rarely extends

Project	Axes	Multi-platform	Accessible software	Build docs	Community	Open-source
Open5x	5	✗	✗		✗	✓
RotBot	4	✗			✗	
PAX	5	✗			✗	✓
3Z (Cocco)	5	✗			✗	✓
Fractal 5	5	✗			✗	✓
Voron 5-axis	5	✗	✗		✗	✓
FullControl	4	✗			✗	✓

TABLE B.1 Existing multi-axis FDM systems against the key adoption barriers.

beyond a research paper, and none has built a community for ongoing builder support. Calibration is a challenge too, but it's part of a broader picture. The real gap isn't any single barrier; it's that no project addresses all of them together.

APPENDIX C

C MULTI-AXIS SLICING AND TOOLPATH TOOLS

Multi-axis printing requires different toolpath generation than conventional 3-axis slicing. This appendix surveys existing slicing and toolpath tools that support multi-axis FDM, from practical tools to research prototypes. The comparison table at the end provides a summary.

Practical tools

FullControl. FullControl is a Python library for direct G-code design, developed by Andrew Gleadall at Loughborough University [30]. Rather than taking an STL and slicing it, FullControl lets the user define every segment of the print path parametrically. The main library (over 900 GitHub stars) handles 3-axis toolpath design. A separate companion repository provides multi-axis support with hardware conversion guides and toolpath generation for 4-axis (XYZ + one rotation) and 5-axis configurations. The 4-axis system is well documented with Jupyter notebook guides; the 5-axis extension exists but is less developed.

FullControl is open-source (GPLv3) and genuinely supports multi-axis printing, not just non-planar 3-axis. But it isn't a slicer in the conventional sense. There's no STL input, no automatic infill, no support generation. The user designs the toolpath from scratch in Python, which requires programming skills and an understanding of the printing process. This makes it a tool for researchers and advanced makers, not for the typical user who wants to import a model and click "print."

Fractal Cortex. Fractal Cortex is a 5-axis slicer developed by Dan Brogan alongside the Fractal 5 Pro printer [26]. It takes a multidirectional approach: the user places "slicing planes" oriented along different sections of the part, and the slicer generates flat layers in each orientation. The printer pauses between sections while the A and B axes reorient, then resumes with planar layers in the new direction.

This isn't true non-planar or conformal slicing. Each section is sliced with flat layers; the innovation is in printing different sections at different orientations. The tool is open-source (GPLv3, Python) but early-stage, with known geometry-related bugs and limited print settings.

RotBot Transform / ConicalSlicer. The RotBot slicing workflow uses two Python scripts developed by Wüthrich et al. [27]. The first script transforms the STL geometry into a conical coordinate system. The transformed model is then sliced with any conventional slicer (Cura, PrusaSlicer). The second script back-transforms the resulting G-code into the original coordinate system with rotation commands added.

Stefan Hermann (CNC Kitchen) forked the tool as ConicalSlicer with usability improvements. Both versions are open-source (GPLv3). The approach is smart because it uses existing slicer infrastructure rather than reimplementing it. But it only works for conical layers (4-axis), not arbitrary multi-axis orientations.

Slicer4RTN. Slicer4RTN, developed by Rene K. Mueller (the PAX creator), is a wrapper that adds conic slicing capability to existing slicers [28]. It supports CuraEngine and Slic3r as backend slicers, with configurable cone angles for 3-axis (non-planar with small angles), 4-axis, and 5-axis configurations.

The tool is open-source (LGPLv3) but written in Perl, which is an unusual dependency. Development appears to have stalled (last commit June 2021), and 4-axis and 5-axis modes have only been tested in simulation.

Grasshopper/Rhino workflows. Several multi-axis toolpath tools exist as Grasshopper plugins for Rhinoceros 3D:

Andersons' iso-curves slicer [18] generates conformal toolpaths by computing iso-curves on the part surface, combined with a G-code bending approach for transforming planar toolpaths onto curved surfaces. This is the workflow that produced the validated results discussed throughout this thesis. Rep5x's browser-based tools were designed to replace the Grasshopper/Rhino toolchain around it, though the slicing step itself is still left to external tools.

Open5x Conformal Slicer [22] takes a similar approach, generating iso-curves with a GUI layer on top of Grasshopper. It adds spiral layer generation and automatic flow-rate calculation.

Caterpillar (Hao Zheng, University of Pennsylvania) translates custom Grasshopper toolpaths to G-code for non-planar printing.

Bark Beetle (Fellesverkstedet) generates 6-axis toolpaths for robotic fabrication.

All Grasshopper workflows share the same fundamental limitation: they require a Rhino licence (approximately €995). This creates an accessibility barrier that contradicts the open-source ethos of the hardware projects they're designed to work with.

Research prototypes

S³-Slicer. S³-Slicer is a general multi-axis slicing framework developed by Zhang et al. [65], which received the Best Paper Award at SIGGRAPH Asia 2022. It addresses support-free printing, strength reinforcement, and surface quality simultaneously using quaternion fields and rotation-driven deformation to compute curved layers.

The tool is open-source (BSD-3-Clause) and represents the academic frontier of multi-axis slicing. But it requires Windows, Visual Studio, and Qt to build, and isn't designed as a user-facing tool.

NeuralSlicer. NeuralSlicer [66] extends the S³-Slicer approach using neural networks to optimise the scalar field that defines curved layers. Published at SIGGRAPH 2024, it achieves improved performance and is representation-agnostic (works on diverse model types). Open-source (GPLv3, Python) but firmly in the research domain.

CurviSlicer. CurviSlicer (INRIA, 2019) generates slightly curved layers for standard 3-axis printers, reducing the staircase effect without requiring additional axes. It uses tetrahedral mesh optimisation and is open-source (AGPLv3). But it's a 3-axis tool; the curved layers are limited to what the printer's Z-axis travel can accommodate without nozzle collisions.

Commercial tools

DotX Control Solutions offers a commercial 5-axis slicer targeting industrial systems (robotic arms, CNC platforms). It supports true curved-layer slicing for welding, continuous fibre, and extrusion processes. Pricing isn't public, and the tool is aimed at industrial users, not the maker community.

Siemens NX AM Multi-Axis is an industrial-grade solution integrated into the NX CAD/CAM platform. It provides simultaneous 5-axis path planning for robots and machine tools, with planar, rotary, freeform, and tubular deposition strategies. It's part of Siemens' broader manufacturing suite and priced accordingly (enterprise licences, typically thousands of euros).

Adaxis AdaOne is a robotic additive manufacturing slicer developed by a French-Swedish startup. It supports multi-axis toolpath generation for industrial robots (ABB, KUKA, FANUC, and others) with up to 14-axis kinematic systems. It handles wire-arc AM, laser metal deposition, pellet extrusion, and filament extrusion. Like DotX and Siemens NX, it targets industrial users rather than the desktop FDM community.

SprutCAM includes additive and hybrid manufacturing programming for robotic systems, supporting multi-axis deposition alongside traditional CNC machining.

AURA by Anisoprint is a slicer for continuous fibre composite printing, with 6-axis robotic arm support under development. It's proprietary and tied to the Anisoprint hardware ecosystem.

AiBuild provides cloud-based multi-axis toolpath generation, recently partnering with Generative Machine for their Generation One 5-axis desktop printer. The software uses AI-driven path optimisation for large-format robotic printing.

The commercial landscape is active but entirely focused on industrial-scale systems. None of these tools targets desktop FDM, and their pricing (enterprise licences, per-seat fees, or hardware lock-in) puts them well beyond the reach of individual makers.

Non-planar extensions for conventional slicers

Several attempts have been made to add non-planar capability to existing slicers:

PrusaSlicer non-planar fork: Based on Daniel Ahlers' 2018 master's thesis, this fork modifies the top layers of a print to follow the surface curvature. It works on standard 3-axis printers by exploiting Z movement, but isn't true multi-axis slicing. The fork is behind current PrusaSlicer versions and hasn't been merged upstream.

OrcaSlicer/BambuStudio: Non-planar Z contouring is an open feature request but hasn't been implemented as of early 2026.

Cura: No native non-planar support or maintained plugin exists.

Comparison

Table C.1 compares the multi-axis slicing and toolpath tools. Rep5x is listed for context, but it isn't a slicer: it provides the browser-based tooling around slicing (firmware, calibration, control, preview, and parametric G-code generation), while full 5-axis slicing of arbitrary models is still left to external tools.

The tools split into three categories. Commercial tools (DotX, AURA) offer the most capability but are inaccessible to makers. Research prototypes (S³-Slicer, NeuralSlicer) push the algorithmic capabilities but aren't usable outside a lab. Practical open-source tools (FullControl, RotBot

Tool	Axes	Open	Approach	STL input	Accessibility
FullControl	4—5	Yes	Parametric	No	Python, Jupyter
Fractal Cortex	5	Yes	Multidirectional	Yes	Python, early-stage
RotBot Transform	4	Yes	Conical	Yes*	Python scripts
Slicer4RTN	3—5	Yes	Conic wrapper	Yes*	Perl, experimental
Grasshopper	3—5	Partial	Various	Yes	Rhino licence needed
S³-Slicer	Multi	Yes	Deformation	Yes	C++, research
NeuralSlicer	Multi	Yes	Neural field	Yes	Python, research
CurviSlicer	3	Yes	Curved layers	Yes	C++, research
DotX	5+	No	Curved layers	Yes	Commercial
AURA	6	No	Fibre + robotic	Yes	Commercial
Rep5x tools	**5**	**Yes**	**No slicer (tooling)**	**No**	**Browser, no install**

TABLE C.1 Multi-axis slicing and toolpath tools compared. *Uses a conventional slicer for the base step, then transforms the output.

Transform, Grasshopper workflows) bridge some of the gap but each has significant limitations: FullControl requires programming, RotBot is limited to 4 axes, and Grasshopper needs a commercial licence.

No existing tool provides the combination that a maker needs: browser-based access, no commercial dependencies, a guided workflow, and support for 5-axis FDM on consumer hardware. Rep5x's browser-based tools cover most of that, the setup, calibration, control, and preview around the slicer, but not the slicing itself. A native browser-based 5-axis slicer is the part the project deliberately leaves open (see the future work).

APPENDIX D

D COMMUNITY SURVEY

This appendix documents the pre-build interest survey described in the methodology and analysed in the evaluation chapter. It contains the full instrument and the aggregated results question by question. The survey was circulated through online maker communities and collected **89 responses** from active 3D-printer makers, and was anonymous.

Percentages are of all 89 respondents. Three questions allowed multiple answers (motivations, resources, willingness to contribute) and one capped selection at three (barriers), so those columns sum to more than 100%.

Survey instrument

The questionnaire had sixteen questions in five parts.

Background. (1) How long have you been 3D printing? (2) What printer(s) do you own or regularly use? *(multiple)* (3) Most involved hardware modification you've done? (4) Experience with printer firmware?

Multi-axis printing. (5) Were you aware FDM printers can be modified to print in 5 axes? (6) How interested would you be in adding multi-axis capability to a printer you own? *(1—5)* (7) What would motivate you to try it? *(multiple)*

What would hold you back. (8) Biggest concerns about attempting a multi-axis mod? *(up to 3)* (9) Most you'd spend on parts to add 5-axis capability? (10) What resources would you need for a build like this? *(multiple)*

Software and calibration. (11) Would you prefer setup, calibration and G-code tools that run in the browser, as a desktop application, or on the command line? (12) How comfortable are you with printer calibration in general?

Open source and community. (13) How important is it that a project like this is open-source? *(1—5)* (14) Would you be willing to contribute, and how? *(multiple)* (15) Open to a short follow-up interview? *(optional contact)* (16) Anything else you'd want? *(optional free text)*

Respondent profile

The sample was experienced and hands-on. Just under half had been printing one to three years, a third had built a printer from a kit or from scratch, and 42% had edited firmware configuration files (Table D.1). The Ender 3 family was the most common platform, which matches the printers Rep5x targets (Table D.2).

Awareness and interest

84% of respondents already knew multi-axis FDM was possible, and 60% rated their interest 4 or 5 out of 5 (mean 3.8), as shown in Fig. D.1. Both figures reflect a self-selected, already-engaged sample.

Motivations and barriers

The motivations track the advantages the literature attributes to multi-axis FDM: support reduction and otherwise-impossible overhangs lead by a wide margin. Among bar-

Experience	n	%	Most involved modification	n	%
Less than 1 year	13	15	Use it stock	14	16
1—3 years	42	47	Minor mods	18	20
3—5 years	13	15	Moderate mods	10	11
More than 5 years	21	24	Major mods	18	20
			Built from kit/scratch	29	33

TABLE D.1 Respondent experience and most involved hardware modification.

Firmware experience	n	%	Printers owned (multiple)	n	%
Don't know what it is	2	2	Creality Ender 3 series	38	43
Know it, haven't touched it	23	26	Bambu Lab	30	34
Flashed pre-built firmware	15	17	Other Creality (CR-10, K1)	28	31
Edited config files	37	42	Other / self-built	23	26
Written/modified firmware	12	13	Creality Ender 5 series	17	19
			Prusa	14	16
			Voron	12	13
			RatRig	9	10

TABLE D.2 Firmware experience and printers owned.

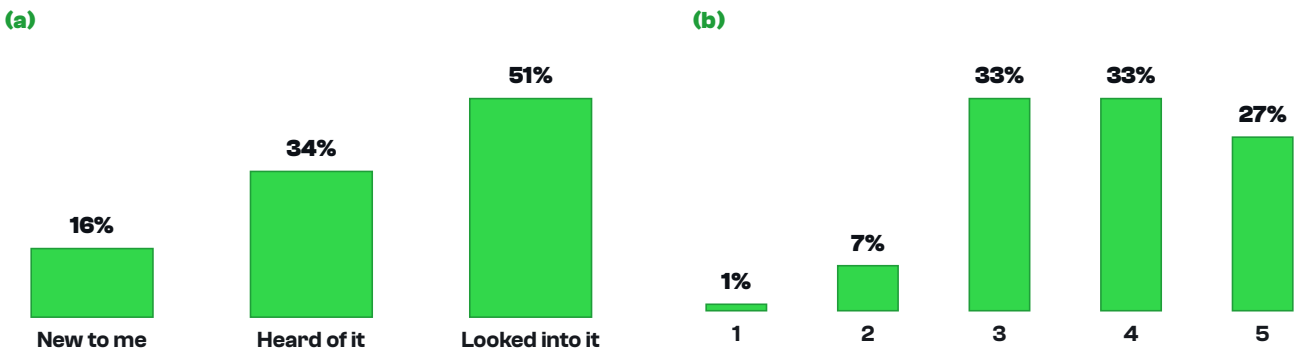


FIG. D.1 (a) Prior awareness that FDM printers can be converted to 5 axes. (b) Interest in adding multi-axis capability, 1 (not interested) to 5 (very interested). N = 89.

Motivation (multiple)	n	%	Biggest concern (up to 3)	n	%
Reduce/eliminate supports	69	78	Cost of parts & electronics	41	46
Impossible overhangs	51	57	Calibrating the extra axes	26	29
Fun challenge	36	40	Specialist/expensive software	25	28
Stronger non-planar parts	33	37	Too little community/support	23	26
Better surface finish	33	37	Missing documentation	22	25
Advancing open-source	33	37	Risk of damaging printer	19	21
Professional/research use	17	19	Mechanical assembly	19	21
Wouldn't be interested	2	2	Unsure what they'd use it for	15	17
			No electronics/wiring skills	11	12
			No concerns, would just try	6	7

TABLE D.3 Motivations for trying multi-axis printing and the biggest concerns about attempting it.

riers, cost is the single biggest, but the combined weight of the toolchain-and-knowledge concerns (software, calibration, documentation, community) is larger (Table D.3).

Cost, resources and software delivery

Willingness to spend centres on €50—200; only 7% would go above €300 (Fig. D.2). Among resources, a written guide with photos is wanted most and interactive browser-based tools least, the result that runs against Rep5x's browser-first decision (discussed in the evaluation

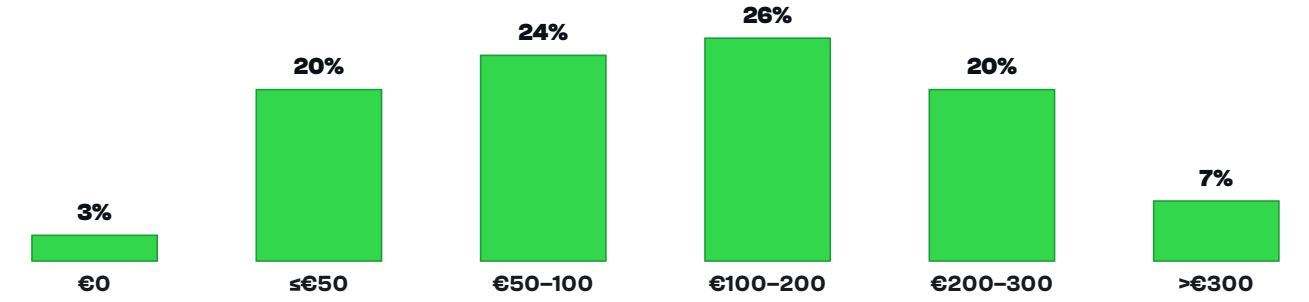


FIG. D.2 Most respondents would spend on parts to add 5-axis capability to a printer they own (N = 89).

Resource needed (multiple)	n	%	Preferred software delivery	n	%
Written step-by-step guide	62	70	Desktop application	31	35
Video walkthrough	48	54	In the browser	25	28
Downloadable 3D models	48	54	Don't care	24	27
Active community	48	54	Command line	9	10
Wiring diagrams/pinouts	43	48			
CAD source files	42	47			
Browser-based setup tools	31	35			

TABLE D.4 Resources makers said they'd need, and preferred software delivery.

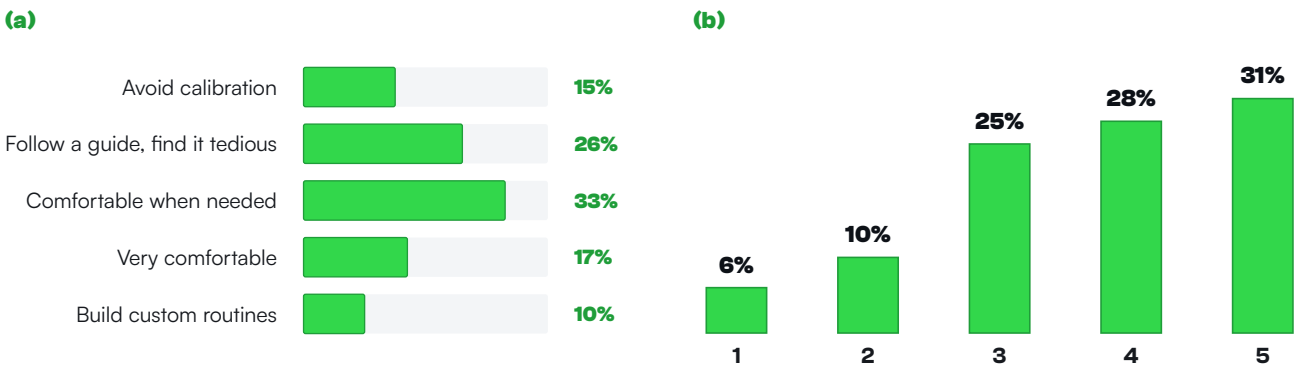


FIG. D.3 (a) General calibration comfort. (b) How important respondents rate open source, 1 (not important) to 5 (essential). N = 89.

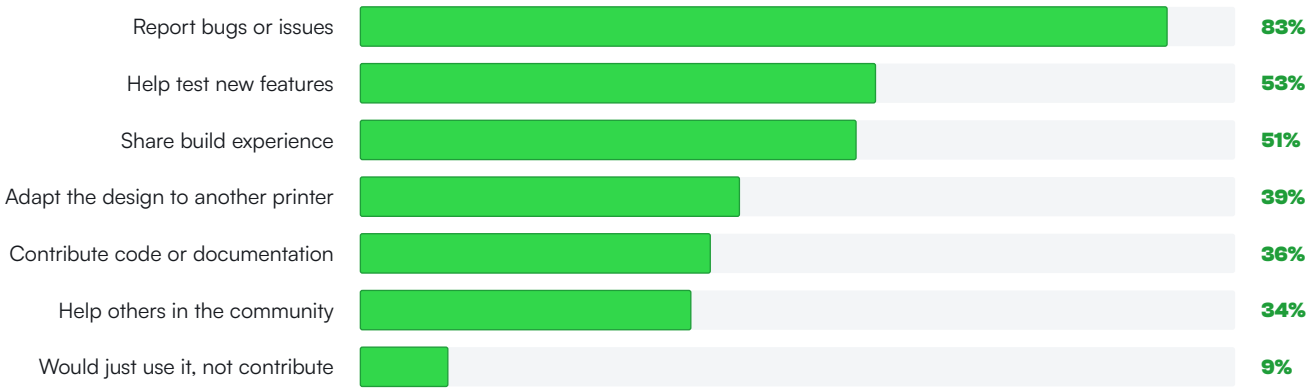


FIG. D.4 How respondents said they'd be willing to contribute to an open-source project like this (multiple selections, N = 89).

chapter). On delivery, desktop is the plurality preference but no option holds a majority (Table D.4).

Calibration comfort, open source and contribution

Calibration anxiety is modest: most respondents are comfortable, and only 15% actively avoid it (Fig. D.3). Open source matters to the sample (mean 3.7 of 5; 59% rated it 4—5), and willingness to contribute is high, only 9% said they'd use the project without giving anything back, though "report bugs" is far easier to tick than "contribute code" (Fig. D.4).

APPENDIX E

E STATEMENT ON THE USE OF AI

During the preparation of this work, I used AI-assisted tools, mainly Grammarly and Anthropic's Claude. Grammarly was used to check grammar and spelling while drafting the report. Claude was used to review drafts for clarity and consistency, and to assist with writing and debugging code for the Rep5x browser tools and the project website, alongside manual development.

After using these tools, I thoroughly reviewed and edited all AI-assisted text and code, taking full responsibility for the final content. The research, design decisions, arguments, and conclusions in this thesis are my own.