Stepdance: A Toolkit for Redesigning Existing CNC Machines Using Physical Metaphors

Ilan Moyer
Massachusetts Institute of Technology
Cambridge, Massachusetts, USA

Emilie Yu
University of California Santa Barbara
Santa Barbara, California, USA

Devon Frost
University of California Santa Barbara
Santa Barbara, California, USA

Maria Yang
Massachusetts Institute of Technology
Cambridge, Massachusetts, USA

Jennifer Jacobs
University of California Santa Barbara
Santa Barbara, California, USA

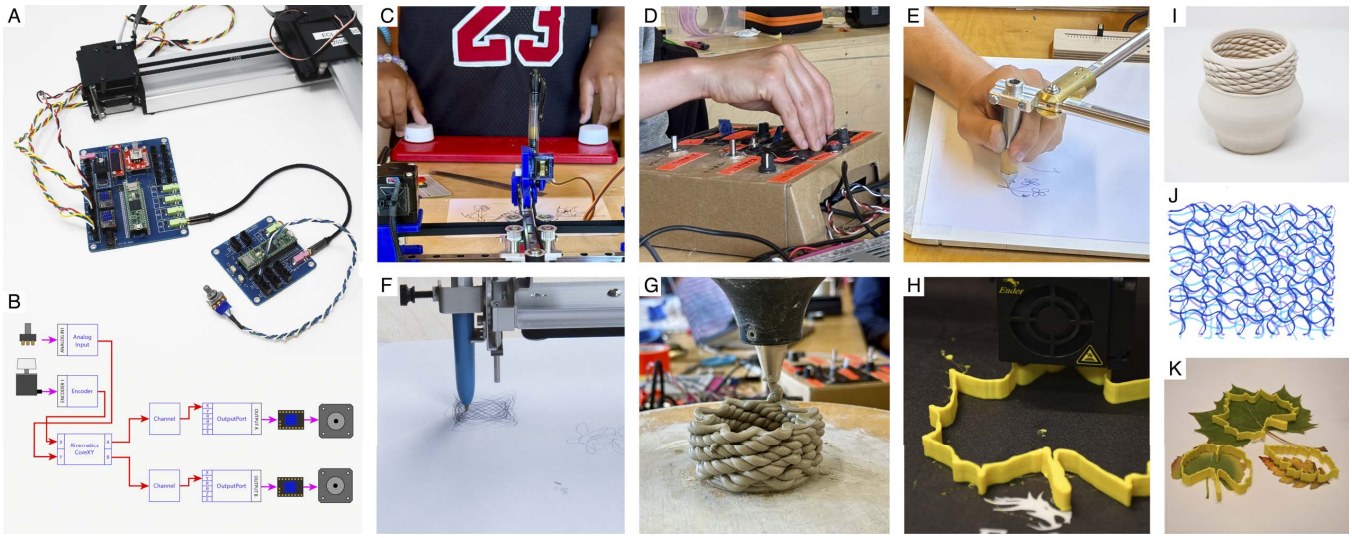

Figure 1: Stepdance is a framework for creative CNC motion control. Stepdance hardware modules (A) replace the GCode controller of existing CNC machines and can be chained together to design new physical interfaces for CNC control. The Stepdance Software Library (B) enables designers to author mappings between physical user interface elements and machine movement. We used Stepdance to rapidly prototype manual user interfaces (C, D, E) to control off-the-shelf plotters (F), clay 3D printers (G), and plastic 3D printers (H). We applied these machines to the production of digitally fabricated artifacts (I, J, K) that blend real-time direct user interaction with automated machining.[1]

Abstract

Researchers can build craft-aligned digital fabrication technologies by designing interfaces inspired by craft tools. This process often demands real-time physical interactions not supported by today's automation-focused CNC control systems. We theorize we can lower engineering challenges for craft-aligned CNC prototyping by allowing designers to modify existing CNCs to support both automated and real-time control. We contribute a new creative motion control system, Stepdance, which consists of two elements: 1) modular controllers that replace the G-code controller of a CNC and can be chained together to develop new interfaces, and 2) a modular programming library that supports declarative mappings between live user input, pre-programmed operations, and machine motion. We developed Stepdance with practitioners at the Haystack Mountain School of Craft, where we used the system to modify commercial plotters and 3D printers. We analyze the resulting artifacts, interactions, and ideas to discuss how Stepdance can broaden the practice of CNC design via physical metaphor.

CCS Concepts

• **Human-centered computing** → **User interface toolkits**.

Keywords

Digital Fabrication, Toolkit, Craft-Aligned CNC, Rapid Prototyping

ACM Reference Format:
Ilan Moyer, Emilie Yu, Devon Frost, Maria Yang, and Jennifer Jacobs. 2026. Stepdance: A Toolkit for Redesigning Existing CNC Machines Using Physical

CHI '26, Barcelona, Spain

ACM ISBN 979-8-4007-2278-3/2026/04
https://doi.org/10.1145/3772318.3791107

[1] Image C courtesy of Haystack Mountain School of Crafts.

Metaphors. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 30 pages. https://doi.org/10.1145/3772318.3791107

1 Introduction

Computer-numerical-control (CNC) machines originated as industrial technology but have since become widely available to individuals and small businesses [16]. While the applications and users of CNC machines have expanded, the workflows for these technologies have remained largely unchanged from their industrial precursors [41]. These standards remain despite practitioners' efforts to establish niche methods [63] and project-specific workflows [23].

Much of the homogeneity in CNC workflows stems from the interaction paradigms used to operate CNC machines. Practitioners can use a range of approaches and representations when digitally designing models for CNC production, but have very few options when fabricating physical artifacts. Nearly all commercial CNC machines require uploading a complete toolpath description, which is then automatically executed by the machine [63]. The operator is limited to intervening when the machine experiences an error or performing minor adjustments to machining parameters.

Researchers can fundamentally reshape digital fabrication by reconceptualizing CNC technologies as extensions of non-digital tools and interfaces. Notable examples include CNCs that resemble a Dremel [70], a manual lathe [60], and a pottery wheel [37]. We categorize this design process as CNC design via a *physical metaphor*, meaning the designer begins with an envisioned control interface and interaction structure informed by an existing physical tool relevant and familiar to their target audience. They then use that metaphor to guide engineering decisions for the CNC technology. The resulting systems combine automated fabrication and machine precision with manual control and live adjustments during fabrication. CNC design via physical metaphor is powerful but difficult for two primary reasons. Designers often must build a CNC machine from scratch [54] or significantly alter the mechanism of an existing machine [40]. Then, designers must develop a control schema that translates high-level user operations into low-level motor-control logic. Collectively, these tasks require expertise in electrical, mechanical, and computer engineering, and it can take weeks or months to produce a prototype of sufficient fidelity to fabricate artifacts. Alternatively, a designer can use an existing CNC machine and a G-code-based control schema [56]; however, this approach limits interaction design to operating within, or attempting to subvert, the established CNC interaction paradigm.

Our objective is to empower designers to radically, rapidly, and playfully re-envision CNC machines by drawing inspiration from physical craft tools. We approach this objective with three key insights. First, the craft tools used as physical metaphors in prior work share qualities of direct manipulation and low latency [34]. Second, *existing CNC mechanisms* are, in and of themselves, not inherently at odds with craft because they stem from manually controlled technologies [41] —e.g., CNC mills and lathes are direct descendants of manual mills and lathes. Third, *established CNC motion control systems*–e.g., G-code interpreters–are a primary limiting factor for alternative CNC interactions. Designers are incentivized to use G-code controllers because they substantially streamline CNC development with built-in motor synchronization and path-planning capabilities, but limit interaction to precompiled operations. We extrapolate from these insights to theorize that we can enable designers to use current CNC machines as platforms for prototyping craft-aligned digital fabrication technologies by providing expressive means to experiment with alternative forms of machine control.

To investigate this theory, we created *Stepdance*, a framework for creative CNC motion control that allows designers to modify existing CNC machines with custom physical control interfaces and real-time interactive machining behaviors. We developed Stepdance based on the concept of manipulating CNC *motion-streams*: signals that reconceptualize the step and direction outputs of G-code-based CNC controllers as a synchronous, general communication scheme. We developed a generic control architecture capable of transmitting and modifying these streams as first-class data.

To use Stepdance, a designer disconnects the stepper motors of a CNC machine from the G-code controller and connects them to the Stepdance Machine Control Module. From there, they can connect any analog or digital physical user interface elements, either directly to the Machine Control Module or to a Stepdance Basic Module board that can be chained to the Machine Control Module. Using the Stepdance Software Library (SSL), they can define the relationship between the user interface and machine behavior by specifying declarative mappings between input values and machine behavior. They can rapidly mix and match multiple interfaces or experiment with different interactions by chaining together multiple Stepdance modules with distinct user interfaces. The Stepdance framework maintains and updates machine state at a rate of 25kHz, using interrupt-driven background processes.

Our approach is different from prior research on interactive fabrication and modular machine toolkits. Unlike prior interactive fabrication research, which focuses on presenting a single new machine or interaction, we contribute a toolkit for rapid prototyping a wide range of interactive CNC machines. General-purpose physical computing toolkits [19] lack methods for motor synchronization, requiring extensive end-user development for coordinating multiple axes of machine movement. Modular CNC toolkits are purpose-built for CNC motion control and enable rapid prototyping of different CNC architectures by allowing the designer to specify the number of axes [42] or the kinds of end effectors [65], but most still rely on G-code-based controllers.

We informed the design of Stepdance by developing and testing the system during a residency at Haystack Mountain School of Crafts. We demonstrate Stepdance's capabilities by presenting a series of modified CNC machines using a pen plotter, a clay 3D printer, and a plastic 3D printer, and by presenting artifacts produced by the research team and community members of the School using these machines. We analyze the results of our design process to discuss the benefits of direct digital fabrication in craft environments, new conceptions of modular machine design, and how prioritizing physical interfaces in CNC rapid prototyping can yield new insights for digital fabrication development.

We make the following contributions:

(1) The Stepdance modules– a modular controller hardware system chainable via audio cables that enables interactive

fabrication with existing CNC technologies. The modules include the machine controller module- a board with on-board stepper drivers that designers can use to replace the G-code controller of an existing 3D printer or other CNC, and the Stepdance Basic Board- a paired down version of the machine controller module that can read the inputs of a physical user interface and stream that data to a machine controller module or other basic module.

(2) The Stepdance Software Library– a dataflow programming library for authoring CNC control interactions that integrate real-time manual operation with automated machine operation. SSL enables the generation of motion streams and the specification of logic to map motion stream inputs and peripherals to output channels.

(3) A demonstration of the capabilities of our approach through a series of novel CNC machines developed that extend interactions from existing physical tools, including a mixer, a pen-based pantograph, and a pottery wheel.

2 Related Work

2.1 CNC machine hardware development and modification

HCI researchers have used art and craft tools as physical metaphors for building new CNC machines. In some cases, researchers develop handheld tools with digital capabilities to reduce risk while maintaining manual expressiveness for a given craft, such as carving [50, 70] or airbrushing [54]. In other instances, researchers seek to apply interactions and skillsets from manual craft to CNC machines, such as lathes [60] and clay 3D printers [37, 39]. These technologies are, in part, examples of interactive fabrication [67], a craft-inspired interaction paradigm in which creators iteratively determine design outcomes during fabrication. We seek to support interactive and craft-inspired CNC prototyping akin to the tools presented above, without building a novel CNC mechanism.

An alternative approach is to modify existing CNC machines to support new interactions. For example, He and Adar modified a plotter for punch-needle embroidery [22], Rivera and Hudson altered a consumer 3D printer for electro-spinning [49], and Lazaro Vasquez et al. repurposed the mechanical housing architecture and controller boards for desktop 3D printing for biofiber spinning [30]. These works demonstrate how machine hacking can replicate and extend methods that usually require industrial equipment. Researchers have also hacked machines to support new CNC workflows. Kaplan et al. control a clay 3D printer with a game controller [26]. Subbaraman and Peek modify a plastic 3D printer with a MIDI controller, enabling the designer to experiment with and document parameters during printing [56]. Kim et al. draw on music production to prototype "compositional" 3D-printing interactions in which creators iteratively design through drawing, scanning, and gestural input [27]. Gui et al. also use drawing as input for CNC milling; they extend a CNC mill with a scanner to detect annotations on the stock [20]. These works demonstrate that CNC modification is a rich area for HCI digital fabrication research; however, except for the punch needle plotter, they all rely on pre-existing G-code interpreters. For example, Subarraman, Peek, and Kim et al. support operator modification during printing by chunking a print into smaller parts, enabling the operator to overwrite chunks that the machine has not yet executed. Chunking creates a tradeoff between the size of the chunk and the degree of operator control. Smaller chunks enable more continuous operator changes but cause machine stutter because the CNC must repeatedly accelerate and decelerate rapidly during very small motions. Larger chunks (e.g., an entire layer of a print) reduce stutter but also substantially reduce responsive real-time control. We build on our prior work to interactively control a custom polar mechanism for digital pottery [37] to bypass G-code, and instead use motion channels whose target position value can be continuously updated by multiple inputs. Unlike the control system from the Digital Pottery Wheel, however, Stepdance is compatible with a wide range of CNC machine architectures.

Robot arms are another popular prototyping platform for integrating automated and manual fabrication. Robot arm-based systems can aid operators in precise manual tool application [59], or enable iterative fabrication in tandem with augmented reality interfaces [35, 43]. Researchers have also enabled operators to control robot arm behavior [15] and fabrication toolpaths [3] through gestures. These methods enable a skilled operator to manually specify machining parameters. While robot arms are remarkably versatile, they are significantly more expensive than desktop CNCs and require the installation of fabrication end-effectors for machining. Stepdance enables the responsiveness and manual control typically afforded by robot arms, while prototyping with less expensive, more common, fabrication-ready CNCs.

2.2 Rapid CNC Prototyping

Studies of real-world CNC use show that artists, designers, and other practitioners regularly circumvent intended CNC machine workflows [63], modify their machines to support a desired workflow [29], and engage in regular maintenance, which can help operators develop craft skills with their machines [58]. Closed CNC control structures hinder practitioner modification by restricting access to machine-state data and exhibit limited modification capabilities [32].

Designers can build custom CNCs through a variety of off-the-shelf platforms. General-purpose physical computing toolkits lower barriers to driving motors in response to analog or digital input components through streamlined hardware modules, software simulation, and learner-oriented software development environments. Notable examples include Arduino, which provides a development environment and breakout board for Atmega microcontrollers [1], and the Phidgets physical interaction design prototyping toolkit, which provides designers with physical input and output modules that emulate software widgets [19]. Similar to Phidgets, the Jacdac prototyping platform provides software device twins for physical modules, and designers can program modules with popular creative coding environments [2]. We share the same broad goal as these toolkits: lowering barriers to physical interaction design; however, general-purpose physical computing platforms do not address key barriers in CNC development. While it is technically feasible to prototype a CNC machine using any toolkit that enables motor control, current toolkits lack purpose-built hardware components and

software abstractions for motor *synchronization*. As a result, building a CNC with a baseline Arduino toolkit requires the designer to implement methods to coordinate the movement of multiple axes to achieve the desired motion. Systems that use PC-software communication to drive motor behavior, like Jacdac and Phidgets, are challenging to use for precise, responsive CNC movement because they introduce latency between command execution and motor movement and lack provisions for synchronized motion. By developing Stepdance as a real-time framework running on an embedded system, we ensure predictable motor control with extremely low latency and high synchronicity for both pre-programmed and manual input.

Many CNC designers instead opt for purpose-built off-the-shelf CNC machine controller systems that provide hardware inputs for limit switches and software support for coordinated motion control and popular machine kinematics. For example, the extensible Duet3D [9] provides generalizable hardware and firmware for G-code-based motion control. Duet uses the RepRap firmware – developed as part of the broader RepRap project, a community effort to create open-source printer designs that are self-replicating [6]. Other popular firmwares include Marlin [33], GRBL [18], and Klipper [28], which offer different tradeoffs between calculation speed and microcontroller compatibility. All of the above platforms use G-code-based motion control. While individual features vary, they reinforce the broader paradigm of controlling CNCs through precompiled toolpaths. Developers using these systems benefit from a prototyping workflow that leverages established CNC control pipelines for homing, movement feeds and speeds across three or more axes, and for translating user toolpaths into machine kinematics. The tradeoff is that user operations are limited to defining precompiled machining parameters. In contrast, Stepdance enables the designer to develop custom CNC operation paradigms that include both real-time and pre-programmed behavior, while providing methods that automatically manage motor synchronization, velocity limits, and kinematic transformations.

To broaden machine building, researchers have developed toolkits for rapid prototyping of CNC mechanisms. The Cardboard Machine Toolkit [42] provides a design template for modular axes and a networked control system [38]. Fossdal et al developed a parametric software system for designing CNC motion platforms [11, 12]. Vasquez et al contribute a platform for multi-end effector CNC machines via a dynamic tool-changing mechanism [65]. Except for the Cardboard Machine Toolkit, which uses PyGestalt (discussed below), the above toolkits rely on off-the-shelf G-code controllers. We see Stepdance and toolkits for rapid prototyping of CNC mechanisms as mutually supportive — Stepdance is compatible with mechanisms produced by the above toolkits, and together they could enable rapid prototyping of both mechanisms and motion-control behavior.

There are a few alternatives to using G-code controllers. pyGestalt virtualizes CNC hardware functionality within a host computer environment. This approach provides flexibility but introduces latency because communication and processing flows through a desktop operating system [38]. Modular Things [46] is a modular, self-registering hardware system for machine motion control. The system uses a USB-based communication protocol and is not yet capable of supporting real-time, synchronized applications due to latency issues. Urumbu [17] uses multi-threading and small USB packet sizes to address synchronization, but may introduce challenges for real-time responsiveness due to round-trip latency. In contrast, Stepdance enables rapid composition of real-time interactions by enacting software modularity at the firmware level and connecting physical modules using synchronous motion streams.

Most similar to our work is MAXL [47], which supports operating on and synchronizing multiple machine component trajectories. Stepdance also manipulates multiple motion-control trajectories, but does so in real time on a fully embedded, custom hardware control system using step and direction signals. In contrast, MAXL primarily operates on the PC using abstract representations to specify motion, which the computer then transmits to modular control boards for execution.

Finally, researchers have developed programming toolkits that support designers in authoring custom CAD-CAM-CNC workflows. Tools like Vesipdae [13] and Dynamic Toolchains [64] use dataflow paradigms that enable designers to sequence CAD and CAM design operations, subroutines, and external data to prototype experimental workflows across different CNC technologies rapidly. Millipath [4] and Tandem [62] support custom workflows for CNC milling using imperative and computational notebook programming paradigms. Like the Stepdance programming library, these systems employ a modular representation of machine operations; however, they primarily enable designers to control workflows up to the point of machine execution, not during it.

Fossdal et al and Subbaraman and Peek extend this approach by augmenting dataflow [10] and imperative creative coding [57] environments to support interactive workflows through iterative transmission of G-code to controllers. These approaches support custom CAD and CAM workflows with a degree of interactive control, while inheriting the limitations of G-code chunking. Stepdance focuses primarily on prototyping design methods that occur during machining. We see opportunities to integrate our approach into authoring tools that span CAD, CAM, and real-time operations.

3 Methodology

To align our work with the practices of contemporary craftspeople, we conducted our research within a craft residency. In this section, we provide a motivating example for CNC prototyping drawn from the residency. We then detail our methodology and design principles.

3.1 Motivating Example

A ceramics instructor is teaching a workshop at a craft school based on his method. He decorates manually thrown vessels using cookie cutters to impress designs in clay bodies, which he later colors with slip and underglaze. The ceramics instructor has his own cookie cutters for students to use; however, he doesn't want them to be limited to this selection. He knows 3D printers can create plastic parts, and the school has a fab lab. The instructor consults with digital fabrication experts working in the fab lab about the possibility of students designing and printing their own stencils. The experts explain that this requires CAD modeling and CAM slicing. Most students lack CAD or digital fabrication experience and wish to focus on learning ceramics. The digital fabrication

Figure 2: Ceramic artist and instructor Masa Sasaki's workflow for creating ceramic surface decoration with stencils. A) Sasaki repurposes cookie cutters and other stencils for surface decoration.[2] B) He instructs students on how to press stencils into leather-hard clay vessels and hand-paint them with slips and underglaze. C) A sample of finished glazed artifacts by students in Sasaki's workshop.

experts wonder if they could redesign the 3D printer to enable students to produce custom stencils without learning 3D-printing software workflows.

3.2 Design Methodology

We chose to design and validate Stepdance during a residency at Haystack Mountain School of Crafts. The motivating example we described above was a real interaction between the authors and the artist and ceramics instructor Masa Sasaki[3] at the School.

Haystack is an international craft school located in Deer Isle, Maine[4]. Founded in 1950 as a research and studio program in the arts, Haystack offers studio craft workshops for participants of all skill levels, during which visiting faculty teach. The School's facilities comprise physical workshops for ceramics, wood, metal, glass, fibers, and graphics. The School also has a Fab Lab, which houses 3D printers, laser cutters, and other CNC machines. The Haystack organizers invited the authors to participate in a two-week residency in the Fab Lab, which overlapped with craft workshops.

We had three objectives for developing Stepdance during our Haystack Residency. First, we sought to better understand how craft practitioners perceive the benefits and limitations of digital fabrication. Second, we aimed to conceptualize potential applications of Stepdance in response to real ideas and challenges faced by craftspeople. Third, we sought to validate Stepdance for rapid CNC prototyping by using it to modify existing CNC machines during the residency and have craftspeople interact with the modified machines.

Preparing for and conducting the residency spanned six months. Prototyping the physical Stepdance controllers took prior to the residency. We prototyped v0 of the Stepdance machine controller and used it to replace a plotter's SVG interpreter. We also manufactured and assembled three Stepdance Machine Controllers and four Stepdance Basic Modules and verified their baseline compatibility with a clay 3D printer. We wrote a barebones v0 of the SSL to support these tasks. In tandem, we discussed our plans with the Haystack staff and instructors and brainstormed possible interactions from these conversations.

We brought an Axidraw pen plotter, a Potterbot Micro Clay 3D printer, and a collection of potentiometers, encoders, and other input components to the residency. The majority of Stepdance design and engineering took place during the residency. In the first week, we used the process of connecting Stepdance to the plotter and clay 3D printer to inform the development of SSL v1.0. We presented our research objectives to the Haystack community on the second day and invited them to visit the Fab Lab to propose ideas. By the middle of the second week, we completed v1.0 of the SSL and engineered multiple interfaces for controlling the plotter and the 3D printer. We invited the Haystack community to test these prototypes and create sample artifacts over two days. Following the residency, we conducted a 1-week prototyping session with a plastic 3D printer in our lab based on ideas from the residency that we did not have time to execute on-site.

We collected data through written notes, photographs, and video recordings of participants using the tools. We collected artifacts produced by participants with the Stepdance-modified clay 3D printer and Axidraw. Broadly, our methodology follows a research-through-design process [69], in which we systematically develop functional artifacts to challenge the status quo in CNC machine development workflows and interaction paradigms. We also build on the model of HCI craft research through residencies, an established method for technical systems development [61]. Residencies provide the means to incorporate technical knowledge outside mainstream engineering [7] and simultaneously examine the practical and cultural implications of digital fabrication technology [55]. The authors conducted the vast majority of technology prototyping in response to ideas that emerged during discussions with craft practitioners. Craft practitioners then used the resulting tools to produce artifacts, identify limitations, and pose questions, facilitating successive iterations.

[2]Image courtesy of Masa Sakaki. Source: instagram.com/masasasakiceramics.
[3]https://www.masasasakiceramics.com
[4]https://www.haystack-mtn.org

3.3 Design Goals

We formulated four design goals (DG) to guide our development of Stepdance. Our goals reflect two interrelated concerns: our theory that craft-aligned CNC interaction benefits from low-latency control, and our ultimate goal of supporting CNC prototyping via physical metaphor by intertwining craft production with CNC rapid prototyping. Below, we describe each goal and provide context in craft theory, interaction design, and prior systems.

(1) **Rapid:** The system should enable rapid prototyping of functional CNC technologies with novel control interfaces. By rapid, we mean hours to days. By functional, we refer to prototypes that satisfy high degrees of both implementation and look-and-feel [24]. A high level of implementation in digital fabrication means a CNC machine capable of fabricating a physical part. A high degree of look-and-feel translates to an interface that enables operators to specify a part design.
(2) **Real-time:** The system should support real-time operation by a human. Traditional craft tools are characterized by direct manipulation, in which the operator can instantaneously and continuously perceive material behavior in response to manual tool operation [34]. We seek to enable CNC machines that offer comparable real-time machining feedback in response to incremental and continuous operator interaction.
(3) **Re-configurable:** Designers should be able to iterate and produce variations of a CNC interaction schema rapidly. Producing multiple prototypes enables designers to gauge the design space better and helps users understand constraints and provide meaningful feedback [21]. We aim to enable designers to experiment with control interfaces across various machines and to test interface variations on the same machine.
(4) **Hybrid:** The system should support blending real-time inputs and manipulation with pre-planned operations. HCI Hybrid craft has emphasized the productive unification of human and digital capabilities to resolve conflicts between authentic craft and automated production [8]. We see utility in automated machining but also seek to challenge the supremacy of the conventional CNC control paradigm. We therefore prioritize prototyping interactions that enable designing *on-tool* while remaining compatible with preprogrammed operations (e.g., G-code).

4 Stepdance System

The Stepdance system consists of Stepdance Module printed circuit boards (PCBs) used to control existing machines or build real-time interface modules, and a Stepdance software library (SSL) that supports the construction of real-time embedded motion control systems through a modular, dataflow-like paradigm. To support future research, we have published Stepdance hardware schematics and the SSL code under an open-source non-commercial license at stepdance.org.

4.1 Stepdance Modules

Stepdance modules (Fig. 3) are custom PCBs that contain a Teensy 4.1 microcontroller [44] running the Stepdance library, ports for intermodule communication, and provisions for interfacing with machine hardware, physical UI elements, and sensors. We designed two types of modules for different applications: *Machine Controller Modules* interface with CNC hardware, while *Basic Modules* can be incorporated into modular user interfaces.

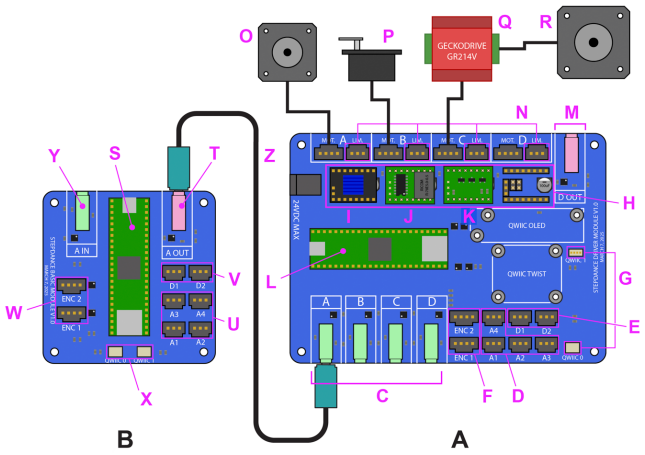

Figure 3: Stepdance PCB Modules. The Machine Controller Module (A) serves as both a platform for driving CNC motors and for rapid prototyping of physical CNC interfaces. It includes four barrel jack input ports (C) and one output port (M); inputs for digital (E), analog (F), and encoder (D) peripherals; and adaptable ports (H) for driving different kinds of motors (O-R) through on-board stepper drivers (I) servo drivers (J) or breakouts for external drivers (K). The board is controlled by a Teensy 4.1 microcontroller (L) and includes two Qwiic connectors (G) for wiring an external display and an input knob, or for additional peripheral input. The Basic Module (B) is a pared-down version of the Machine Controller that supports rapid prototyping of modular physical interfaces that can be chained together or connected to the machine controller (Z) via the input (Y) and output (T) ports. It contains four analog (U), two digital (V), and two encoder (W) inputs, two Qwiic connectors (X), and is also controlled by a Teensy 4.1 (S).

4.1.1 Machine Controller. As a CNC-specific control system, the Stepdance Machine Controller (Fig. 3A) shares some features common to GCode controllers, including limit switch inputs (Fig. 3N) and motor drivers. These features enable powering and driving a CNC machine without adding external control boards or sensing hardware. The Machine Controller deviates from prior CNC controllers by including *input* elements that enable a wide variety of interface components and data sources to drive machine behavior. (Fig. 3I). The module contains four *analog inputs* (Fig. 3D) for analog sensors or controls like potentiometers, four *quadrature encoder inputs* (Fig. 3F), and two *digital I/O* inputs for attaching buttons (Fig. 3E). (The analog inputs can also double as digital inputs.) While the Machine Controller can drive a CNC without any additional peripherals, we enable modular development through four Stepdance *motion stream input ports* (Fig. 3C) that can receive motion streams from other modules and process them to drive machine

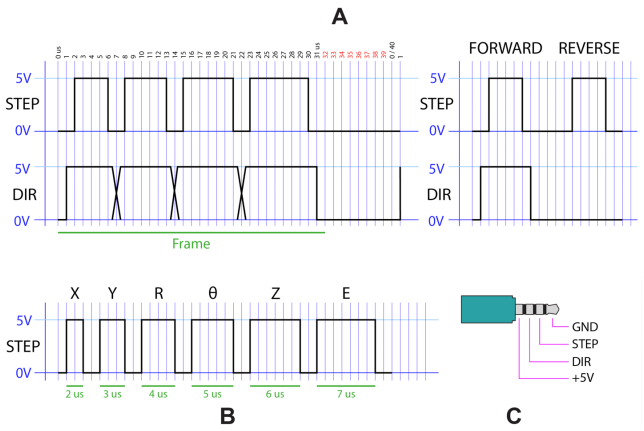

Figure 4: Stepdance modules transmit and receive motion streams using pulse-width multiplexed step and direction signals. A) Up to four axes can be transmitted synchronously within a 40μs-long frame. B) Six distinct axes can be encoded. C) Signals are carried over standard 3.5mm TRRS audio cables.

behavior. We also provide one *motion stream output port* that can be chained to downstream Machine Controller modules, expanding the number of machine axes the system can control if necessary. We describe motion stream port functionality in greater detail in sections 4.1.3 and 4.2.

CNC mechanisms exhibit a variety of control requirements. To make Stepdance compatible with a wide range of motors, we adopt and extend an existing modular driver socket system [52] (Fig. 3I) capable of driving a range of stepper motors. To support modification of machines that use servo motors or large steppers, we created additional compatible modules, including the *hobby servo drive module* (Fig. 3J) and the *external drive module* (Fig. 3K). As a result, the Stepdance Machine Controller supports a range of common actuators across both small and large CNC machines (3O-R).

4.1.2 Basic Module. While the Machine Controller can be self-contained, more complex CNC interfaces may benefit from modularization through dedicated hardware. We sought to support design patterns that separate specific interface control logic from the machine behavior logic, enabling repurposable, agnostic control interfaces across different CNC machines. To support this, we designed the *Basic Module* (Fig. 3B)– a smaller, pared-down module with components for connecting interface elements and chaining with other modules. Designers can connect the Basic Module to both upstream and downstream modules through one Stepdance input port (Fig. 3Y) and one Stepdance output port (Fig. 3T). This structure allows the module to accept motion streams from another module, modify them based on interactions with its interface components, and pass them to a downstream module. Like the machine controller, the basic module also includes a range of perhperal input elements including four analog inputs (Fig. 3U) doubling as digital I/Os, two dedicated digital I/O (Fig. 3V) and two quadrature encoder input ports (Fig. 3W).

4.1.3 Module Interconnect. As mentioned in the previous sections, Stepdance modules communicate with each other to enable rapid reconfiguration with multiple interface elements and control logics. We avoid dominant CNC control abstraction schemas like G-code because they represent machine movement exclusively as an absolute, precompiled set of motion commands, and cannot be modified in real time without high latency or kinematic disruption. Instead, our communication schema uses synchronous *motion streams*. Motion streams reconceptualize the step and direction pulses commonly used to control CNC machine motors as synchronous first-class data types that designers can read and modify. With this approach, we can generate, modify, and mix motion streams with control logic specified by physical inputs, procedural operations, and predefined toolpaths– all in real time without machine disruption. Our use of motion streams builds upon our prior work building a polar coordinate 3D printer [37]. In this prior work, we found motion streams to be powerful for interactive machine control, however we relied on modular hardware system that allowed only one motion axis to be transmitted over an input/output port. We found that wiring a large number of motion stream inputs correctly across multiple modules proved difficult and caused significant issues during prototyping. We address this hardware issue with Stepdance by enabling *up to four* axes of motion to be transmitted over a single cable, thereby significantly reducing the system's physical wiring complexity. We accomplish this by modulating the duration of each step pulse based on the axis it represents, and transmitting up to four of these pulses within a 40μS long frame, enabling multi-axis step rates of up to 25 kHz. Figure 4 illustrates our encoding scheme. Up to four axes can be transmitted within a frame (Fig. 4A), and our scheme allows up to six axis encodings (Fig. 4B), providing flexibility in how a module might address machines with 4+ motion axes. We provide details on motion stream control logic in section 4.2.

4.1.4 Module Alignment with Design Goals. We enable **rapid** CNC prototyping through a self-contained Machine Controller that designers can wire with manual input components and adapt to different motors on existing CNC machines (DG1). We support **reconfigurable** development approaches through a simplified Basic Module that supports mixing control inputs across different purpose-built user interfaces (DG3). We support **real-time** operation by extending our prior motion stream control schema to maintain low-latency adjustment of motor behavior while reducing physical prototyping complexity (DG2).

4.2 Stepdance Software Library

In addition to modular hardware, physical computing toolkits rely on software building blocks that correspond to hardware functionality and afford a *path of least resistance* to the development of interactive applications for a given domain [31]. For craft-aligned CNC interaction prototyping, we augment the Stepdance hardware modules with a software library (SSL) that provides abstractions for authoring firmware to manipulate motion streams. In this subsection, we describe the SSL underlying architecture that ensures precision and low latency, before detailing end-user programming abstractions for real-time *transmission*, *generation*, and *modification*

of motion streams. Table 1 provides a high-level overview of all components and functionality.

4.2.1 Core Architecture. Within the SSL, motion streams consist of positional information, internally represented as signals that can be transmitted internally between *SSL components* or externally across hardware modules. SSL motion streams provide a flexible software representation for designating input to the corresponding electrical signaling motion stream schema for the hardware modules (described above in section 4.1.3. Motion streams can be conveyed either incrementally as relative changes in position each interrupt frame or as absolute values. To optimize Stepdance performance in response to user inputs and changes in motion streams, most SSL components execute sequentially within an *interrupt frame* that runs at 25kHz, synchronously with the Stepdance physical input and output ports. This ensures low latency and eliminates the need to perform interpolation when generating output motion streams. We perform internal motion calculations incrementally, utilizing 64-bit floating-point numbers to avoid the accumulation of significant rounding errors, which is vital to retain precision in the actual machining output. For further details on component execution implementation, see Appendix A.1.

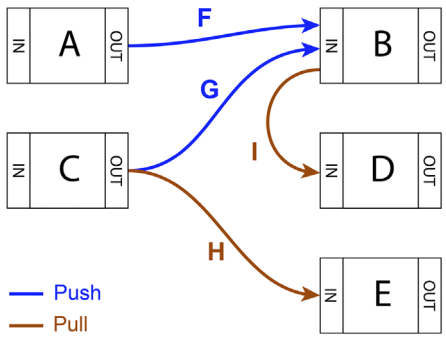

Figure 5: Stepdance enables declarative mapping of motion streams between components using BlockPorts: standardized input and output interfaces. Multiple outputs can map to a single input (F,G) and output or input BlockPorts can read from other BlockPorts (I,H)

Declarative Component Mapping Through BlockPorts. The SSL contains scaffolding infrastructure and collections of modular software components that designers can connect through a modified dataflow approach. To support automatic transmission of motion streams between SSL components, we developed *BlockPorts*: code objects that serve as standardized input and output interfaces. Designers using the SSL do not directly initialize BlockPort objects but instead initialize higher-level components that make use of them as inputs and outputs. Figure 5 illustrates typical "wiring" patterns. Components A thru E have inputs `A_in` thru `E_in` and outputs `A_out` thru `E_out`, all implemented as BlockPort objects. A module author would connect `A_out` to `B_in` by calling `A_out.map(&B_in)` (Fig. 5F). Multiple outputs can map to a single input, for example, with `C_out.map(&B_in)` (Fig. 5G). Internally, each BlockPort has position buffers accessible to the parent component and to mapped BlockPorts. On each interrupt frame, a component reads its input BlockPorts' position buffers, executes internal calculations on that data, stores the results in its output BlockPorts, and then calls `BlockPort.push()` on each output, which sends a motion stream from the BlockPort to its mapped target's position buffers. By default, this signal is sent incrementally, allowing multiple outputs to affect a common downstream input additively. However, in special cases, a user may configure a mapping to transmit absolute positions. Our implementation of BlockPorts is distinct from software-based dataflow systems. Because Stepdance programs must run entirely in embedded hardware, we have substantial memory and performance limitations not present in desktop systems, making arbitrary pointer storage and reference calls expensive. We account for this by limiting BlockPorts to only one target of their `map(&target)` function. We enable many-to-many mappings by allowing inputs to be mapped to outputs or other inputs.

4.2.2 Motion Stream Transmission. The SSL enables motion stream transmission across modules and to CNC machine motors through interrupt-driven software components that correspond to physical hardware elements. All transmission modules are optimized to ensure low latency and precise transmission of positional data during continuous real-time modification.

OutputPorts. Each physical output port can be associated with a matching OutputPort software component, which contains step and direction flags for each of six output signal types (X, Y, Z, E, R, and T). OutputPort components execute at the end of the interrupt frame, and synthesize an electrical motion stream in the form of a multiplexed 40μs-long step/direction frame on the physical output port (Fig. 4). Designers can also use OutputPorts to control motor drive modules by synthesizing a single signal per frame, as is done with the Stepdance Driver Module.

Channels. Channels manage the machine's positional state, ensure motion streams stay within velocity limits, and directly interface with OutputPort components by mapping to a particular signal flag (e.g., "X"). Internally, channels operate in fractional steps, where each step corresponds to a step pulse on the output port. Channels contain a target position variable that upstream components can modify. They also contain a variable representing their current position. Channels automatically manage positional updates driven by upstream inputs. In each interrupt frame, the channel increments or decrements the current position when needed to drive $|target - current| \leq 0.5$, and, correspondingly, sets the step and direction flags on its mapped OutputPort. Any number of channels can be instantiated, often corresponding to axes of a machine.

InputPorts. Each physical input port can be associated with a corresponding InputPort component. InputPorts receive multiplexed motion streams on physical Stepdance input ports, and map these signals to downstream components. To reduce latency, receiving and decoding of input streams occur asynchronously using low-level hardware peripherals, and the results are processed during the interrupt frame.

4.2.3 Motion Stream Modification. To enable live mapping of user interaction to machine behavior, the SSL includes methods for

Component	Functionality
Core Architecture	
BlockPort	Internal motion stream conduit. Maps signals between SSL components for automatic transmission of motion streams. Not directly instantiated by the developer.
External Motion Stream Interfaces	
OutputPort	Synthesizes motion stream signal to transmit to motor drivers or other Stepdance Modules.
Channel	Intermediary between all other SSL components and OutputPorts. Maintains machine positional state and limits velocity.
InputPort	Receives multiplexed input motion streams from other modules and maps signals to downstream components.
Motion Stream Modification	
Inputs (AnalogInput, DigitalInput, Encoder)	Software interface for physical input ports on Stepdance SSL.
Filters (Scaling 1D/2D)	Continuously transforms input motion streams based on predefined mathematical functions.
Kinematics (CoreXY, PolarToCartesian/Lever, FiveBar)	Performs coordinate transforms between the interface coordinate system to mechanism coordinates.
Motion Stream Generation	
Generators (Velocity, Position, Wave1D, PathLength 1D/2D)	Synthesize motion streams using simple mathematical functions.
Interfaces (GCodeInterface, Eibotboard)	Enables automated operation by converting common pre-complied motion trajectories to motion streams.
Recording (FourTrackRecorder, FourTrackPlayer)	Enables recording and playback of motion streams produced through any other component.

Table 1: Stepdance Software Library primary components organized across core architecture, motion stream transmission, modification, and generation.

transforming motion streams through physical inputs, filters, and coordinate transformations.

Inputs. We include three input component types– AnalogInput, DigitalInput, and Encoder– that correspond with the peripheral ports on the Stepdance hardware modules and enable designers to map the values generated by interface elements on these ports to other SSL components. Designers can directly map input readings to control parameters rather than imperatively specify read-and-update logic. (e.g., an AnalogInput can be mapped to the amplitude parameter of a sine wave generator. For further details on Input implementation, see Appendix A.2.

Filters. To support continuous adjustment of motion streams, similar to media dataflow languages that filter audio data, we provide filter components that accept a motion stream as input, perform a mathematical transformation, and pass the modified stream as output. At present, we implement 1D and 2D ScalingFilters, which scale input motion streams based on a scaling factor control parameter. Future filter components could perform skewing and rotational transformations.

Kinematics. CNC machines often feature mechanisms that are precise and efficient, but that enforce non-intuitive or inconvenient coordinate systems from a control perspective. Kinematics are critical abstractions for CNC control that translate between the desired tool position in the designer's coordinate space to the necessary movements for a target mechanism. We incorporate kinematics within the SSL as components that transform motion streams from one set of working coordinates to another, to support common mechanisms. For example, the CoreXY component converts from XY space into differential actuator motion ($\Delta A = \Delta X + \Delta Y$, $\Delta B = \Delta X - \Delta Y$). Designers can also use SSL kinematics to translate across multiple coordinate spaces in a single workflow to perform different interaction operations more easily. The PolarToCartesian component converts from rotational space ($R\theta$) into XY space. To generate an ellipsoid motion path for a CoreXY plotter, an SSL program could first transform the X output of a PolarToCartesian component using a 1D scaling filter, then map the filter's X output and the PolarToCartesian component's Y output to a CoreXY component.

4.2.4 Motion Stream Generation. Stepdance is premised on the notion that CNC interaction at the intersection of skilled manual input and automated machining is underexplored. We therefore integrate methods for manual manipulation of motion streams with components for automated motion stream generation.

Generators. Generators are SSL components that synthesize motion streams using low-level mathematical functions based on control parameters that designers can update in real time. For example, the 1DWaveGenerator produces a sine wave with amplitude, frequency, and phase shift. Frequency and phase shift can be synchronized to an input motion stream, enabling a certain number of waves per revolution of a rotary axis. Alternatively, the PathLengthGenerator2D synthesizes a motion stream proportional to the Cartesian distance traversed by the two input motion streams. Designers can use path length generators to control extruders on the fly when the extrusion distance is not known in advance. For a list of all Generators see table 1.

Interpreters. While generators support basic motion control with simple functions, they are not well-suited to automated movement that follows a predefined form or irregular trajectory. To support such pre-planned motion paths without sacrificing real-time control, we provide *Interpreter* components that synthesize motion

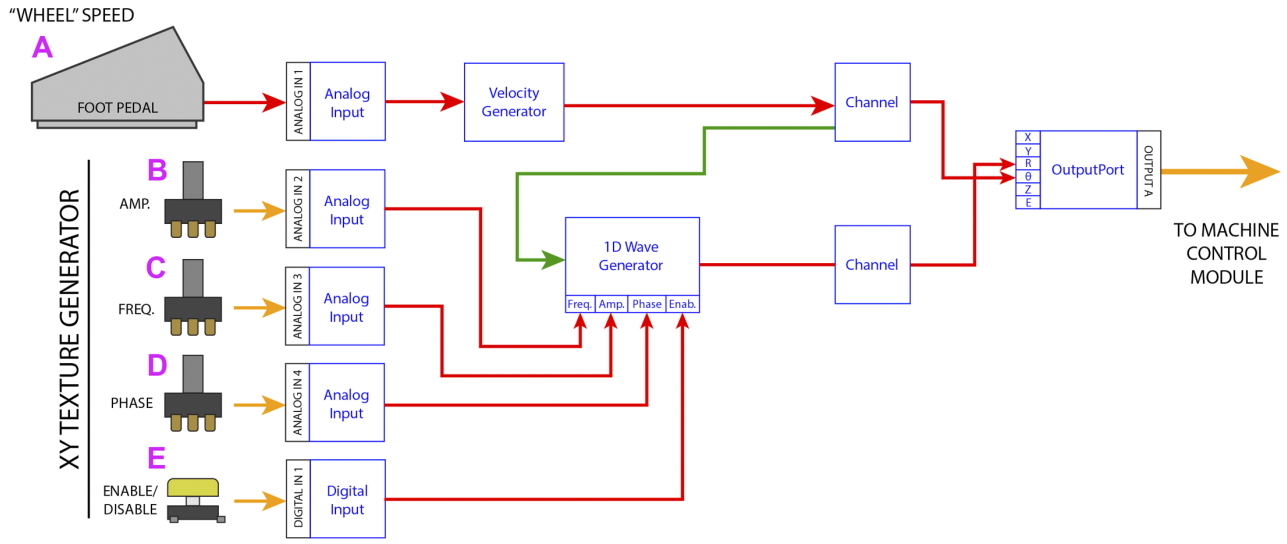

Figure 6: Mixer Module Firmware Architecture

streams from existing motion control protocols. We implemented an Eibotboard interpreter that accepts the same commands as the Axidraw pen plotter, enabling SVG drawings as input to Stepdance. Similarly, we implemented a GCode Interpreter that accepts standard GCode commands sent via serial using the gSender CNC control software, allowing the use of toolpaths generated by a slicer or other CAM method. Our use of interpreters differs from that of current motion control systems, which are single-purpose G-code interpreters, and may enable extensibility through plugins [25]. Further, because SSL Interpreters ultimately output motion streams from predefined paths, the operator can flexibly adjust predetermined motion without kinematic disruption.

Recorder and Playback. We extend manual input to enable automated machining through a set of SSL components that support motion stream recording and playback. A FourTrackRecorder synchronously reads any four motion streams and records their incremental changes to an onboard SD card at the interrupt frame rate of 25kHz. A complementary FourTrackPlayer synthesizes motion streams from the recording. Together, these components allow motion sampling and looping, as demonstrated in Sec. 5.1.4.

4.2.5 SSL Alignment with Design Goals. The SSL further supports our design goals by extending our modular hardware platform with flexible components for composing digital fabrication interactions. We further enable **rapid** prototyping (DG1) by providing CNC-specific abstractions (Kinematics, Channels, OutputPorts) that streamline the control of multiple motors in a precise, coordinated manner. We support **real-time** operation through a declarative dataflow programming paradigm compatible with embedded systems, enabling continuous modification during operation with extremely low accumulation of positioning errors (DG2). Finally, we support **hybrid** control through a wide variety of components for reading manual input and automated motion, as well as methods for combining them to control machine behavior (DG4).

4.3 Example Basic Module Prototypes

The Basic Modules support CNC design via physical metaphor by enabling the prototyping of CNC control inputs inspired by manual tools and interfaces. We describe two example basic module interfaces we created: a low-fidelity Mixer Module and a high-fidelity Pantograph Basic Module. We constructed the mixer from cardboard in under an hour and spent several days machining the pantograph.

4.3.1 Mixer Basic Module. Figure 6 illustrates how a Basic Module is configured to realize a *mixer* module (Fig. 8B) that generates sinusoidal toolhead motions in sync with a machine's motion, with real-time control over frequency, amplitude, and phase. We connect an analog foot pedal (6A) to synthesize a velocity that synchronizes a 1DWaveformGenerator and also passes through to the Stepdance output. We use three potentiometers to control the waveform generator's amplitude (6B), frequency (6C), and phase (6D). The resulting motion stream, along with the synthesized velocity, is sent via Stepdance OutputPort A to a downstream Machine Controller Module. A pushbutton (6E) enables or disables output from the waveform generator. We used this module to control the Wheelbot described in section 5.2.2, and, as a result, it also includes other interface elements not represented in Figure 6.

4.3.2 Pantograph Basic Module. Figure 7 shows how we configured a Basic Module as part of a *digital pantograph* interface (Fig. 8A). The digital pantograph is a real-time stylus-based input device that continuously converts motion of a stylus fitted with a variety of pen tips (e.g., a ballpoint pen, a needle-point stylus, or ball-tipped styli of various diameters) into a real-time 3-dimensional Stepdance

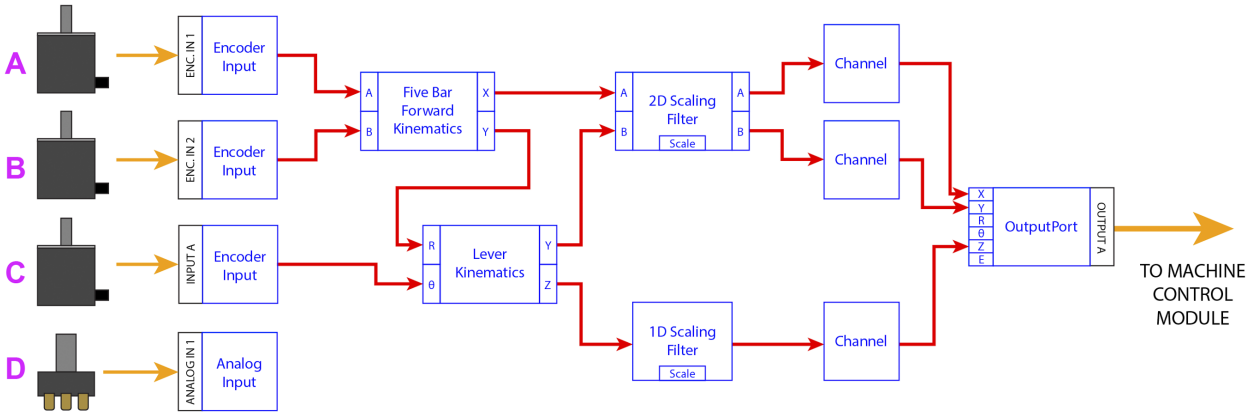

Figure 7: Pantograph Module Firmware Architecture

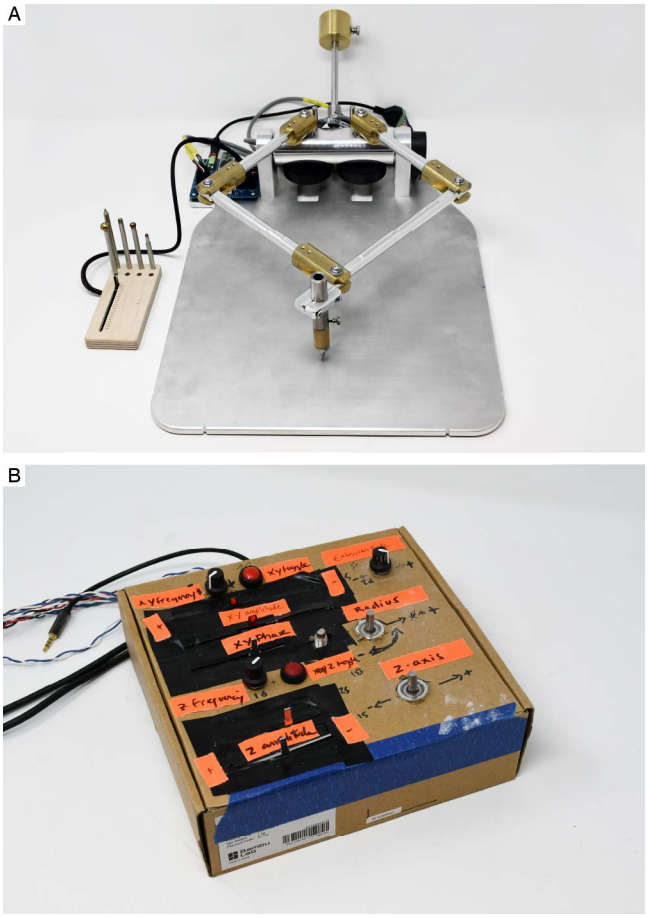

Figure 8: A) A high-fidelity prototype of a digital pantograph input device. B) A low-fidelity prototype of a mixer input device.

motion stream. Stylus position is sensed through the use of a parallel SCARA kinematic mechanism [68] that transforms translational motion of the stylus into rotational motion at two pivot points, which is then captured with high resolution optical encoders (US Digital E3-10000, with 40K pulses per revolution) (Fig. 7A, 7B). Digitizing resolution in the XY plane is approximately 25μm over a range of approximately 200x200mm. Height information of the stylus is additionally captured by mounting the linkage mechanism on a pivoting trunnion and capturing its angle with a third optical encoder (US Digital E5-5000, 20K pulses per revolution) (Fig. 7C). The linkage encoders connect to the basic module via the encoder ports. They are internally mapped to a FiveBarForward kinematics component that converts rotational positions to XY coordinates in the plane of the linkage. The third encoder repurposes the Stepdance input port and is internally mapped to the angle input of a Lever kinematics component. This both projects the stylus's Y position onto the plane of the pantograph base, and also finds its Z position. A sliding potentiometer (Fig. 7D) attaches to analog input A1, and provides control for the output scale via ScalingFilter components, ranging between 1:10 reduction and 3:1 amplification. When the pen is lifted above a certain height, a main loop routine disables the XY output, allowing a user to reposition the stylus without affecting the attached machine. The Digital Pantograph Basic Module converts between encoder readings and output motion streams at an interrupt frame rate of 25kHz, enabling real-time control of a downstream module or machine.

5 Applications

To demonstrate how Stepdance supports CNC design via physical metaphor, we developed a series of CNC machines inspired by manual design and fabrication tools. Some examples reference craft tools, whereas others reference physical interfaces from gaming and music. By presenting machines in the order in which we developed them before and during the Haystack residency, we convey how Stepdance facilitates repeated cycles of ideation and development both within and across machines. We provide full SSL source code for each example in the appendix.

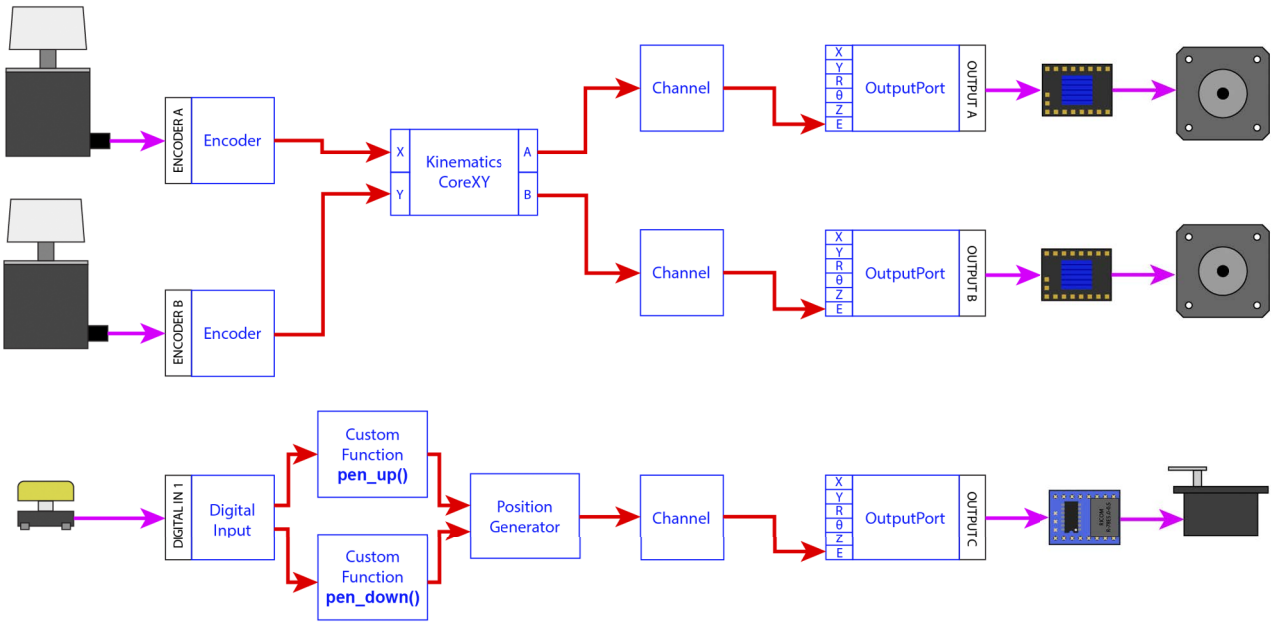

Figure 9: Step-a-Sketch SSL Firmware Architecture

5.1 Plotter as...

We began our development process with the Axidraw [51], a plotter that is compatible with a wide range of media and popular among artists and craftspeople. The Axidraw has two stepper motors for XY planar motion and a servo for pen height. It uses a single-belt plus-shaped differential kinematics system for XY positioning. We configured the Stepdance machine controller to drive the XY-plane motors with two TMC2209 stepper drivers and one hobby servo driver to control the pen movement.

5.1.1 Etch-a-Sketch. We used the Etch A Sketch—a mechanical plotter and popular toy—as a metaphor for a "Hello World" prototype because it would enable us to verify the real-time performance of our motion control framework quickly. We reasoned that the simplicity of the Etch-a-Sketch interaction should translate to a simple Stepdance physical setup and program.

We created the hardware interface for the Stepdance Etch-a-Sketch (re: Step-a-Sketch) by wiring two encoders and a push button to the machine controller module. The program for Step-a-Sketch instantiates three OutputPorts and Channels for the corresponding Axidraw motor drivers, as well as two encoders and a button. We map the encoder outputs to the CoreXY kinematics' X and Y inputs, and map the CoreXY output to the X and Y channels. To control the pen movement, we use a position generator mapped to the z channel. We use a callback that, on button press, toggles the position generator to an absolute position of 200 or -200 steps. Figure 9 shows the full Step-a-Sketch firmware architecture.

The result is an interaction where the operator can control the position of the Axidraw in real-time on the X and Y axes by turning the encoder knobs (figure 10A). They can enable drawing by toggling the button to raise and lower the pen to produce rectilinear drawings (figure 10B). We provide this example as a hello-world introduction demonstrating the floor of development for converting an existing CNC machine into a real-time interactive tool. Note that in the figure, the community member is interacting with a smaller plotter we built from scratch for another project. The full code for the Step-a-Sketch is available in appendix B.1

5.1.2 Harmonograph. Harmonographs are a class of mechanical drawing instruments where two or more pendulums control the movement of the pen. In a two-pendulum harmonograph, one pendulum moves the pen along the x-axis while the other moves it along the y-axis. Varying the pendulums' frequencies and phases creates different Lissajous curves. While it is possible to mathematically model a harmonograph and convert that model into a machine toolpath using a computer program [53], we can achieve the same effect by attaching two pendulums to the same encoders we used for the Step-a-Sketch, allowing them to swing freely. The result is a mechanical harmonograph that drives the plotter's motion. This example demonstrates the power of enabling real-time control (DG2). Rather than relying solely on software modification to control behavior, a designer can modify the mechanical aspects of a physical interface to quickly and fundamentally alter machine behavior. Figure 11 A shows a Haystack community member swinging the harmonograph pendulums, and B displays the resulting Lissajous curve plots.

5.1.3 Game Controller. In building Stepdance, we sought to leverage rapid prototyping speed to act on craftspeople's ideas for machine control quickly. Our first opportunity for this occurred when Haystack's Fablab coordinator wanted to experiment with controlling a 3D printer like a video game, using a joystick to drive the toolpath and a button to jump to the next layer. We had a joystick

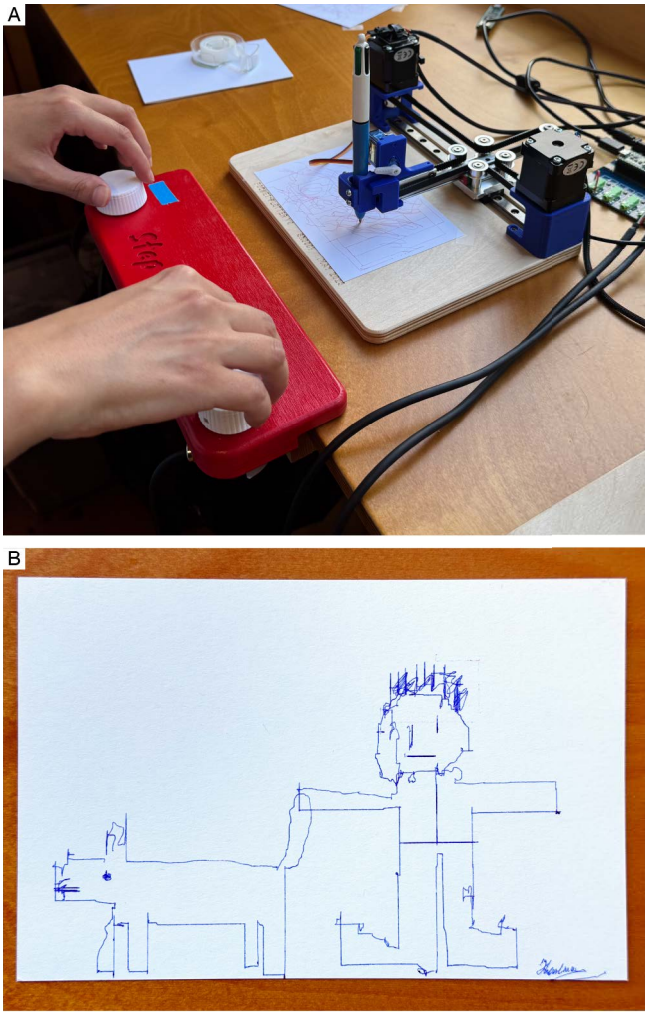

Figure 10: We converted a plotter into an Etch-a-Sketch (Step-a-Sketch) using two encoders and a push button as an interface. A) A Haystack community member drawing with the Step-A-Sketch system. B) A community member-produced drawing.

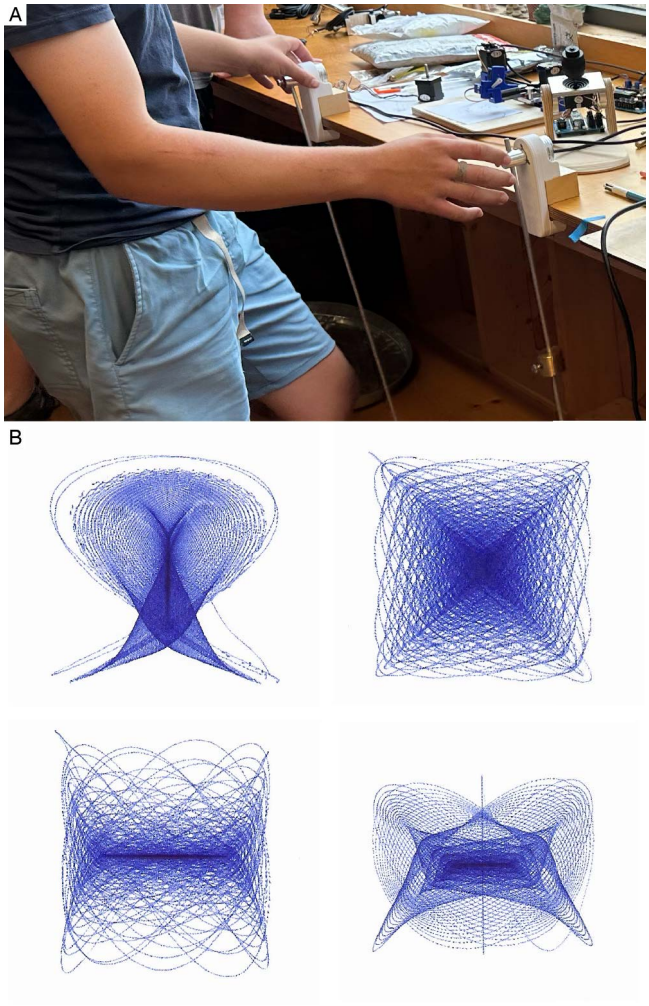

Figure 11: Plotter as Harmonograph: We create a mechanical-digital harmonograph by attaching pendulums to the two encoders that control the X-Y position of the plotter. A) a Haystack community member interacting with the harmonograph. B) Lissajous curve plots produced by the harmonograph.

with us and quickly prototyped an initial version of the joystick control on the Axidraw.

The joystick had three analog outputs, one for up-down motion, one for left-right motion, and a rotary twist knob built into the handle. We wired these to analog inputs on a Basic Module. We defined three velocity generators and mapped the analog inputs to them. We constrained the floor and ceiling of the analog input to -50 and 50 mm/sec, respectively. We then mapped the velocity generators to the x, y, and z channels for the module output ports. Figure 13A shows the completed game controller module. We plugged the joystick module's output into the current Step-a-Sketch machine controller. We made a minor change to the Step-a-Sketch code to map Input A to the CoreXY kinematics' X and Y inputs and the Channel Z input target position. This allowed an operator to control the plotter either through the Step-a-Sketch encoder knobs or using the joystick. Figure 12 shows a Firmware schematic for the modified Step-a-Sketch example with the game controller attached. By moving the joystick, the operator can control the x-y position, and by twisting the joystick, they can raise or lower the z servo. The rapid addition of a new interface component to an existing mechanism enabled us to test the coordinator's idea, and spawned an unexpected interaction where multiple people could simultaneously use the original and joystick interface. Figure 13B shows two community members driving the plotter with the Step-a-Sketch and Game Controller interfaces at the same time.

5.1.4 Pantograph and Inkscape Integration. As described in section 4.3.2, we developed a Basic Module that outputs the X, Y, and Z motion of a mechanical pantograph. By plugging the pantograph

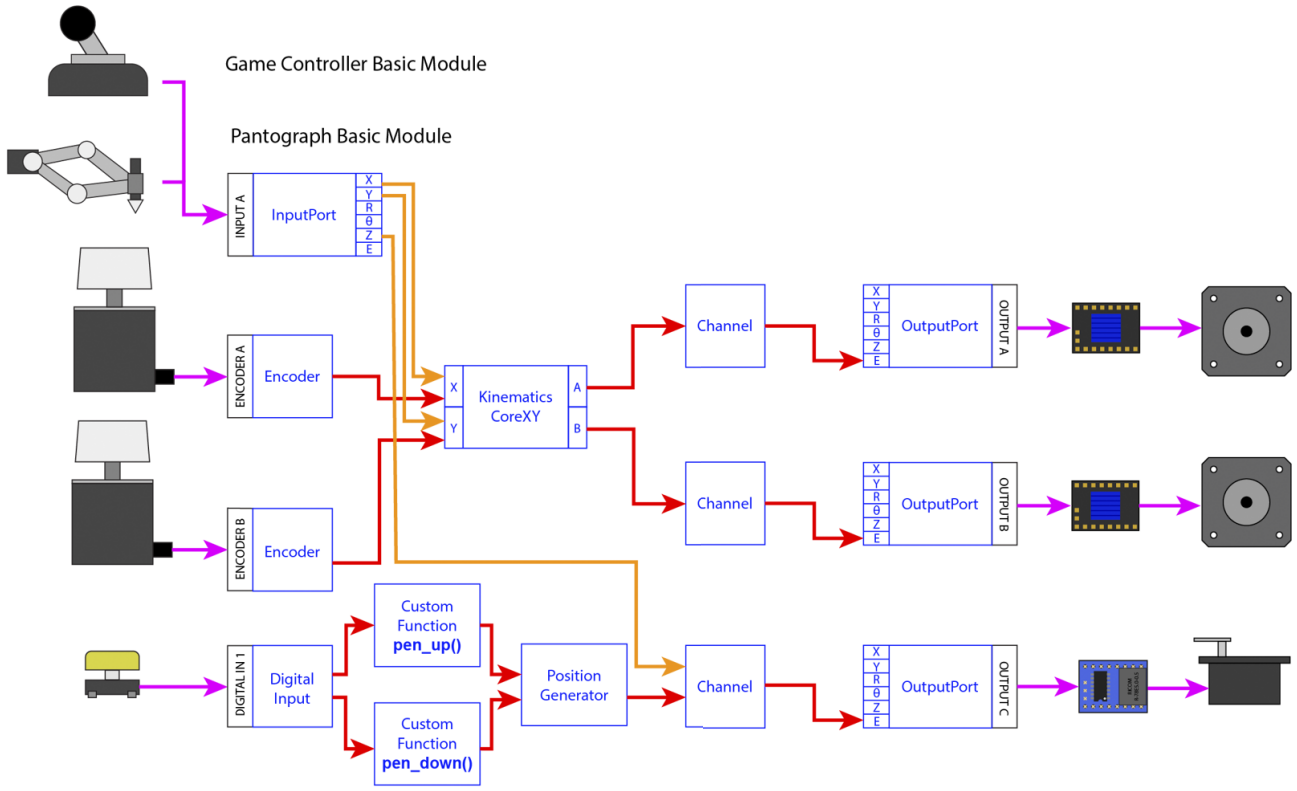

Figure 12: Plotter Game Controller and Plotter Pantograph Firmware Architecture. The program is identical to the original Step-a-Sketch code (figure 9) except for initializing Input A and mapping the X and Y outputs to the X and Y inputs of the Core XY Kinematics, and the Z output to the Output C Channel target position. Additions are shown via orange arrows.

into the Step-A-Sketch machine controller, we can control the plotter's position through drawing-based input (figure 14A). We can use the pantograph's onboard scaling slider to scale up or down the drawing output on the plotter, replicating the original intended use of a mechanical pantograph (figure 14C).

We explored integrating the pantograph with predefined motion paths to better align the Axidraw interaction with manual drawing. We reprogrammed the Axidraw machine controller to use the Eibotboard Interface to drive the plotter with SVG drawing commands from Inkscape– effectively reproducing the original Axidraw functionality. We then extended this functionality with the pantograph, allowing us to manipulate plotted SVG designs on the fly with manual input. This led to a series of experiments to determine possible applications. Starting with a grid design in Inkscape, we traced the inside of a roll of electrical tape with the pantograph (figure 15B). The combination of the pantograph's circular motion with the linear SVG motion results in a sine wave with uneven frequency from the irregular timing of the manual tracing (figure 15F, J). By scaling the pantograph input, the circular modification progresses from small to large waves. Other experiments include tracing popsicle sticks in the perpendicular axis to the motion of the Axidraw to create non-looping offsets (figure 15A, H, I), and using pre-measured markers on the pantograph surface to spatially relocate repetitions of the SVG through manual placement (figure 15G, K). Similar to the harmonograph, real-time control with the pantograph enabled us to quickly shift our prototyping from the hardware domain to the physical domain, interspersing drawing and plotting with adding physical constraints and guides.

5.2 Clay 3D printer as...

We moved on to exploring 3D machining using Stepdance to modify the Potterbot Micro 10—a popular clay 3D printer [45]. The Potterbot extrudes wet clay into layered vessels. We selected the Potterbot specifically because of our proximity to ceramics faculty and students at Haystack. The Potterbot Micro controls extruder position via three rectilinear axes and drives extrusion with a piston. Because wet clay extrusion requires substantial torque, we used external Gecko GR214V stepper driver modules connected to the Machine Controller.

5.2.1 Pottery Wheel. Although clay 3D printers use the same material as manual ceramics, they involve a design and interaction process nearly identical to plastic printing. As mentioned in section 2.1 we previously built a specialized polar 3D printing mechanism– the DPW– that enables operators to print cylindrical volumes in a manner reminiscent of, and compatible with, manual throwing [37]. Using the DPW, the operator prints a cylinder by rotating the wheel with a foot pedal, akin to manual throwing, while controlling the

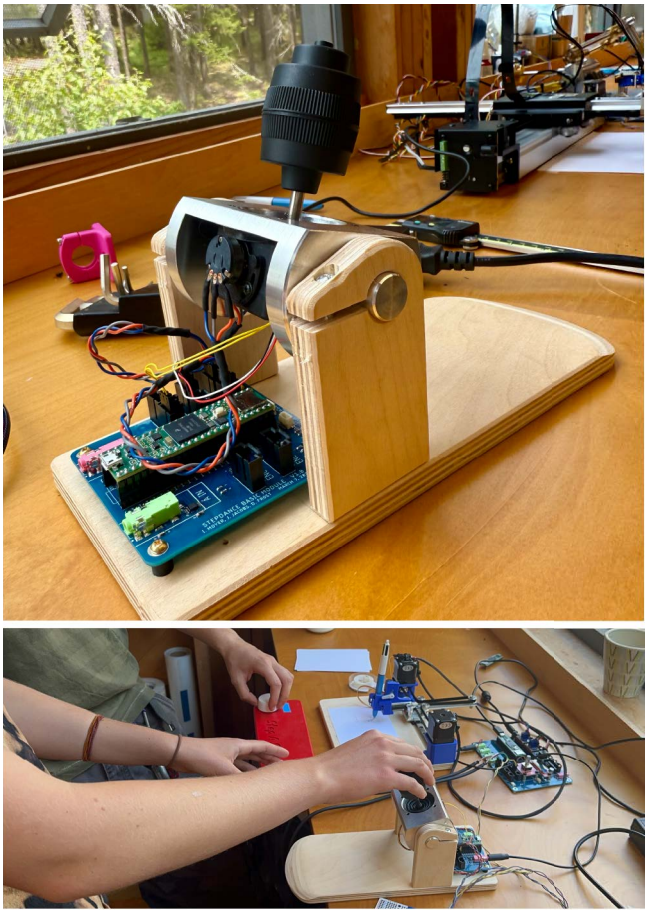

Figure 13: Plotter as Video Game: We developed a Basic Module joystick controller interface (A) and connected it to a plotter, enabling control of the plotter position through joystick movements. B) Two Haystack community members drive the plotter by simultaneously using the joystick and Step-a-Sketch interfaces.

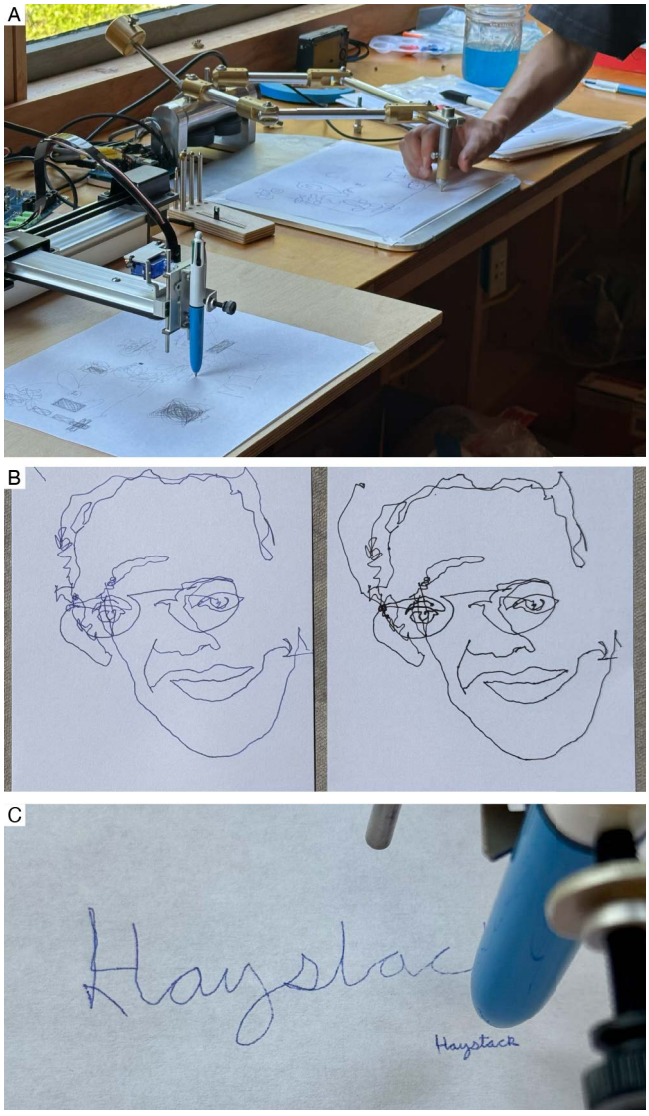

Figure 14: A) We connected the Pantograph Basic Module to the plotter, enabling real-time control of plotter positioning through drawing. B) Left: plotted drawing, right: pantograph drawing. C) Unscaled and scaled-down plotter output of manual handwriting input.[5]

extruder arm's radius with a lever. The Z-axis and extruder automatically move up and extrude material proportional to the wheel velocity, creating a spiralized toolpath. We explored using Stepdance to create a similar interaction on a widely available commercial machine. We created the hardware interface for the Potterbot wheel (re: Wheelbot) by wiring two encoders and a pottery wheel pedal to the machine controller. For the Wheelbot program, in addition to the channels and inputs, we use a Polar-to-Cartesian Kinematics object coupled with a Velocity Generator. To drive the movement of the XY stage like a pottery wheel, we map the analog pedal input to the velocity generator speed input. We then map the velocity generator to the polar kinematics input angle. We use the polar kinematics x and y to drive the XY stage of the potterbot, and we map the encoder to the polar kinematics input radius.

For the z axis, we want to incrementally raise the z as the "wheel" turns at a rate that maintains a target layer height. We define a layer height parameter in the program that corresponds to 1/2 our nozzle diameter. We initialize a 1D scaling filter (z_gen) and set the ratio to `layerHeight`$/2\pi$. We then map the polar kinematics input angle to the scaling filter input and map the scaling filter output to the z channel.

Lastly, to drive the extruder, we need to extrude sufficient clay relative to the distance traversed on the x-y plane. We initialize a 2D Path Length Generator (e_gen) and map the input channel positions corresponding to the X and Y axes of the Potterbot. We can then use the 2D Path Length Generator's output to drive the extruder position, ensuring the extrusion rate automatically adjusts

[5]Image C courtesy ofHaystack Mountain School of Craft

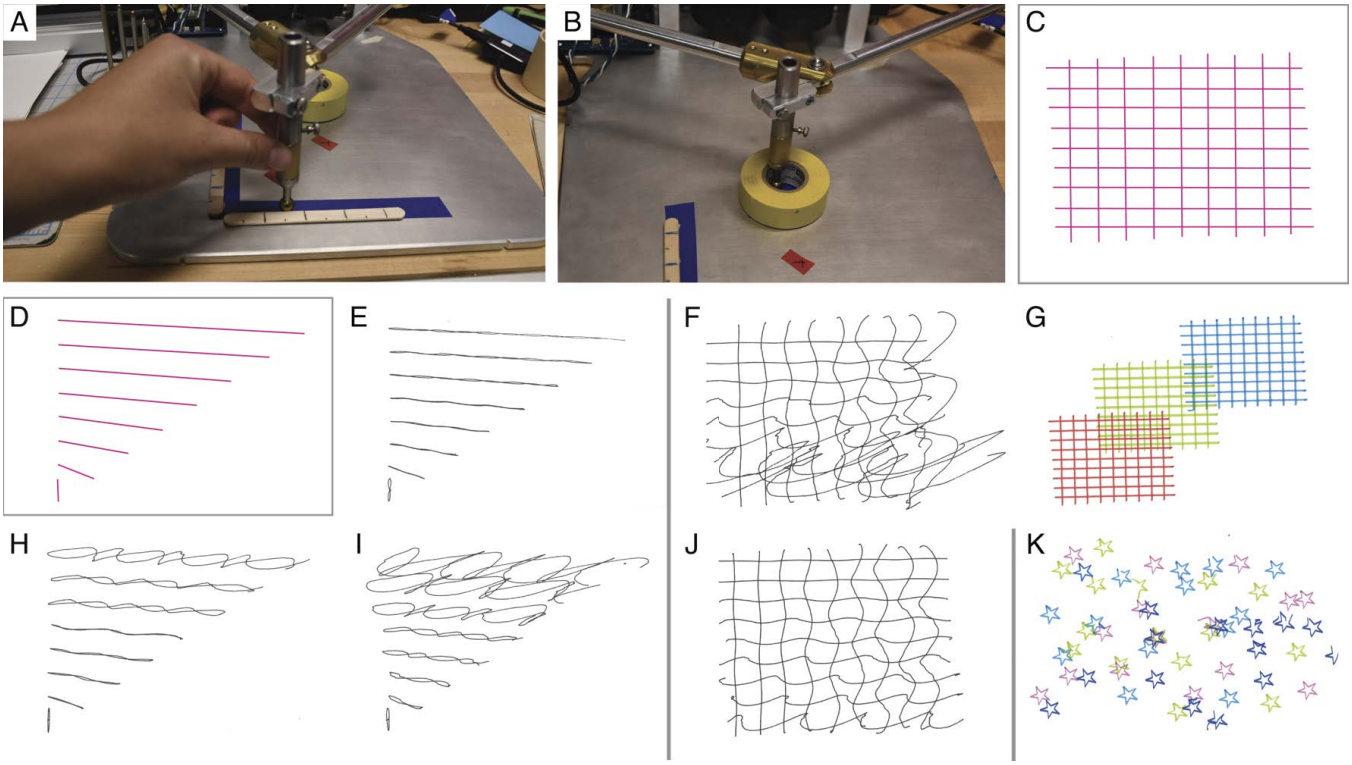

Figure 15: Stepdance SVG combined with pantograph plots. A) Using Popsicle sticks as linear X or Y motion guides for pantograph motion. B) Using electrical tape as a circular guide for pantograph motion. C) Input SVG grid. D) Input SVG lines. E) Lines with low-scaled circular motion. F) Grid with progressive circular scaling motion. G) Grid plotted three times with manual relocation through the pantograph. H) Lines with mid-scaled circular motion. I) Lines with progressive high-scale circular motion. J) Grid with linear X or Y motion. K) Star SVG repeated 15 times, manually relocated.

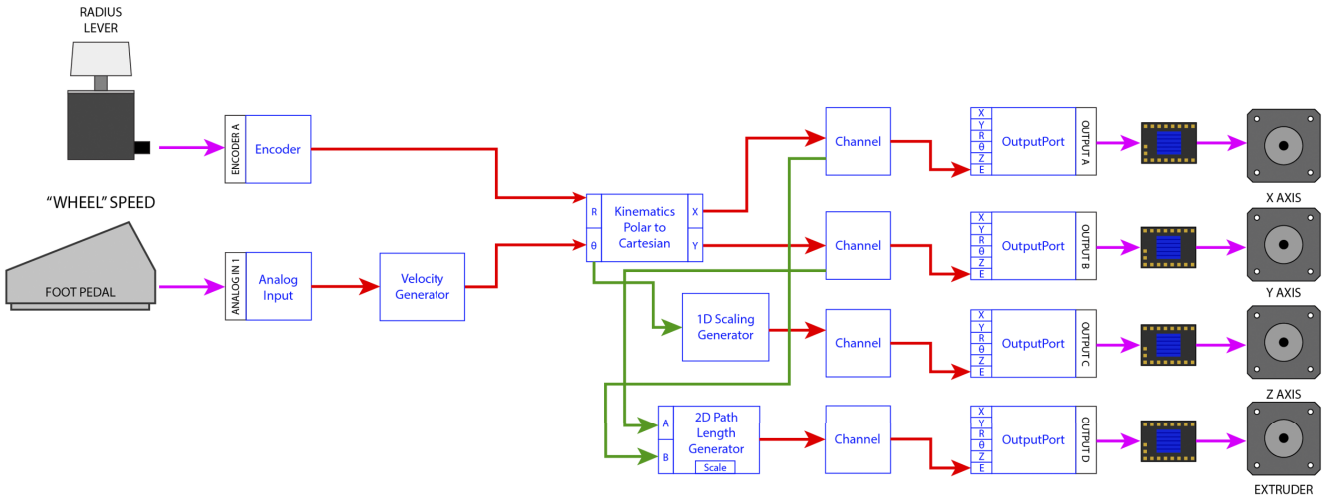

Figure 16: Wheelbot Firmware Architecture.

to proportional movements on the X-Y plane. Figure 17 shows the Wheelbot firmware architecture. Complete code for the Wheelbot is available in appendix B.2. The Wheelbot creates an interaction identical to the pot-assist modality of the DPW: the operator can

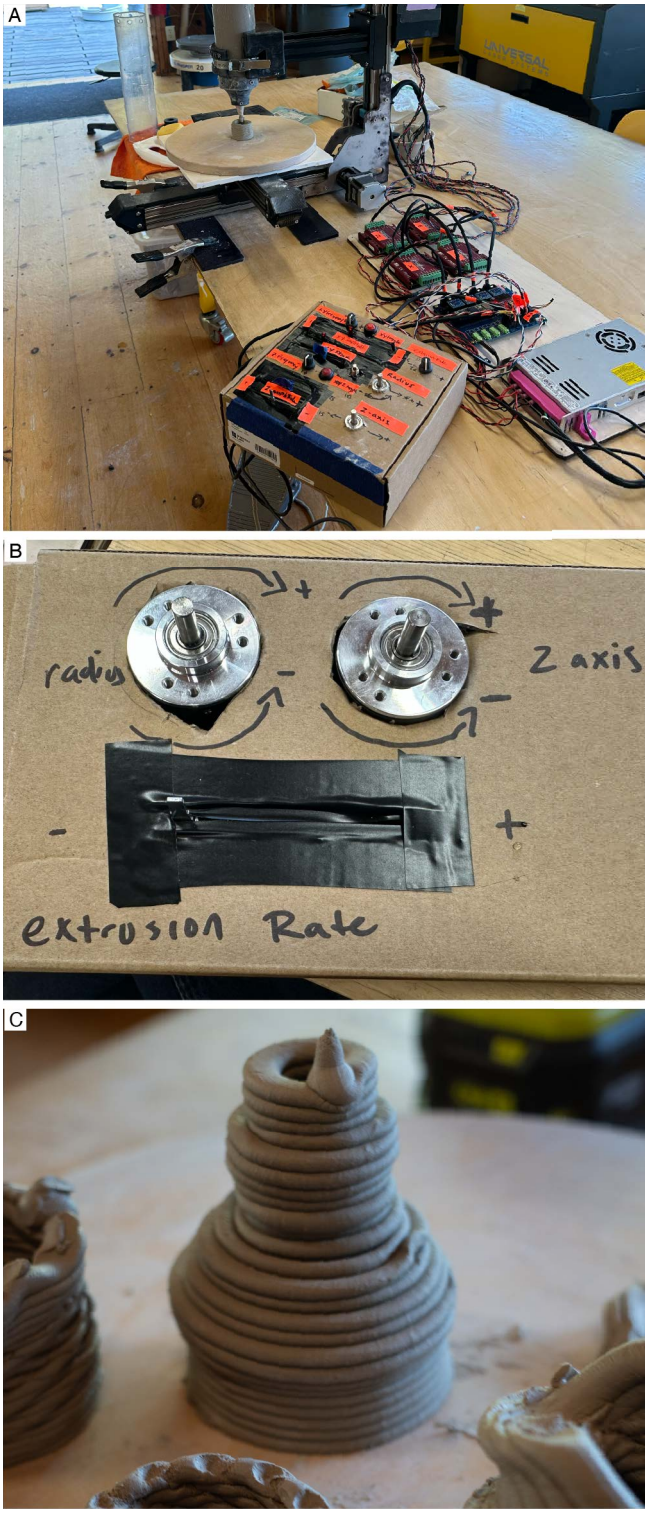

Figure 17: Wheelbot v1: A) The Potterbot wired to the Stepdance Machine controller, external power supply, and stepper drivers. The foot pedal is visible under the table. (Note that this image also contains the mixer referenced in section 5.2.2. The cardboard interface consists of an encoder to control the radius, an encoder to control the z-height to set the initial printing position, and a linear potentiometer to tune the extrusion multiplier. C) The first 3D printed vessel with varying radius created using Wheelbot v1.

3D print a clay pot without a pre-programmed design by spinning a simulated wheel and adjusting the traversed circle to the desired radius as the print progresses.

Using Stepdance to emulate a highly specialized CNC on a general-purpose mechanism created a design scenario in which we sought to push the resulting machine to support as many of the original's affordances as possible, albeit through alternative strategies. This, in turn, revealed new workflows that surpassed the functionality of the original machine. One of the benefits of the DPW is that it enables potters to alternate between printing and throwing without any CAD or CAM. While we could not reconfigure a Cartesian mechanism to enable throwing directly, we found we could enable the same workflow by drilling two holes in the x-y stage that correspond to the aspect ratio of standard wheel bat holes.[6] This allowed us to throw a vessel on a bat on a standard pottery wheel (figure 18A), remount the bat on the Wheelbot, zero the extruder in the center of the XY stage, and then manually adjust the radius of the extruder until it corresponded with the diameter of the thrown pot (figure 18B-C). We could then print on the pot, remove the bat, and return it to the wheel to throw elements of the printed form- for example, smoothing and compressing the lip of the printed coil (figure 18D-E).

In observing the ceramics class, we noticed how potters would often manually deform their vessels off the wheel to achieve non-circular geometries. A limitation of the original DPW polar mechanism was that it was difficult to fabricate shapes other than circles without using G-code. However, because the Potterbot is a Cartesian printer, we found we could easily modify our Wheelbot code to support printing elliptic cylinders by adding a single 1D Scaling Filter that we mapped to the x channel's target position. Because Stepdance allows mixing of multiple motion streams, this mapping scaled the x dimension of the cylindrical pot without disrupting any other aspect of printing. Figure 19 shows the process of manually warping a thrown pot into an ellipsoid and then printing an oval form on top of the thrown ellipsoid.

5.2.2 Audio Mixer. During our residency, we observed ceramics practitioners creating complex surface textures and ornamentation through manual carving, stamping, and painting. While offering fundamentally different aesthetic qualities than manual ornamentation, surface texturization is one of the primary design affordances in clay 3D printing. Using CAM programming, skilled creators can produce complex surfaces by oscillating the toolpath between layers [5, 14]. We therefore sought to prototype a method for designing the surface qualities of clay 3D-printed vessels that ceramics practitioners could use without resorting to desktop CAM. We informed our approach by drawing from prior exploratory digital fabrication tools that reference mixing in music production [27, 56]. We investigated using Stepdance to extend the Wheelbot with a mixer-like interaction to control clay 3D printing surface texture in real time.

To enable surface texturization without CAM, we attached the basic mixer module described in section 4.2.4 to the Wheelbot input. We refactored the Wheelbot code so that the Velocity Generator

[6] A bat is a common pottery tool consisting of a circular wooden board with two holes that a potter mounts on the wheel using corresponding screws in the wheel platter. The potter can then throw on the bat and easily remove their finished piece from the wheel.

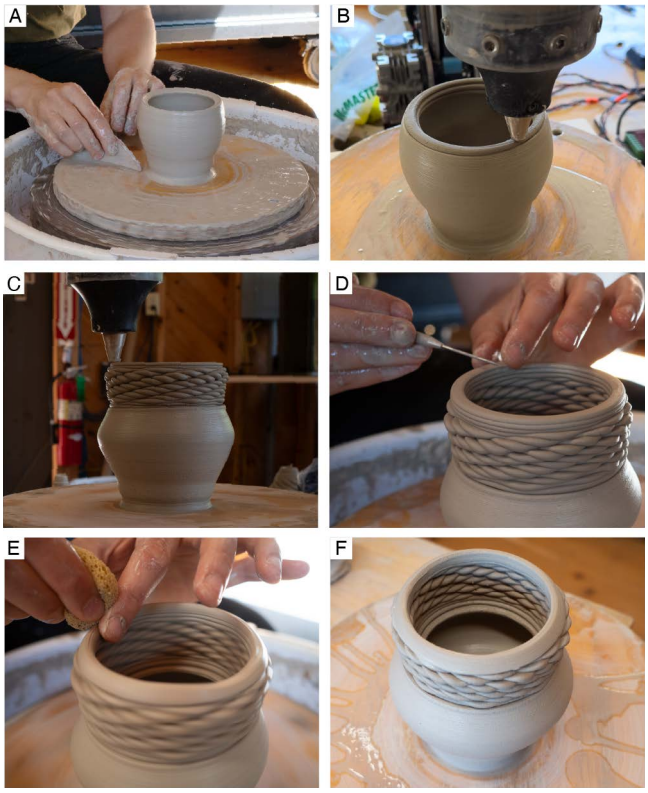

Figure 18: The workflow for transitioning between printing and throwing using the Wheelbot. A) An operator throws a vessel on a bat using a traditional pottery wheel. B) The operator mounts the bat on the XY stage of the Wheelbot using the pre-drilled screw mounts. It is automatically centered on the stage. The operator adjusts the Wheelbot's radius until the extruder is over the pot lip, then begins printing. C) The operator prints their desired form using the Wheelbot Mixer. D) If desired, the operator can return the printed form to the wheel for further manual modification. Here, they trim off the uneven top of the coil left by the printing process, and E) smooth and reshape the printed lip with a sponge. F) The completed pot.

output from the mixer module mapped to the Polar Kinematics input angle. This retained the pedal control over the wheel rotational velocity, mapping it over from the mixer. We then mapped the output from the waveform generator from the mixer to the Polar Kinematics input radius. The result is an interaction where the operator can control the Wheelbot as before, and by activating the waveform generator with the mixer, can introduce oscillations in the x-y values of the toolpath in real-time. Varying the amplitude increases the amount of overhang for each oscillation, varying the frequency increases the number of oscillations in a layer, and varying the phase will cause the layers to either stack or be offset. Figure 23 shows the firmware architecture for the Wheelbot with mixer and figure 20A shows a Haystack community member printing a textured pot with the mixer, alongside other community member-created results with varying degrees of XY and Z oscillation. Overall, this prototyping process is similar to mapping digital audio workstation tracks and effects into a MIDI mixer and using the mixer to experiment with different variations. We discuss craftspeoples' response to the Wheelbot and texturization in section 6.1.

5.3 Plastic 3D printer as...

Our time at Haystack was extremely generative. Both the authors and craftspeople came up with many more ideas than we could execute on site. We chose to focus on the Axidraw and Potterbot during our residency. When we returned to our lab, we investigated the idea generated by our interactions with the ceramics instructor and applied Stepdance to a plastic 3D printer.

We selected a Creality Ender-3 Pro, a cheap but robust printer with a rectilinear mechanism with four NEMA17 steppers controlling the X, Y, Z, and extruder axes. We wired the steppers to the Stepdance machine controller with four TMC2209 drivers. Because the current version of Stepdance lacks outputs for heating elements, we controlled the nozzle and bed temperature through the original Ender control board.

5.3.1 Sketch-based stencil extruder. We returned to the original motivating example of enabling 3D printing of stencils for clay stamping (see section 3.1.) We already had the Pantograph module and reasoned that 2D drawing would be a familiar interaction for people without prior CAD or 3D printing experience. We therefore sought to create an interaction where the operator drew their desired stencil with the pantograph, which would result in the Ender 3D printing multiple repeating layers of the 2D drawing to create a 3D stencil object.

We plugged the pantograph module into the Ender machine controller's input port and wired two buttons to the digital inputs. The program for the Ender machine controller is as follows. We set up four output ports and channels to correspond with the Ender's four axes, identical to the previous example with the Potterbot. We initialize two Position Generators, a 2D Path Length Generator, a Four Track Recorder, and a Four Track Player. We map the position generator outputs to the a and b channels (corresponding to the x and y output ports), and we map the recorder inputs and player outputs to the X and Y channel input target positions.

To take the input from the pantograph and repeat it across multiple layers, we use the Stepdance recording and playback functionality. We set the first button as a toggle and assign a callback to start and stop recording. For the start recording callback, we have the recorder store the current X and Y absolute positions. We set the second button as a standard push button and on press, check whether the player is actively playing. If it is, we stop the playback. Otherwise, we call the initiate playback function, which triggers the x and y position generators to move to the start x and y positions. This ensures that playback starts from the same position as when the recording was initiated. We also move up a layer by triggering the z-position generator to increment by the layer height.

To handle the recording and playback logic, we write imperative code within the loop function. We check the system's state. If the system is waiting to start play (e.g., the player is not actively

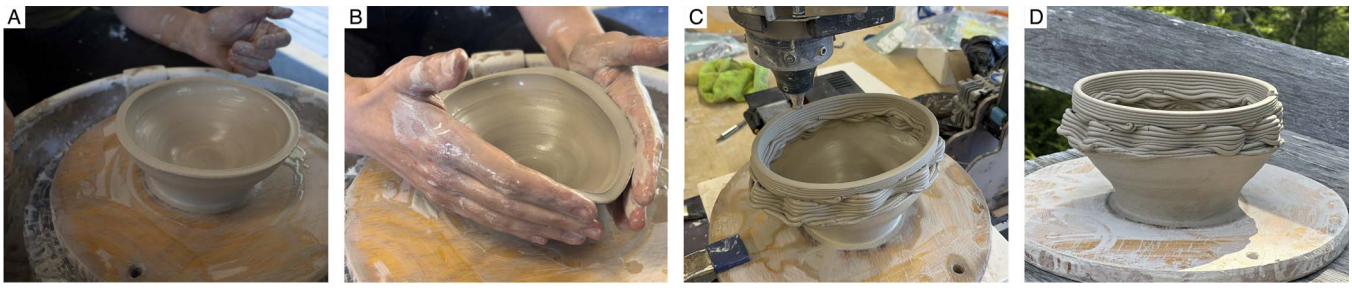

Figure 19: Using the scaling functionality of the Wheelbot, it is possible to print ellipsoid forms. Here, an operator throws a circular pot (A), then manually deforms it to an ellipsoid (B), then mounts it on the Wheelbot and adjusts the 1D Scaling Filter to print an oval toolpath (C) that matches the shape of the deformed pot (D).

Figure 20: We used the mixer module in tandem with the Wheelbot to create a mixer-like interaction for introducing surface texture into clay 3D printed vessels on the fly. A) A Haystack community member using the mixer. B-D) Vessels produced with the mixer by Haystack community members. E) Bisqued vessels printed with the Wheelbot.

engaged in playback, but the operator has triggered playback and the extruder is currently moving back to the start position), we check whether the channel a and b positions have returned to the start point. If this is true, we begin playback. Otherwise, the system checks whether the printer has finished a layer. This occurs when playback is no longer active. When this happens, we trigger another layer. Figure 24 shows the firmware architecture for the Stencil Maker. The full code is available in appendix B.3.

The resulting interaction is as follows. When playback is not engaged, the operator can begin recording and draw with the pantograph (figure 21A). The printer will follow the motion of the pantograph and, in real-time, print a single layer on the bed that corresponds with the pantograph motion (figure 21B). We depend on the operator's discretion to draw a path with approximately the same start and end points. Once the operator has finished drawing, they can hit the record button again to stop recording. If they are

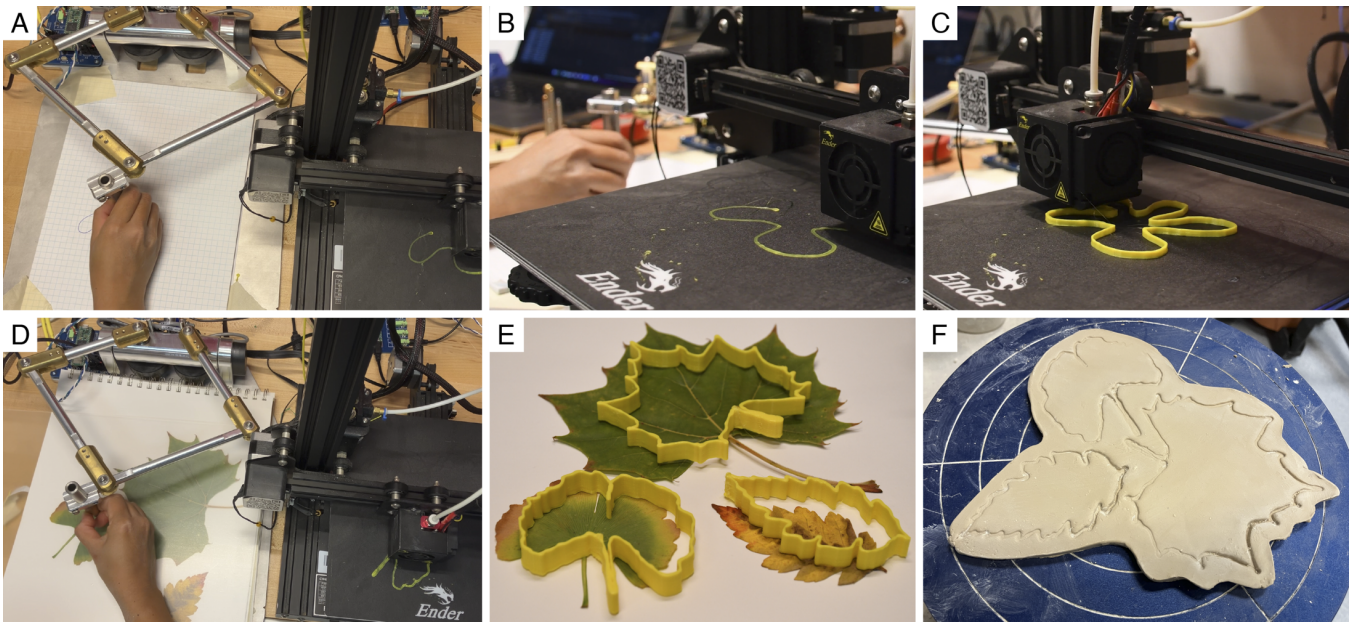

Figure 21: Sketch-to-stencil. A) We demonstrate operating a plastic 3D printer via pantograph input: the operator can draw while the printer prints on the bed. B) This first layer is recorded as motion streams. C) These can be replayed on a loop to print an extruded 3D stencil shape. D) Tracing leaves with a probe tip on the pantograph. E) Plastic stencils and the original leaves. F) We use the stencils to mark clay slabs.

satisfied with the first layer, they can hit play, and the printer will automatically repeat the pattern, advancing to the next layer at every cycle of playback (figure 21C). The operator can stop playback when they are satisfied with the stencil's height. We used the stencil maker to produce stencils from free-form doodles, as well as from the traced form of leaves we found outside our lab (figure 21D-F).

5.3.2 Limitations of Stepdance for state-based workflows. The sketch-based stencil extruder was the most complex interaction we created with Stepdance. Attempting to prototype a device of this complexity validated the ceiling of Stepdance; while the program is non-trivial, it was feasible. At the same time, we recognize that some elements of the system that should be relatively straightforward to implement, like recording and playback, require substantial imperative code to function practically. This is, in part, due to the absence of state-based abstractions that enable developers to manage mapping behavior across discrete workflow stages. We see this as vital future work.

6 Discussion

We developed Stepdance based on the theory that a real-time motion-control framework would enable the use of existing CNC machines to prototype craft-aligned forms of digital fabrication. In our discussion, we reflect on the role Stepdance played in CNC prototyping in the context of our three residency objectives and four system design goals. We then consider how modular interface prototyping may enable future opportunities for CNC functionality.

6.1 Rapid CNC prototyping to engage practitioners in imagining alternatives

We stipulated that our system should enable *rapid* motion control prototyping (DG1). Our applications and sample artifacts provide one preliminary indicator of the development speed. We were able to prototype multiple novel interactions across a plotter and two different 3D printers within a matter of weeks. We contextualize this rate of development by noting that **we, the authors, performed all machine prototyping.** We see future opportunities to assess how Stepdance supports rapid prototyping by other digital fabrication researchers and designers. For the remainder of this section, we discuss two benefits Stepdance provided the authors for rapid prototyping in the residency, that with our second and third methodological objectives, wherein we aimed to conceptualize applications in response to craftspeople's ideas, and sought to validate Stepdance through craftspeople's interactions with modified CNCs.

An idealized conception of a craft-engineering collaboration might consist of an experience in which craftspeople have inventive ideas, engineers quickly build tools that reflect those ideas, and craftspeople delightedly apply those tools to making things. While we had no expectations that this would be our experience, we believe it is illuminating to look at how our actual collaboration process aligned with or diverged from this idealized conception. First, Stepdance did in fact enable quick and productive feedback on experimental ideas for CNC interaction. As the joystick applications (section 5.1.3) illustrate, people with digital fabrication experience have ideas about alternative ways to interact with CNC machines. Whereas developing real-time joystick-based control with a G-code controller would be infeasible, Stepdance enabled us

to build a working version of the community member's envisioned interaction in a matter of minutes. The truly critical element of the process, however, occurred when the coordinator tried out the joystick interface– and *didn't like it.* She described how the control didn't work as she expected, and we could observe her actively struggling to achieve the intended machine movement. Contrary to the idealized scenario we presented earlier, we argue that, if one's goal is to invent *meaningful* CNC technologies, this response is actually the optimal outcome in early-stage CNC prototyping. Within the design process, being able to experience the drawbacks of one's ideas in practice in a low-risk setting is extremely useful [48]. The Haystack community member could bridge the gap between what she perceived as a desirable interaction and its real drawbacks. As designers, we felt no attachment to the joystick controller because we had invested little effort in prototyping it. These combined factors created a context for quickly discussing alternative designs. Ultimately, defining meaningful applications of interactive fabrication is still an unresolved challenge. Toolkits that encourage rapid testing and pivoting from initial ideas without extensive engineering effort will increase the chances of discovering truly powerful applications of real-time control in digital fabrication without early design fixation.

Second, Stepdance enabled rapid prototyping of new artifacts that could shape people's perceptions of the overall potential of digital fabrication. Many Haystack community members were unfamiliar with or skeptical of digital fabrication. This attitude was especially present for clay 3D printing. Some ceramic practitioners were unsure of engaging with the technology and had negative associations with automated production. Once they began creating artifacts with the machine or observing others' work, they began providing feedback and suggesting alternative directions. For example, when the ceramics instructor observed the creation of a pot that integrated throwing and printing (Fig. 1I). He became interested in this piece, and described it as "honest". He discussed with us how the machine and the hand qualities were clearly evident in the piece, but didn't seem to be trying to replicate aspects of the other form of making. Previous researchers have found that clay 3D printing can be contentious in traditional ceramics communities, with some viewing it as an undesirable [55] or uninteresting [61] form of automation. We respect this skepticism given the history of CNC technologies supplanting skilled manual fabricators [41], and we see opportunities for productive dialog with skilled craftspeople on how to build future digital fabrication technologies that support rather than supplant skilled production. For this dialog to happen, however, it is first necessary to establish a shared point of connection or relevance with the digital fabrication technology. Our research suggests that multiple prototyping cycles may be needed to produce machines that are compelling enough for craftspeople to engage in the ideation process. Technologies like Stepdance may help entice a broader range of practitioners into the conversation on digital fabrication by lowering barriers to building CNC machines that blend hand and machine work.

6.2 Benefits of prioritizing physical-manual interfaces in CNC prototyping

The core of Stepdance is a motion control paradigm that enables direct manual input to drive the synchronized movement of multiple CNC motors (DG2). In analyzing the benefits of this paradigm, we return to our original motivating example of developing custom stencils for ceramic ornamentation (section 3.1), and our prototyped application in response to this example, the pantograph-controlled plastic 3D printer (section 5.3). There are multiple potential responses to the motivating example– we could enable designers to sketch on a digital tablet and computationally repeat the drawing to create a spiralized toolpath as demonstrated by Frost et al. [14] or use computer vision to digitize pen and paper sketches and convert those representations to STLs as demonstrated by Kim et al. [27]. Both of these approaches, while retaining manual input, are rooted in software-based solutions in which the bulk of design activities are performed through a desktop software pipeline that treats meshes, toolpaths, and G-code instructions as first-class objects, and which affords screen-based interactions and visual feedback. In contrast, Stepdance relocates design activities into an embedded pipeline, treating low-level abstractions of motor movement as first-class objects, and affords prototyping physical controllers.

This focus on physical controllers had several benefits. First, it encouraged the development of machines that supported immediate feedback in response to operator action. Haystack community members, including children, could walk up to a machine, move a controller, and get an output. While the output might not have always been desirable, this interaction was a striking contrast to watching people come into the FabLab, upload a print to the commercial 3D printer, encounter an error, and then need a coordinator's assistance to address it. Stepdance enables building CNC machines that correspond with Victor's creative coding principle of "getting something on the screen" to encourage people to create by reacting [66].

Second, Stepdance's emphasis on prototyping in the physical world created opportunities for physically situated collaboration. For example, three community members collaborated to create a 3D-printed clay vessel by taking turns using the mixer controller while the others observed (figure: 20C). While desktop software also affords collaboration, the emphasis is often on supporting people in different physical locations.

Finally, Stepdance's circumvention of digital CAD and CAM aligned with the preferred workflows of some craftspeople. This point is perhaps the most important because–in line with our first methodological objective– it helps illustrate how craft practitioners understand the benefits and limitations of digital fabrication in ways that are fundamentally different than many HCI researchers. To provide a particularly relevant example, after experimenting with some of the Stepdance machines, a graphics instructor specializing in paper-craft asked us if we were aware of any digital fabrication tools that could approximate scissors while reducing manual hand strain. When we asked her why she didn't use a CNC vinyl cutter or laser cutter, her answer was simple: "Because then I'd have to use CAD." While it may be counterintuitive to digital fabrication researchers to reject precise, automated software, we are undoubtedly underestimating the creative costs of sacrificing direct

material engagement for digital manipulation. What draws many people to craft is the opportunity to work with physical materials. Software workflows create separations and delays between design and material response– which can in turn decrease expressiveness, efficiency, and discovery. This is a tension we experienced directly as residents, as we spent much of our time looking at a screen, building the Stepdance programming library. At the same time, we observed the majority of other community members engaged in the physical world with their bodies- stamping metal with a press, weaving on a loom, and throwing with clay on a wheel. It was only when we were quickly transitioning between building physical interfaces and fabricating with them that we were able to engage with and learn from the craftspeople surrounding us.

6.3 Modularity benefits

Stepdance was informed by and shares some motivations with Peek et al.'s Cardboard Machine kit [42]. One key insight from that work was that CNC machine infrastructure and applications can be separated, providing a platform for creating CNC machines with diverse applications. Like the Cardboard Machine Kit, we also sought to develop a modular CNC prototyping toolkit (DG3). Our experimentation during the residency with the Stepdance basic modules and the modular SSL led to a new insight. By separating CNC interfaces and machine infrastructure, we can enable rapid prototyping across different materials and mechanisms. The Stepdance basic module enables control interfaces that pass relative motion data rather than predefined toolpaths or absolute units. In contrast, the SSL enables easy conversion between a given input range and machine motion. As our applications demonstrate, a designer can quickly swap input modules between different machines (e.g., applying Pantograph to the Axidraw Plotter and the Ender) or rapidly test out different input modules on the same machine (e.g., using the harmonograph, Etch-a-Sketch interface, and game controller all to control the same plotter). In both cases, the designer can perform these tasks without modifying any code in the modules, and, for the plotter, without modifying any code in the machine controller either. While our intended audience for Stepdance is HCI digital fabrication researchers who likely have some programming experience, we envision a future iteration of these modules consisting of plug-and-play physical interfaces and a purpose-built CNC mechanism that could support audiences without programming experience.

The modularity of Stepdance opens the door to future forms of CNC rapid prototyping with minimal software development requirements. In considering playful ways to work with CNCs, applying two different user interfaces to the same machine creates an opportunity for "multiplayer" digital fabrication, where two users work collaboratively to realize an artifact, or in opposition to thwart the other operator's actions. In more practical applications, we could use the input ports on the Machine Controller module to chain multiple CNC machines together, allowing operation via a single control interface and enabling the production of multiple copies from a single input. Alternatively, we could split the output of an interface across two different CNC machines, using an audio jack splitter, and enable an operator to provide a single input to machines that complement each other's processes, e.g., laser cutting the inlay pattern and CNC milling the corresponding inlay pattern for a decorative wooden part.

7 Conclusion

We presented Stepdance, a system for creative motion control. Through a craft residency, we used Stepdance to reconfigure commercial CNC machines, drawing inspiration from physical art and craft tools to support direct, real-time control. While the technical contribution of our work is a hardware platform that supports CNC production, we argue that the broader point to take from our research is that the CNC technologies alone are *not* the most significant outcome of the Stepdance system. Rather, the power of Stepdance arguably lies in its ability to produce machines that encourage dialog with both digital fabrication enthusiasts and skeptics. By quickly prototyping new CNC machines using metaphors from physical interactions outside digital practice, we can challenge the digital fabrication status quo and imagine powerful ways of making with our hands and machines.

Acknowledgments

This work was partially supported by the National Science Foundation, Grant Number 2441766, and a gift from TTS Tooltechnic Systems. Any opinions, findings, and conclusions in this material are those of the authors and do not necessarily reflect the views of funding institutions. This work was conducted as a part of the Haystack Fab Lab Residency. The Residency is supported by Haystack's Program Endowment, with additional operating support provided by individual donors and granting agencies. We are grateful to the Haystack organizers who organized and facilitated our residency, including Executive Director Perry Price, Technology Director James Rutter, Fab Lab Coordinator Phoebe Zildjian, and Communications Coordinator Kate Lochner. We thank Masa Sasaki and the artists within the ceramics workshop for inspiring us and engaging with our research. We extend our gratitude to all Haystack community members and Haystack staff, particularly the kitchen staff. We would also like to thank Jake Read, Quentin Bolsee, Nadya Peek, and Neil Gershenfeld for their inspirational prior work on flexible machine control systems.

References

[1] Arduino. [n. d.]. Arduino Official Website. www.arduino.cc. Accessed: December 03 2025.

[2] Thomas Ball, Peli de Halleux, James Devine, Steve Hodges, and Michał Moskal. 2024. Jacdac: Service-Based Prototyping of Embedded Systems. *Proc. ACM Program. Lang.* 8, PLDI, Article 175 (June 2024), 24 pages. doi:10.1145/3656405

[3] Samuelle Bourgault, Alejandro Aponte, Megumi Ondo, Emilie Yu, and Jennifer Jacobs. 2025. WORM: Programming Collaborative Robots Through Manual Actions for Craft-Aligned Digital Fabrication. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)* (Busan, South Korea) *(UIST '25)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3746059.3747616

[4] Sam Bourgault and Jennifer Jacobs. 2024. Millipath: Bridging Materialist Theory and System Development for Surface Texture Fabrication. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference (DIS '24)*. Association for Computing Machinery, New York, NY, USA, 50–68. doi:10.1145/3643834.3661599

[5] Samuelle Bourgault, Pilar Wiley, Avi Farber, and Jennifer Jacobs. 2023. CoilCAM: Enabling Parametric Design for Clay 3D Printing Through an Action-Oriented Toolpath Programming System. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23)*. Association for Computing Machinery, New York, NY, USA, 1–16. doi:10.1145/3544548.3580745

[6] Adrian Bowyer. 2014. 3D Printing and Humanity's First Imperfect Replicator. *3D Printing and Additive Manufacturing* 1, 1 (March 2014), 4–5. doi:10.1089/3dp.

2013.0003

[7] Laura Devendorf, Katya Arquilla, Sandra Wirtanen, Allison Anderson, and Steven Frost. 2020. Craftspeople as Technical Collaborators: Lessons Learned through an Experimental Weaving Residency. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3313831.3376820

[8] Laura Devendorf and Daniela K. Rosner. 2017. Beyond Hybrids: Metaphors and Margins in Design. In *Proceedings of the 2017 Conference on Designing Interactive Systems (DIS '17)*. Association for Computing Machinery, New York, NY, USA, 995–1000. doi:10.1145/3064663.3064705

[9] Duet3D. 2023. Duet 3 Mainboard 6XD. https://docs.duet3d.com/Duet3D_hardware/Duet_3_family/Duet_3_Mainboard_6XD_Hardware_Overview

[10] Frikk Fossdal, Rogardt Heldal, and Nadya Peek. 2021. Interactive Digital Fabrication Machine Control Directly Within a CAD Environment. In *Proceedings of the 6th Annual ACM Symposium on Computational Fabrication (SCF '21)*. Association for Computing Machinery, New York, NY, USA, 1–15. doi:10.1145/3485114.3485120

[11] Frikk H. Fossdal, Jens Dyvik, Jakob Anders Nilsson, Jon Nordby, Torbjørn Nordvik Helgesen, Rogardt Heldal, and Nadya Peek. 2020. Fabricatable Machines: A Toolkit for Building Digital Fabrication Machines. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction (TEI '20)*. Association for Computing Machinery, New York, NY, USA, 411–422. doi:10.1145/3374920.3374929

[12] Frikk H Fossdal, Rogardt Heldal, Jens Dyvik, and Adrian Rutle. 2020. A parametric model for creating customized fabrication machines. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS '20)*. Association for Computing Machinery, New York, NY, USA, 143–153. doi:10.1145/3365438.3410960

[13] Frikk H Fossdal, Vinh Nguyen, Rogardt Heldal, Corie L. Cobb, and Nadya Peek. 2023. Vespidae: A Programming Framework for Developing Digital Fabrication Workflows. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference (DIS '23)*. Association for Computing Machinery, New York, NY, USA, 2034–2049. doi:10.1145/3563657.3596106

[14] Devon Frost, Raina Lee, Eun-ha Paek, and Jennifer Jacobs. 2024. SketchPath: Using Digital Drawing to Integrate the Gestural Qualities of Craft in CAM-Based Clay 3D Printing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA.

[15] Madeline Gannon. 2016. Mimus. https://atonaton.com/mimus

[16] N.A. Gershenfeld. 2005. *Fab: The Coming Revolution on Your Desktop–from Personal Computers to Personal Fabrication*. Basic Books.

[17] Neil Gershenfeld, Quentin Bolsée, and Robert Hart. 2022. Urumbu: Minimal machine building. In *Proceedings of the 7th Annual ACM Symposium on Computational Fabrication*. 1–2.

[18] GRBL. 2025. Grbl CNC controller. https://github.com/grbl/grbl original-date: 2009-01-24T23:47:13Z.

[19] Saul Greenberg and Chester Fitchett. 2001. Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*. ACM, Orlando Florida, 209–218. doi:10.1145/502348.502388

[20] Xinyue Gui, Ding Xia, Wang Gao, Mustafa Doga Dogan, Maria Larsson, and Takeo Igarashi. 2025. Draw2Cut: Direct On-Material Annotations for CNC Milling. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, 1–17. doi:10.1145/3706598.3714281

[21] Björn Hartmann, Loren Yu, Abel Allison, Yeonsoo Yang, and Scott R. Klemmer. 2008. Design as exploration: creating interface alternatives through parallel authoring and runtime tuning. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. ACM, Monterey CA USA, 91–100. doi:10.1145/1449715.1449732

[22] Shiqing He and Eytan Adar. 2020. Plotting with Thread: Fabricating Delicate Punch Needle Embroidery with X-Y Plotters. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference (DIS '20)*. Association for Computing Machinery, Eindhoven, Netherlands, 1047–1057. doi:10.1145/3357236.3395540

[23] Mare Hirsch, Gabrielle Benabdallah, Jennifer Jacobs, and Nadya Peek. 2023. Nothing Like Compilation: How Professional Digital Fabrication Workflows Go Beyond Extruding, Milling, and Machines. *ACM Transactions on Computer-Human Interaction* (July 2023). doi:10.1145/3609328 Just Accepted.

[24] Stephanie Houde and Charles Hill. 1997. What do Prototypes Prototype? In *Handbook of Human-Computer Interaction* (2nd ed.). Elsevier Science. Editors: M. Helander, T.Landauer, and P. Prabhu.

[25] Terje Io. 2025. grblHAL. https://github.com/grblHAL

[26] Daphna Kaplan, Ronia Lancer, and Yoav Sterman. 2024. CeramiStick: a joystick controller for real-time interaction and reproduction with a ceramic 3D printer. In *Adjunct Proceedings of the 9th ACM Symposium on Computational Fabrication (SCF Adjunct '24)*. Association for Computing Machinery, New York, NY, USA, 1–3. doi:10.1145/3665662.3673266

[27] Jeeeun Kim, Clement Zheng, Haruki Takahashi, Mark D Gross, Daniel Ashbrook, and Tom Yeh. 2018. Compositional 3D printing: expanding & supporting workflows towards continuous fabrication. In *Proceedings of the 2nd Annual ACM Symposium on Computational Fabrication (SCF '18)*. Association for Computing Machinery, New York, NY, USA, 1–10. doi:10.1145/3213512.3213518

[28] Klipper. 2025. Klipper documentation. https://www.klipper3d.org/

[29] Sophie Landwehr Sydow, Martin Jonsson, and Jakob Tholander. 2022. Modding the Pliable Machine: Unpacking the Creative and Social Practice of Upkeep at the Makerspace. In *Proceedings of the 14th Conference on Creativity and Cognition (C&C '22)*. Association for Computing Machinery, New York, NY, USA, 220–233. doi:10.1145/3527927.3532804

[30] Eldy S. Lazaro Vasquez, Mirela Alistar, Laura Devendorf, and Michael L. Rivera. 2024. Desktop Biofibers Spinning: An Open-Source Machine for Exploring Biobased Fibers and Their Application Towards Sustainable Smart Textile Design. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–18. doi:10.1145/3613904.3642387

[31] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. 2018. Evaluation Strategies for HCI Toolkit Research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–17. doi:10.1145/3173574.3173610

[32] Roby Lynn, Moneer Helu, Mukul Sati, Tommy Tucker, and Thomas Kurfess. 2020. The State of Integrated CAM/CNC Control Systems: Prior Developments and the Path Towards a Smarter CNC. *Smart and sustainable manufacturing systems* 4, 2 (July 2020), 10.1520/SSMS20190046. doi:10.1520/SSMS20190046

[33] MarlinFirmware. 2025. Home. https://marlinfw.org/

[34] M. McCullough. 1998. *Abstracting Craft: The Practiced Digital Hand*. MIT Press. https://books.google.com/books?id=PcwH1WricJEC

[35] Daniela Mitterberger, Selen Ercan Jenny, Lauren Vasey, Ena Lloret-Fritschi, Petrus Aejmelaeus-Lindström, Fabio Gramazio, and Matthias Kohler. 2022. Interactive Robotic Plastering: Augmented Interactive Design and Fabrication for On-site Robotic Plastering. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1–18. doi:10.1145/3491102.3501842

[36] mjs513. [n. d.]. Teensy 4.x Quad Encoder Library. https://github.com/mjs513/Teensy-4.x-Quad-Encoder-Library

[37] Ilan Moyer, Sam Bourgault, Devon Frost, and Jennifer Jacobs. 2024. Throwing Out Conventions: Reimagining Craft-Centered CNC Tool Design through the Digital Pottery Wheel. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA.

[38] Ilan Ellison Moyer. [n. d.]. A Gestalt Framework for Virtual Machine Control of Automated Tools. ([n. d.]).

[39] Ilan E Moyer, Samuelle Bourgault, Devon Frost, and Jennifer Jacobs. 2024. Don't Mesh Around: Streamlining Manual-Digital Fabrication Workflows with Domain-Specific 3D Scanning. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology (UIST '24)*. Association for Computing Machinery, New York, NY, USA, 1–16. doi:10.1145/3654777.3676385

[40] Martin Nisser, Christina Chen Liao, Yuchen Chai, Aradhana Adhikari, Steve Hodges, and Stefanie Mueller. 2021. LaserFactory: A Laser Cutter-based Electromechanical Assembly and Fabrication Platform to Make Functional Devices & Robots. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–15. doi:10.1145/3411764.3445692

[41] David F. Noble. 1984. *Forces of production: a social history of industrial automation* (1st ed.). Knopf, New York.

[42] Nadya Peek, James Coleman, Ilan Moyer, and Neil Gershenfeld. 2017. Cardboard Machine Kit: Modules for the Rapid Prototyping of Rapid Prototyping Machines. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, New York, NY, USA, 3657–3668. doi:10.1145/3025453.3025491

[43] Huaishu Peng, Jimmy Briggs, Cheng-Yao Wang, Kevin Guo, Joseph Kider, Stefanie Mueller, Patrick Baudisch, and François Guimbretière. 2018. RoMA: Interactive Fabrication with Augmented Reality and a Robotic 3D Printer. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. doi:10.1145/3173574.3174153

[44] LLC. PJRC.COM. 2025. Teensy 4.1. https://www.pjrc.com/store/teensy41.html

[45] 3D Potter. 2023. 3D PotterBot 10 PRO Real Clay 3D Ceramic Printer. https://3dpotter.com/printers/potterbot-10-pro

[46] Jake Robert Read, Leo Mcelroy, Quentin Bolsee, B Smith, and Neil Gershenfeld. 2023. Modular-Things: Plug-and-Play with Virtualized Hardware. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems (CHI EA '23)*. Association for Computing Machinery, New York, NY, USA, 1–6. doi:10.1145/3544549.3585642

[47] Jake Robert Read, Nadya Peek, and Neil Gershenfeld. 2023. MAXL: Distributed Trajectories for Modular Motion. In *Proceedings of the 8th ACM Symposium on*

Computational Fabrication (SCF '23). Association for Computing Machinery, New York, NY, USA, 1–11. doi:10.1145/3623263.3623362
[48] Mitchel Resnick and Eric Rosenbaum. 2013. Designing for tinkerability. In *Design, make, play*. Routledge, 163–181.
[49] Michael L. Rivera and Scott E. Hudson. 2019. Desktop Electrospinning: A Single Extruder 3D Printer for Producing Rigid Plastic and Electrospun Textiles. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. doi:10.1145/3290605.3300434
[50] Alec Rivers, Ilan E. Moyer, and Frédo Durand. 2012. Position-correcting tools for 2D digital fabrication. *ACM Transactions on Graphics* 31, 4 (July 2012), 88:1–88:7. doi:10.1145/2185520.2185584
[51] Evil Mad Scientist. [n. d.]. AxiDraw V3. https://shop.evilmadscientist.com/846
[52] Ltd. Shenzhen BIQU Technology Co. [n. d.]. BIGTREETECH TMC2209 V1.2. https://github.com/bigtreetech/BIGTREETECH-TMC2209-V1.2
[53] Daniel Shiffman. 2017. Lissajous Curve Table. https://thecodingtrain.com
[54] Roy Shilkrot, Pattie Maes, and Amit Zoran. 2014. Physical rendering with a digital airbrush. In *ACM SIGGRAPH 2014 Studio (SIGGRAPH '14)*. Association for Computing Machinery, New York, NY, USA, 1. doi:10.1145/2619195.2656328
[55] Monica Silva Lovato, Jeff Suina, Jared Tso, Alexis Kaminsky, Camila Friedman-Gerlicz, and Leah Buechley. 2025. American Indian Pottery and Clay 3D Printing: An Exploration of Opportunities and Risks in Professional Practice. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, 1–23. doi:10.1145/3706598.3713159
[56] Blair Subbaraman, Nathaneal Bursch, and Nadya Peek. 2025. It's Not the Shape, It's the Settings: Tools for Exploring, Documenting, and Sharing Physical Fabrication Parameters in 3D Printing. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (CHI '25)*. Association for Computing Machinery, New York, NY, USA, 1–19. doi:10.1145/3706598.3713354
[57] Blair Subbaraman and Nadya Peek. 2022. p5.fab: Direct Control of Digital Fabrication Machines from a Creative Coding Environment. In *Proceedings of the 2022 ACM Designing Interactive Systems Conference (DIS '22)*. Association for Computing Machinery, New York, NY, USA, 1148–1161. doi:10.1145/3532106.3533496
[58] Blair Subbaraman and Nadya Peek. 2023. 3D Printers Don't Fix Themselves: How Maintenance is Part of Digital Fabrication. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference (DIS '23)*. Association for Computing Machinery, New York, NY, USA, 2050–2065. doi:10.1145/3563657.3595991
[59] Rundong Tian and Eric Paulos. 2021. Adroid: Augmenting Hands-on Making with a Collaborative Robot. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21)*. Association for Computing Machinery, New York, NY, USA, 270–281. doi:10.1145/3472749.3474749
[60] Rundong Tian, Vedant Saran, Mareike Kritzler, Florian Michahelles, and Eric Paulos. 2019. Turn-by-Wire: Computationally Mediated Physical Fabrication. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. Association for Computing Machinery, New Orleans, LA, USA, 713–725. doi:10.1145/3332165.3347918
[61] Mert Toka, Devon Frost, Samuelle Bourgault, Avi Farber, Camila Friedman-Gerlicz, Raina Lee, Eun-Ha Paek, Pilar Wiley, and Jennifer Jacobs. 2024. Practice-driven Software Development: A Collaborative Method for Digital Fabrication Systems Research in a Residency Program. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference (DIS '24)*. Association for Computing Machinery, New York, NY, USA, 1192–1217. doi:10.1145/3643834.3661522
[62] Jasper Tran O'Leary, Thrisha Ramesh, Octi Zhang, and Nadya Peek. 2024. Tandem: Reproducible Digital Fabrication Workflows as Multimodal Programs. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (CHI '24)*. Association for Computing Machinery, New York, NY, USA, 1–16. doi:10.1145/3613904.3642751
[63] Hannah Twigg-Smith, Jasper Tran O'Leary, and Nadya Peek. 2021. Tools, Tricks, and Hacks: Exploring Novel Digital Fabrication Workflows on #PlotterTwitter. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–15. doi:10.1145/3411764.3445653
[64] Hannah Twigg-Smith and Nadya Peek. 2023. Dynamic Toolchains: Software Infrastructure for Digital Fabrication Workflows. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*. Association for Computing Machinery, New York, NY, USA, 1–20. doi:10.1145/3586183.3606802
[65] Joshua Vasquez, Hannah Twigg-Smith, Jasper Tran O'Leary, and Nadya Peek. 2020. Jubilee: An Extensible Machine for Multi-tool Fabrication. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. doi:10.1145/3313831.3376425
[66] Bret Victor. [n. d.]. Learnable Programming. https://worrydream.com/LearnableProgramming/
[67] Karl D.D. Willis, Cheng Xu, Kuan-Ju Wu, Golan Levin, and Mark D. Gross. 2010. Interactive fabrication: new interfaces for digital fabrication. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction (TEI '11)*. Association for Computing Machinery, New York, NY, USA, 69–72. doi:10.1145/1935701.1935716
[68] Shanjiang Zhang and Guanglu Liu. 2024. Design of a multipurpose SCARA-like parallel robot. In *2024 4th International Symposium on Artificial Intelligence and Intelligent Manufacturing (AIIM)*. IEEE, 18–21.
[69] John Zimmerman, Jodi Forlizzi, and Shelley Evenson. 2007. Research through design as a method for interaction design research in HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. Association for Computing Machinery, New York, NY, USA, 493–502. doi:10.1145/1240624.1240704
[70] Amit Zoran and Joseph A. Paradiso. 2013. FreeD: A Freehand Digital Sculpting Tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2613–2616. doi:10.1145/2470654.2481361 event-place: Paris, France.

A Stepdance SSL Implementation Details

A.1 Component Execution

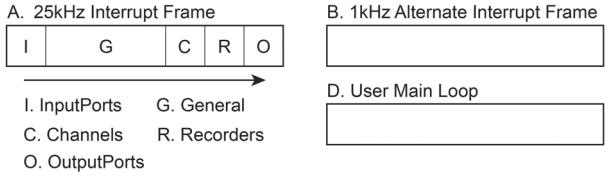

Figure 22: Stepdance Component Execution

To ensure correct maintenance and update of position data, Stepdance components run in one of three contexts, shown in Figure 22. Most components run in the primary 25kHz interrupt frame (Fig. 22A). Within this frame, execution occurs in five phases. The first *input phase* (Fig. 22I) runs InputPortand Playback components, which have no internal dependencies. The second *pre-channel phase* (Fig. 22G) runs the majority of the component types, in the order they are instantiated in a program. The third *channel phase* (Fig. 22C) runs the Channel components. The fourth *post-channel phase* (Fig. 22R) runs Recorder components, ensuring after channels state is updated. A final *output phase* (Fig. 22O) runs the OutputPort components, transmitting a motion stream frame. Components such as buttons, which run in the background but don't require tight timing, execute in a 1kHz alternative interrupt frame (Fig. 22B). Some components, such as the interpreters, have elements that run in the main program loop (Fig. 22D) before the user loop executes.

A.2 Input Implementation

We extend or modify existing embedded programming input types to be compatible with the dataflow structure and performance requirements of Stepdance. For AnalogInputs, we re-implemented Arduino analogIn functions to run as interrupt-driven background processes. For digital inputs, we implement an independent 1khz alternative interrupt frame to support peripherals that don't require the standard 25kHz update rate. Digital inputs are read during this alternative frame, debounced, and can be read directly or used to generate function callbacks. We support quadrature encoders using the Teensy 4.x Quad Encoder Library [36], which we wrap in an SSL component running in the interrupt frame.

B Sample StepDance Programs

B.1 Step-a-Sketch Code

```
#define module_driver
#include "stepdance.hpp"

// -- Define Output Ports --
OutputPort output_a;  // Axidraw left motor
OutputPort output_b;  // Axidraw right motor
OutputPort output_c;  // Z axis, a servo driver for the AxiDraw

// -- Define Motion Channels --
Channel channel_a;  //AxiDraw "A" axis --> left motor motion
Channel channel_b;  // AxiDraw "B" axis --> right motor motion
Channel channel_z;  // AxiDraw "Z" axis --> pen up/down

// -- Define Kinematics --
KinematicsCoreXY axidraw_kinematics;

// -- Define Encoders --
Encoder encoder_1;  // left knob, controls horizontal
Encoder encoder_2;  // right knob, controls vertical

// -- Define Input Button --
Button button_d1;

// -- Position Generator for Pen Up/Down --
PositionGenerator position_gen;

void setup() {
  // -- Configure and start the output ports --
  output_a.begin(OUTPUT_A);
  output_b.begin(OUTPUT_B);
  output_c.begin(OUTPUT_C);

  // Enable the output drivers
  enable_drivers();

  // -- Configure and start the channels --
  channel_a.begin(&output_a, SIGNAL_E);
  channel_a.set_ratio(25.4, 2874); // Sets the input/output transmission ratio for the channel.

  channel_b.begin(&output_b, SIGNAL_E);
  channel_b.set_ratio(25.4, 2874);

  channel_z.begin(&output_c, SIGNAL_E); //servo motor
  channel_z.set_ratio(1, 1); //straight step pass-thru.

  // -- Configure and start the encoders --
  encoder_1.begin(ENCODER_1);
  encoder_1.set_ratio(24, 2400);
  encoder_1.output.map(&axidraw_kinematics.input_x);

  encoder_2.begin(ENCODER_2);
  encoder_2.set_ratio(24, 2400);
  encoder_2.output.map(&axidraw_kinematics.input_y);

  // -- Configure and start the kinematics module --
  axidraw_kinematics.begin();
  axidraw_kinematics.output_a.map(&channel_a.input_target_position);
  axidraw_kinematics.output_b.map(&channel_b.input_target_position);

  // -- Configure Button --
  button_d1.begin(IO_D1, INPUT_PULLDOWN);
  button_d1.set_mode(BUTTON_MODE_TOGGLE);
  button_d1.set_callback_on_press(&pen_down);
  button_d1.set_callback_on_release(&pen_up);

  // -- Configure Position Generator --
  position_gen.output.map(&channel_z.input_target_position);
  position_gen.begin();

  // -- Start the stepdance library --
  dance_start();
}

void loop() {
  dance_loop(); // Stepdance loop provides convenience functions, and should be called at the end of the main loop
}

void pen_down(){
  position_gen.go(-200, ABSOLUTE, 2000);
}

void pen_up(){
  position_gen.go(200, ABSOLUTE, 2000);
}
```

B.2 Wheelbot Code

```
#define module_driver

#include "stepdance.hpp"

OutputPort output_a;
OutputPort output_b;
```

```
OutputPort output_c;
OutputPort output_d;

Channel channel_a;
Channel channel_b;
Channel channel_z;
Channel channel_e;

KinematicsPolarToCartesian polar_kinematics;

AnalogInput analog_a1; //foot pedal
AnalogInput analog_a2; //rotary pot
AnalogInput analog_a3; //rotary pot

Encoder encoder_1; //hand lever
Encoder encoder_2; //z babystep

VelocityGenerator velocity_gen;

ScalingFilter1D z_gen; //generates z signal
ScalingFilter1D x_stretch;

PathLengthGenerator2D e_gen; //generates extruder signal

float64_t layerHeight = 2.0;
float64_t nozzleDiameter = 4.0;

void setup() {

  output_a.begin(OUTPUT_A);
  output_b.begin(OUTPUT_B);
  output_c.begin(OUTPUT_C);
  output_d.begin(OUTPUT_D);

  enable_drivers();

  channel_a.begin(&output_a, SIGNAL_E);
  channel_a.set_ratio(1, 40);
  channel_a.enable_filtering(200);

  channel_b.begin(&output_b, SIGNAL_E);
  channel_b.set_ratio(1, 40);
  channel_b.enable_filtering(200);

  channel_z.begin(&output_c, SIGNAL_E);
  channel_z.set_ratio(1, 1201); // testing with an 8mm lead leadscrew
  channel_z.invert_output();

  channel_e.begin(&output_d, SIGNAL_E);
  channel_e.set_ratio(1, 70); // testing with an 8mm lead leadscrew
  channel_e.invert_output();
  channel_e.enable_filtering(1000);

  velocity_gen.begin();
  velocity_gen.output.map(&polar_kinematics.input_angle);

  encoder_1.begin(ENCODER_1);
  encoder_1.set_ratio(1, 2400); //1mm per revolution
  encoder_1.output.map(&polar_kinematics.input_radius);

  encoder_2.begin(ENCODER_2);
  encoder_2.set_ratio(1, 2400); //1mm per revolution
  encoder_2.output.map(&channel_z.input_target_position);

  polar_kinematics.output_x.map(&channel_a.input_target_position);
  polar_kinematics.output_x.map(&x_stretch.input);

  polar_kinematics.output_y.map(&channel_b.input_target_position);
  polar_kinematics.begin();

  z_gen.begin();
  z_gen.set_ratio(layerHeight, TWO_PI); //raise a distance of a single layer for 1 revolution
  z_gen.input.map(&polar_kinematics.input_angle);
  z_gen.output.map(&channel_z.input_target_position);

  x_stretch.begin(ABSOLUTE);
  x_stretch.output.map(&channel_a.input_target_position);

  e_gen.begin();
  e_gen.input_1.map(&channel_a.input_target_position);
  e_gen.input_2.map(&channel_b.input_target_position);
  e_gen.output.map(&channel_e.input_target_position);

  //pedal
  analog_a1.set_floor(0, 25);
  analog_a1.set_ceiling(2.75, 1020);
  analog_a1.map(&velocity_gen.speed_units_per_sec);
  analog_a1.begin(IO_A1);

  //extrusion multiplier knob
  analog_a2.set_floor(0, 25);
  analog_a2.set_ceiling(10, 1020);
  analog_a1.begin(IO_A2);

  //x-stretch knob
  analog_a3.set_floor(0, 25);
```

```
  analog_a3.set_ceiling(100, 1020);
  analog_a3.map(&velocity_gen.speed_units_per_sec
  ↪ );
  analog_a3.begin(IO_A3);

  dance_start();
}


void loop() {

  extrusionMultiplier = analog_a2.read();
  extrusionRate = (4*layerHeight *
  ↪ extrusionMultiplier * nozzleDiameter) /
  ↪ (PI*nozzleDiameter*nozzleDiameter);
  e_gen.set_ratio(extrusionRate);

  x_stretch.set_ratio(analog_a3.read());

  dance_loop();
}
```

B.3 Stencil Maker Code

```
#include <SD.h>
#define module_driver
#include "stepdance.hpp"

// Pantograph XY input plugged in input_a
InputPort input_a;

// Ender XYZE plugged into output ports
OutputPort output_a;
OutputPort output_b;
OutputPort output_c;
OutputPort output_d;

// Corresponding channels for the Ender
Channel channel_a;
Channel channel_b;
Channel channel_z;
Channel channel_e;

// Knobs input
Encoder encoder_1;
Encoder encoder_2;

// Slider input
AnalogInput analog_a1; //extrusion rate controller

// -- Define Input Button --
Button button_d1; // start/stop recording
Button button_d2; // start/stop replay

PositionGenerator start_position_gen_x; // To
↪ bridge gap between end/start of layer path
PositionGenerator start_position_gen_y; // To
↪ bridge gap between end/start of layer path
PositionGenerator z_gen;
// PathLengthGenerator2D z_gen; //generates z
↪ signal
PathLengthGenerator2D e_gen; //generates extruder
↪ signal

// -- Recorder --
FourTrackRecorder recorder;
FourTrackPlayer player;

// Should continuously replay?
bool replay_until_stopped = false;
bool waiting_to_start_play = false;

// Start position of layer
float64_t start_pos_x;
float64_t start_pos_y;

float64_t layerHeight = 0.5;
float64_t nozzleDiameter = 1.0;
volatile float64_t extrusionRate = 0;

void setup() {
  input_a.begin(INPUT_A);

  input_a.output_x.set_ratio(0.01, 1); //1 step is
  ↪ 0.01mm
  input_a.output_x.map(&channel_a.input_target_po
  ↪ sition);

  input_a.output_y.set_ratio(0.01, 1); //1 step is
  ↪ 0.01mm
  input_a.output_y.map(&channel_b.input_target_po
  ↪ sition);

  output_a.begin(OUTPUT_A);
  output_b.begin(OUTPUT_B);
  output_c.begin(OUTPUT_C);
  output_d.begin(OUTPUT_D);

  enable_drivers();

  // We took these ratios from default Ender
  ↪ steps/mm settings in menu
  channel_a.begin(&output_a, SIGNAL_E);
  channel_a.set_ratio(1, 80);
  channel_a.enable_filtering(200);

  channel_b.begin(&output_b, SIGNAL_E);
  channel_b.set_ratio(1, 80);
  channel_b.invert_output();
  channel_b.enable_filtering(200);

  channel_z.begin(&output_c, SIGNAL_E);
  channel_z.set_ratio(1, 400);
```

```
  channel_z.invert_output();

  channel_e.begin(&output_d, SIGNAL_E);
  channel_e.set_ratio(1, 93);

  encoder_1.begin(ENCODER_1);
  encoder_1.set_ratio(1, 2400);
  encoder_1.output.map(&channel_e.input_target_po⌋
  ↪ sition);
  encoder_1.invert();

  encoder_2.begin(ENCODER_2);
  encoder_2.set_ratio(10, 2400);
  encoder_2.output.map(&channel_z.input_target_po⌋
  ↪ sition);
  encoder_2.invert();

  //extrusion knob
  analog_a1.set_floor(0, 25);
  analog_a1.set_ceiling(2, 1020);
  analog_a1.begin(IO_A1);

  z_gen.output.map(&channel_z.input_target_positi⌋
  ↪ on);
  z_gen.begin();

  e_gen.begin();
  e_gen.input_1.map(&channel_a.input_target_posit⌋
  ↪ ion);
  e_gen.input_2.map(&channel_b.input_target_posit⌋
  ↪ ion);
  e_gen.output.map(&channel_e.input_target_positi⌋
  ↪ on);

  // -- Configure the Position Generator
  ↪ start_position_gen_x.output.map(&channel_a.⌋
  ↪ input_target_position);
  start_position_gen_x.begin();
  start_position_gen_y.output.map(&channel_b.inpu⌋
  ↪ t_target_position);
  start_position_gen_y.begin();

  button_d1.begin(IO_D1, INPUT_PULLDOWN);
  button_d1.set_mode(BUTTON_MODE_TOGGLE);
  button_d1.set_callback_on_press(&start_recordin⌋
  ↪ g);
  button_d1.set_callback_on_release(&stop_recordi⌋
  ↪ ng);

  button_d2.begin(IO_D2, INPUT_PULLDOWN);
  button_d2.set_mode(BUTTON_MODE_STANDARD);
  button_d2.set_callback_on_press(&playback_contr⌋
  ↪ ol);

  recorder.input_1.map(&channel_a.input_target_po⌋
  ↪ sition);
  recorder.input_2.map(&channel_b.input_target_po⌋
  ↪ sition);
  recorder.begin();

  player.output_1.map(&channel_a.input_target_pos⌋
  ↪ ition);
  player.output_2.map(&channel_b.input_target_pos⌋
  ↪ ition);
  player.begin();

  dance_start();
}


void loop() {
  float64_t extrusionMultiplier = analog_a1.read();
  float64_t segmentLength = 1.0;
  extrusionRate = (4*layerHeight *
  ↪ extrusionMultiplier * nozzleDiameter *
  ↪ segmentLength) /
  ↪ (PI*nozzleDiameter*nozzleDiameter);
  e_gen.set_ratio(extrusionRate);

  if (waiting_to_start_play) {

    float64_t x_dist_to_start_pt = channel_a.inpu⌋
    ↪ t_target_position.read(ABSOLUTE) -
    ↪ start_pos_x;
    float64_t y_dist_to_start_pt = channel_b.inpu⌋
    ↪ t_target_position.read(ABSOLUTE) -
    ↪ start_pos_y;

    if (x_dist_to_start_pt*x_dist_to_start_pt <
    ↪ 0.0001 &&
    ↪ y_dist_to_start_pt*y_dist_to_start_pt <
    ↪ 0.0001) {
      start_playback();
    }
  }

  if (replay_until_stopped &&
  ↪ !waiting_to_start_play &&
  ↪ !player.playback_active) {
    initiate_playback(); // launch another layer

  }

  dance_loop();
}

void start_recording(){
  recorder.start("eazao_layer_sketch");

  // Recording first point value
  start_pos_x = channel_a.input_target_position.r⌋
  ↪ ead(ABSOLUTE);
```

```
  start_pos_y = channel_b.input_target_position.r
  ↪ ead(ABSOLUTE);
}

void stop_recording(){
  recorder.stop();
}

void playback_control(){
  if(player.playback_active){
    stop_playback();
    replay_until_stopped = false;
  }else{
    initiate_playback();
    replay_until_stopped = true;
  }
}

void initiate_playback(){
  // First go to start point
  start_position_gen_x.go(start_pos_x - current_x
  ↪ , INCREMENTAL, 50);
  start_position_gen_y.go(start_pos_y - current_y
  ↪ , INCREMENTAL, 50);

  // Go up a layer (trying it here instead of after
  ↪ start_playback)
  z_gen.go(layerHeight, INCREMENTAL, 50);

  waiting_to_start_play = true;
}

void start_playback(){
  waiting_to_start_play = false;

  // // Go up a layer
  // z_gen.go(layerHeight, INCREMENTAL, 50); //
  ↪ works

  player.start("eazao_layer_sketch");
}

void stop_playback(){
  player.stop();
}
```

C Additional Firmware Schematics

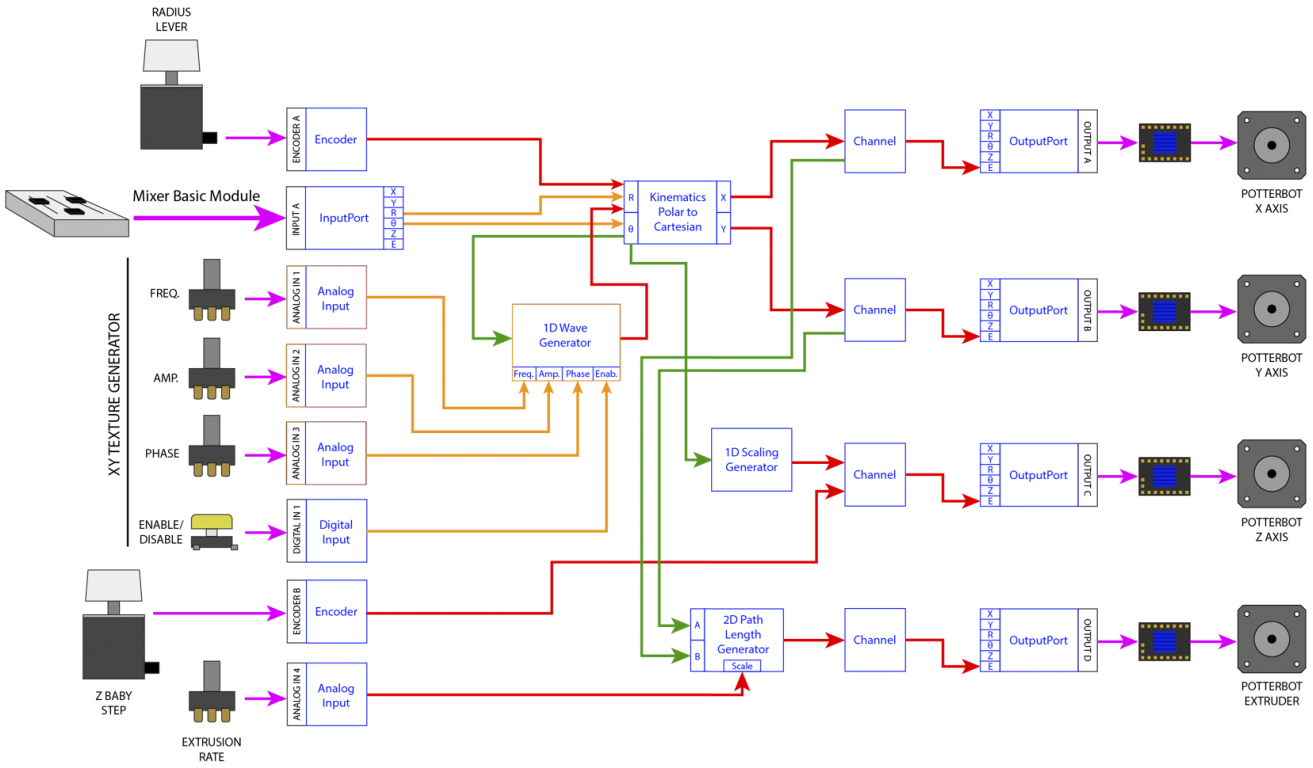

Figure 23: Wheelbot Mixer Firmware Architecture.

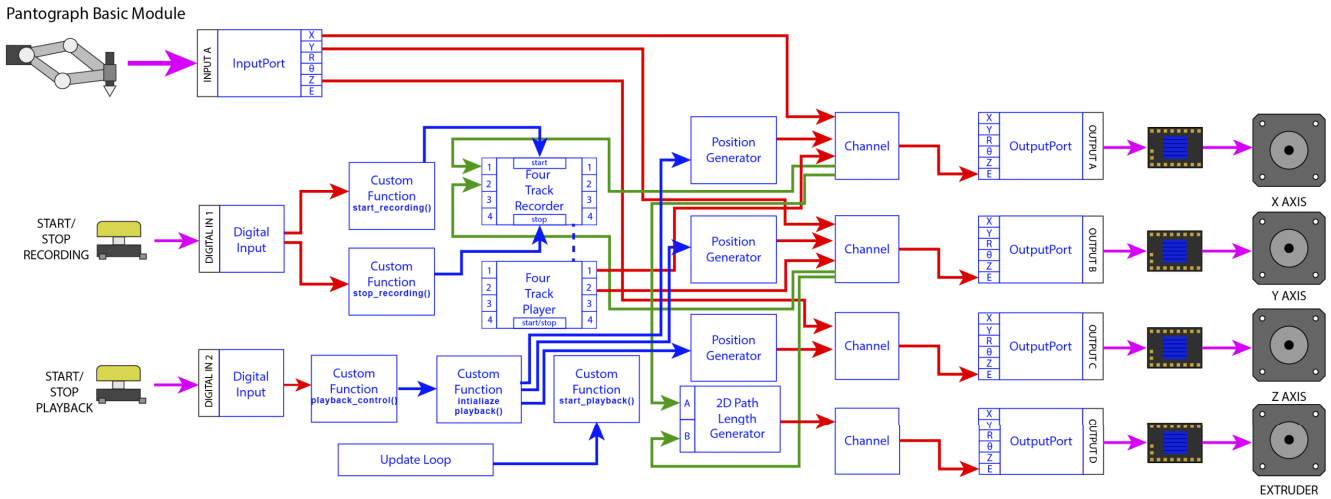

Figure 24: Sketch-based Stencil Extruder Firmware Architecture.