Urumbu: Minimal Machine Building

Neil Gershenfeld
Massachusetts Institute of Technology
USA
neil.gershenfeld@cba.mit.edu

Quentin Bolsée
Vrije Universiteit Brussel
Belgium
quentinbolsee@hotmail.com

Robert Hart
Independent researcher
USA
roberthart56@gmail.com

ABSTRACT

Computational fabrication has enabled the widespread adoption of rapid-prototyping. There has however been a much higher barrier to entry to make a machine. We investigate rapid-prototyping of rapid-prototyping, so that designing a machine can become as accessible as producing projects on one. We characterize minimum viable components for machine building, including "controllerless" designs that virtualize low-level controls in high-level parallel software, simplified systems for force transmission, and tradeoffs in structural systems. In this demo, a machine controlled entirely by a Python application on a host laptop is presented, with communication between the host and the machine at kHz rate. User interaction is enabled by including polling of input modules in the loop. The machine's behavior is entirely software-defined within the host application, and new modules can be dynamically plugged in with no reconfiguration on the firmware side.

KEYWORDS

Rapid-prototyping, Machine building, Control Systems

ACM Reference Format:
Neil Gershenfeld, Quentin Bolsée, and Robert Hart. 2022. Urumbu: Minimal Machine Building. In *Symposium on Computational Fabrication (SCF '22), October 26–28, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3559400.3565599

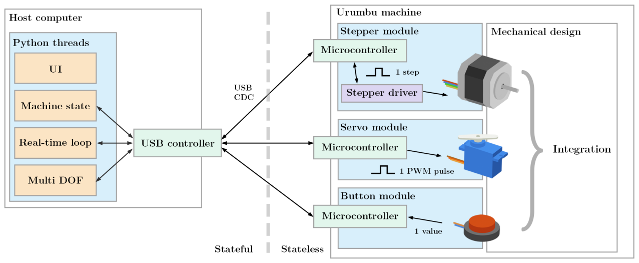

Figure 1: Urumbu architecture: each machine functionality is implemented through a state-less, individual USB device.

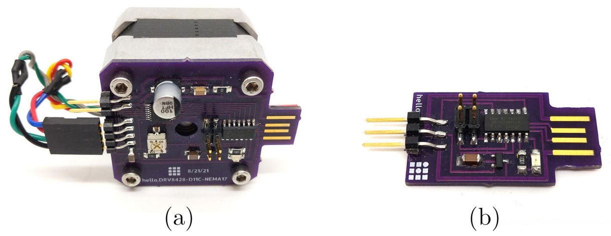

Figure 2: Urumbu USB device examples: stepper driver (a) and servo actuator (b).

1 INTRODUCTION

Widespread access to rapid-prototyping tools has had a transformative impact, however there remains a high barrier to entry to create these tools. Rapid-prototyping of rapid-prototyping would enable machine design to become as expressive and personal as the projects produced on them, but is inhibited by the range of skills required for a full custom design, the constraints imposed by off-the-shelf solutions, and the specialized inventory required. The most common controllers for additive and subtractive machines use a G-code interpreter. This embeds complex state to describe a machine's configuration, which must be changed if anything about the machine changes. It has a fixed set of commands that must be mapped onto any new processes. And it's intended for feed-forward control, not for communicating with machines that are sources as well as sinks of data.

Real-time machine networks can reduce the embedded state, so that degrees of freedom can be added or removed without being limited by assumptions imposed by a fixed controller. We investigate here what can be thought of as the asymptotic limit of this approach, asking how far the interfaces to a machine's actuators and sensors can be simplified, to go directly from high-level goals to low-level control.[1]

Our approach is motivated by two changes in computing architecture since the advent of G-codes. The first is the adoption of multi-core processors in general-purpose computers that can run multiple parallel processes simultaneously, and the second is the appearance of low-cost embedded processors that support high-speed network protocols. Together these allow the logic of machine control to be virtualized in a real-time parallel process, sending the smallest primitive operations to minimal nodes, for example a single step made by a stepper motor, or a single PWM pulse for a servo motor. Along with this simplification it also eliminates the complexity of coordination, since all commands are happening in real time.

[1]The project title "Urumbu" is the Malayalam word for small ants found in the south of India, which are individually simple but collectively perform complex tasks.

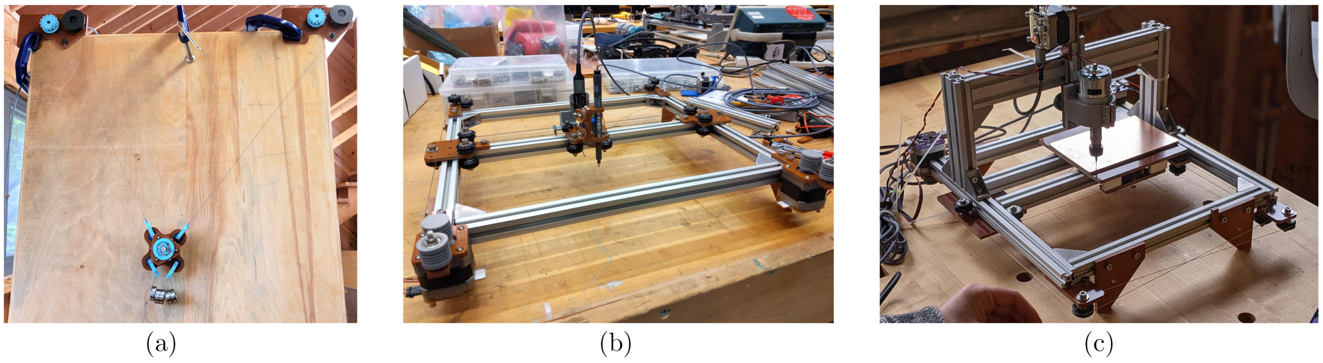

Figure 3: Examples of Urumbu machines: polar plotter (a), CoreXY plotter (b) and CNC mill (c). The use of kevlar thread and 3D printed capstans is a minimalist motion transmission aimed at reducing the cost-of-entry to machine prototyping.

2 SYSTEM DESCRIPTION

The core concept of the Urumbu architecture is to provide a distinct USB module for each functionality of the machine. Each module is simultaneously powered and controlled through USB 2.0, making it trivial to add and remove modules at will. As shown in figure 1, modules are stateless and do not require any configuration. The host computer implements the machine's state and motion planning in real-time by sending USB packets every time an update is needed. We found that packets can be sent out at 10kHz with no significant skew, enabling control of a stepper motor at the microstep level. Each USB packet sent to the motor is minimal, containing a single CDC character ('f' for forward, 'b' for backwards stepping). Another example of module focuses on servo motor actuation by sending a single PWM pulse width as specific by two CDC bytes. Stepper and servo modules are illustrated in figure 2. To enable closed-loop control within the host application, analog or digital input modules can also be realized through polling. Examples include switches, potentiometers, light sensor and magnetometers.

In addition to this highly modular, software defined machine architecture, we present a cheap motion system aimed at lowering the cost of entry to machine prototyping. Minimalist motion transmission is realized by a kevlar thread and 3D printed capstans for tensioning, visible in figure 3 (b). Successful applications of the Urumbu architecture include a polar plotter, coreXY 2D plotter and CNC milling machine for PCB prototyping.

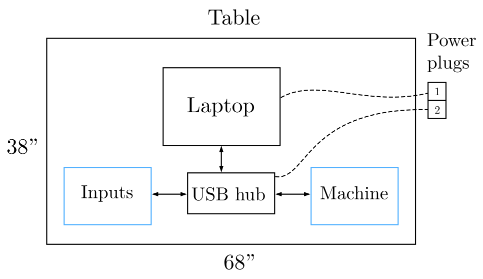

Figure 4: Dimensions of the demo setup.

3 DEMO REQUIREMENTS

The demo will showcase an Urumbu machine actuated by USB modules only (i.e. stepper and servo modules). Figure 3 provides examples of Urumbu machines. The machine's state and motion planning is entirely managed by a Python application running on a typical laptop. User interaction is enabled by input Urumbu modules presented next to the machine, including potentiometers, magnetometers and light sensors. Input states are polled by the Python application and affect the machine's motion, showcasing real-time closed loop control at the OS level. Modules can be plugged in dynamically, proving the high modularity of the system. Moreover, the machine itself highlights cheap, minimalist motion transmission through a kevlar thread and delrin bearing wheels. The physical layout of the demo is presented in figure 4. The following equipment should be put on display onto a standard-sized table:

- A powered USB 2.0 hub (power plug needed)
- The Urumbu machine
- Input modules for user interaction
- A laptop computer (power plug needed)

Optionally, a poster next to the table will illustrate the principles of the Urumbu architecture in more details, listing available modules and machine realizations. There is no specific requirement on the lighting, the table should be exposed to typical room lighting so that users can see the equipment and interact with it.

4 CONCLUSIONS

This demo illustrates rapid prototyping of digital fabrication machines through highly modular USB modules. Each module implements a single functionality and runs on a minimalist stateless firmware, while a host computer sends USB packets in real-time to all modules to enable actuation and input polling. This is an effective example of a software defined machine, with no configuration needed on the firmware side, independently of the machine's finality. By interacting with the demo, users can experience real-time closed loop control entirely managed by a Python application running on a standard laptop. The high modularity of the system is showcased by various input modules available next to the machine; each module can be dynamically plugged in and affects the machine's behavior through the host application.