PaperSignals Parameterized

Nao Ouyang
Laura Zharmukhametova
Jamelle Watson-Daniels
nouyang@g.harvard.edu
lzharmukhametova@college.harvard.edu
jwatsondaniels@g.harvard.edu

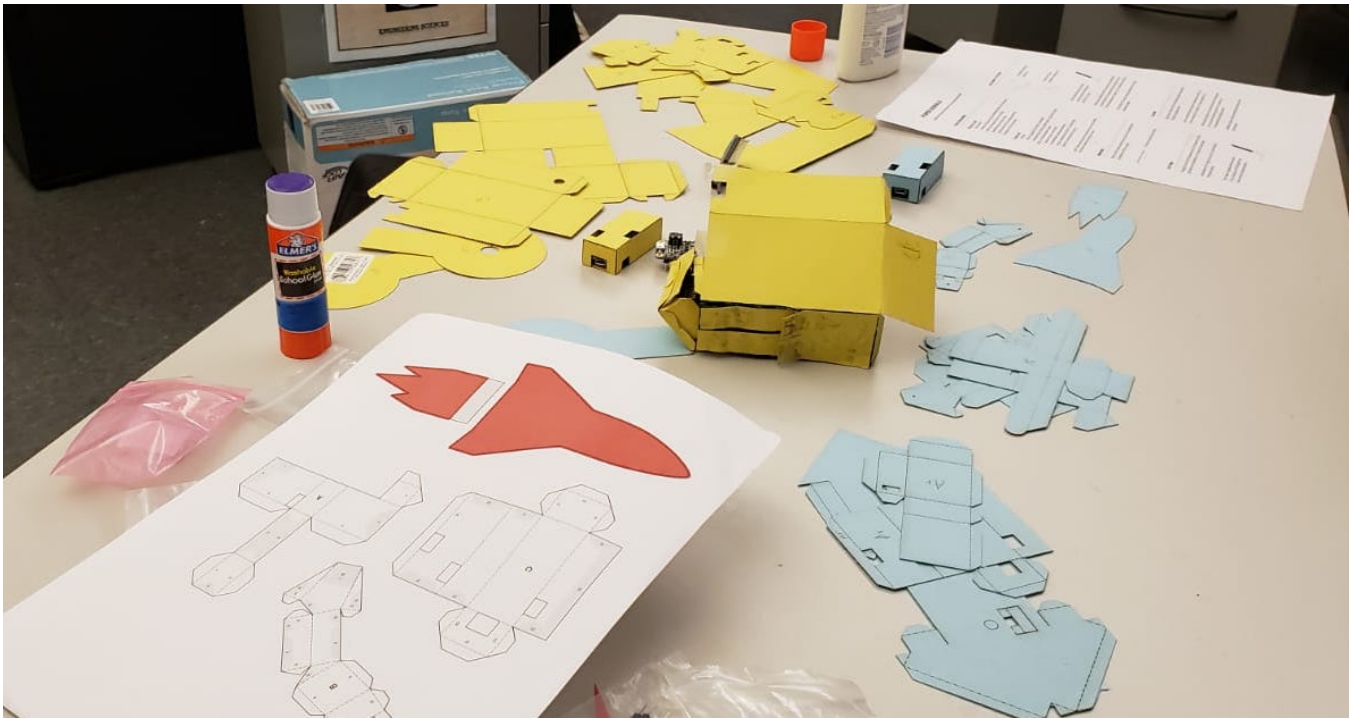


Figure 1: Building PaperSignals.

ABSTRACT

PaperSignals are build-it-yourself printable robots that a user can control with their voice. This study presents an open-source browser-based user interface (written in Python) aimed at allowing users to easily customize existing Papersignal templates. The fabrication process is intended to be inexpensive and accessible to non-technical experts. The tool developed in this work allows users to address two major limitations of PaperSignals templates - change dimensions and produce template components in svg instead of pdf (a more usable file format). This work also takes these static templates, which are not parameterized, and presents a set of design primitives that would inform a domain specific language to fully parameterize this work in the future. Our source code can be found at github and a demo online at heroku.

1 INTRODUCTION

PaperSignals is an experiment by Google that allows non-expert users to fabricate functional robots that act as a tracking device. Users gather a small electronics kit and print paper templates to ultimately assemble the devices. The electronic parts shown in 2 can be ordered online from Adafruit and the template can be downloaded from the website and printed using a standard inkjet printer.

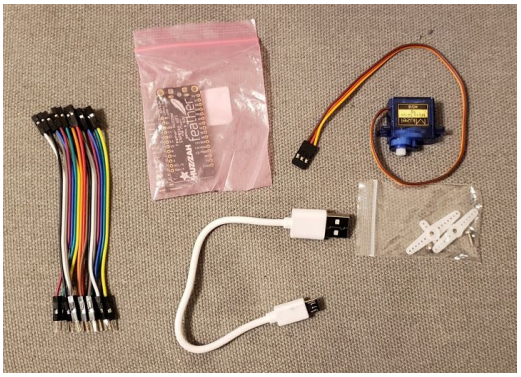


Figure 2: Electronic parts to turn the papercraft into robotic papercraft.

The final PaperSignals devices can be controlled by voice via Google Assistant. To set it up, the user connects to it through WiFi and, after some registration steps, configures the device by saying (via e.g. an Android phone) a phrase such as "Track the weather in Seattle" or "Track rocket launches in China". There are currently six templates available online for users to fabricate these printable robots. The process can take a few hours to complete once a user has all of the necessary components. While the user does not need to be a technical expert, the process does require some experience with programming if the user would like to make changes to the templates given.

This project investigates specific limitations of PaperSignals in an effort to improve the user experience by introducing PaperSignals parameretized, and defines a Domain Specific Language that would facilitate easy configuration of both the robotics and the template parts of the device. Note that we do not cover the programming aspects of the design, unlike other projects, and focus on the paper templating process and how the use of robotic components can change the need thesis.

The structure of the paper begins with discussing the motivation (existing limitations and need thesis) behind this project, followed by a review of academic literature and similar interfaces available online. Next, the design and implementation of this project is covered. The following section presents an initial draft of a domain-specific language (DSL) for this project. Finally, we conclude and discuss future work.

1.1 Limitations of Google PaperSignals

We identified the following limitations of the PaperSignals, through preliminary interviews with Harvard students specializing in art and papercraft, and building PaperSignals using the provided templates.

- Existing templates are not parametric. It is difficult to change dimensions.
- Personalized and customized designs are not possible.
- The template file format cannot be modified.
- If a mistake is made during fabrication, the user cannot undo glued paper without destroying the paper. This problem is particularly important to robotic papercraft as it is difficult to debug an opaque box. See Figures 3 and 4.
- The fabrication process can take hours

The core tension is therefore usability and customizability of existing PaperSignals templates. Consider the limited template file format. Currently, there are only PDFs available, which doesn't allow users to change vectors and dimensions of shapes relative to one another. Additionally, if a user wants to use a laser-cutting machine (which is a reasonable expectation for a user with a robotics background), then alternative file formats are needed. The presented work evaluates the limitations of PaperSignals templates and offers a customizable option for changing dimensions, thus

Figure 3: Arrow was installed incorrectly onto motor, and as the screw was inside the box and the template folded and glued over it, the fix required destructively modifying the box.

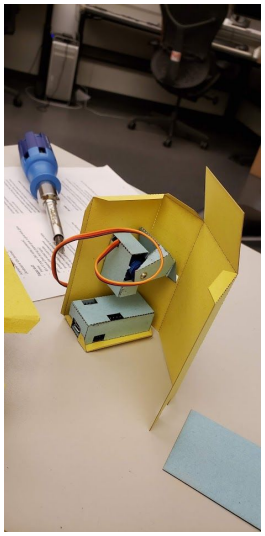


Figure 4: The inside of the pants templates shows the robotic components that must be debugged.

increasing the customizability and usability of existing PaperSignals templates.

1.2 Need Thesis

The listed limitations imply that Google PaperSignals could be greatly improved. The first step of defining the pattern has been taken care of by PaperSignals templates, but issues ensue when a user attempts to produce a functional robot. One issue is difficulty in specifying parameter dimensions and updating the PaperSignals templates accordingly. Additionally, when ordering electronic components, the more affordable options sometimes have different dimensions than the PaperSignals templates. There is no way to change dimensions just using the standard templates and source code provided by Google. Our goal is to develop a parameterized version of one of the PaperSignals templates to address this end-user problem. In doing so, we create an open-source software project that can be modified by other template designers to provide end-users with modifiable template designs.

1.3 Academic Literature Overview

What are print-and-fold robots? The term, printable robots, refers to a robot developed via a three step fabrication process. First, the user defines the pattern and mechanical design. Second, the user prints the design and gathers electronic components. Lastly, the user assembles the robot via folding and gluing. At the center of this process is folding which is efficient compared to other available techniques but not commonly used in engineering and science. Only requiring folding means the resulting object is compact and lightweight. Borrowing from the Japanese art of origami, this print-and-fold process involves printing 2D patterns that can be folded into functional 3D robots. For a given mechanism, theoretical analysis is needed in order to first show that it

can in fact be fabricated in this manner [1].

Previous work has involved folding micro/nanostructures providing a basis for the use of folding in the field of robotics. But there is not much work demonstrating a complete electro-mechanical system using this approach. Foldable robots present two critical design challenges: how to identify foldable 3-D mechanism to achieve desired task and how to specify a corresponding 2D fold pattern [1].

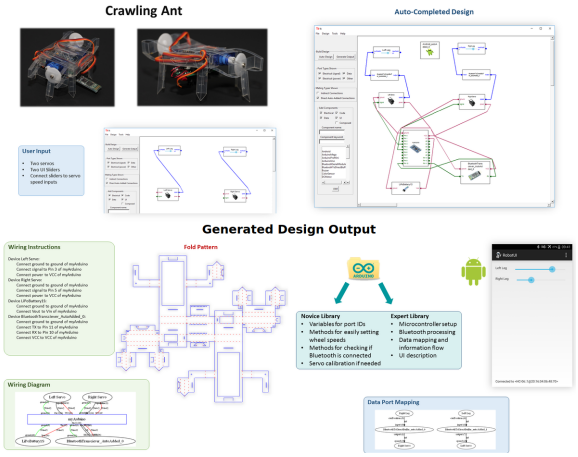


Figure 5: Interface for robot compiler. Source.

Why is print-and-fold useful? Robots have a variety of uses in today's society with applications in academic research, educational outreach, general household activities, and more. Yet, if an average person is interested in making a robot they must overcome significant barriers to design and manufacture one. Users often need highly specialized engineering skills in order to build functional robots. Additionally, tools and software can be quite expensive and time consuming to use. The few options available are not intuitively customizable for the generation of new devices. High prototyping costs combined with fabrication time illustrate a need for alternative fabrication techniques and strategies.

Studies on alternative approaches have offered customizable fabrication of robots using origami-inspired techniques. Ref [1] proposes printable robots to create electromechanical components that are folded into functional 3-D machines employing origami-inspired techniques. For an interface example see 5. The study presents the design, fabrication and test of prototype origami robots to address mobility and manipulation issues often found when using origami-inspired approaches. Starting with a parameretized template, they instantiate devices with performance characteristics optimized for a particular application. In order to demonstrate the print-and-fold process, they created prototypes (legged robot and a gripper). They were able to mathematically compose the 2D patterns for a legged robot and a robotic gripper into an ant-inspired design. Ultimately, this work demonstrated that printable prototypes can be designed and fabricated quickly, at low cost, with a functional end result.

Ref [2] presents a toolbox-like system to simplify and streamline the design and manufacture of printable robot bodies. The authors offer designs for origami-inspired structures that are generated by a collection of Python scripts, which allow a user to define complex geometries by hierarchical composing of simpler building blocks starting from a library of basic primitives. This study involved developing a script-based environment to allow users to easily design and create mechanical bodies for folded plastic robots. This work makes the approach accessible to a casual hobbyist.

Ref [3] addresses the issue of high prototyping costs and fabrication time. The authors present a new scripted design platform and printed folding fabrication process for

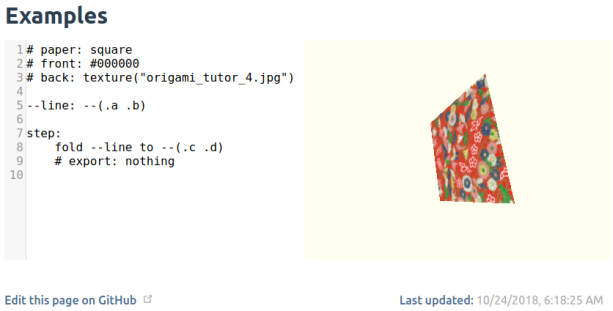


Figure 6: A screenshot of the origamiDSL website.

producing mechanical structures. Specifically, this study produces a lightweight, low cost and rapidly designed and manufactured mechanical air vehicle (MAV). This process is better than other robot design methodologies because it allows for modular design, requires low cost software, fabricates in shorter time, produces lighter structures, and enables rapid iteration (scripted design). This study offers a framework to design and fabricate simple robots using origami-inspired methods for sheets of plastic film.

A lot of research has been dedicated to developing geometry based domain specific languages, and there is clearly a lot of research on DSLs for robotics. However, we don't have many specification instruments, even in theory, that incorporate both papercraft geometry and robotics.

Thus, [4] demonstrated great progress in developing a geometry-based domain specific language for specifying origami folds and creases. Their language, OrigamiDSL, makes use of Huzita–Hatori seven origami axioms [5] that completely capture all the mathematical rules governing origami paper folding. A brief example of the DSL can be found in 6.

Origami robots[6, 7] is a line of research focusing on the creation of "live" origami. Origami robots autonomously change shape based on predetermined folds and creases. Assembling such robots requires a lot of expertise, and parameratizing them is a similar challenge to ours. Thus, if we develop a language for designing and parameratizing PaperSignals, it might be useful for systems like origami robotics.

1.4 Software Prior Art

Much online prior art exists, as papercraft generation is not a new topic. The prior art can be delineated into a few broad categories. The commercial prior art focuses on "die cut patterns." These are, for instance, the designs used to cut out cardboard in such a way that can be shipped flat and then assembled into a box (or other shape).

2 PROTOTYPE: PAPERSIGNALS PARAMERETIZED

2.1 Design Principles

The ultimate goal is to enable personal robotics users to complete the fabrication process with better usability and customizability. Therefore, the presented interface must be usable by those with and without a background in engineering design. With this in mind, the Parametrize PaperSignals interface was developed with a number of guiding principles.

First and foremost, the interface environment must be intuitive so that users are able to easily understand navigation and implement desired changes.

Ideally, what is displayed on the screen should be directly equivalent to the specifications made by the user. The designs generated by the interface must adequately display updates to the user as specifications change.

Additionally, the end-user should not be required to modify the source code in order to customize the template. Printable-robot templates should be customizable at least in

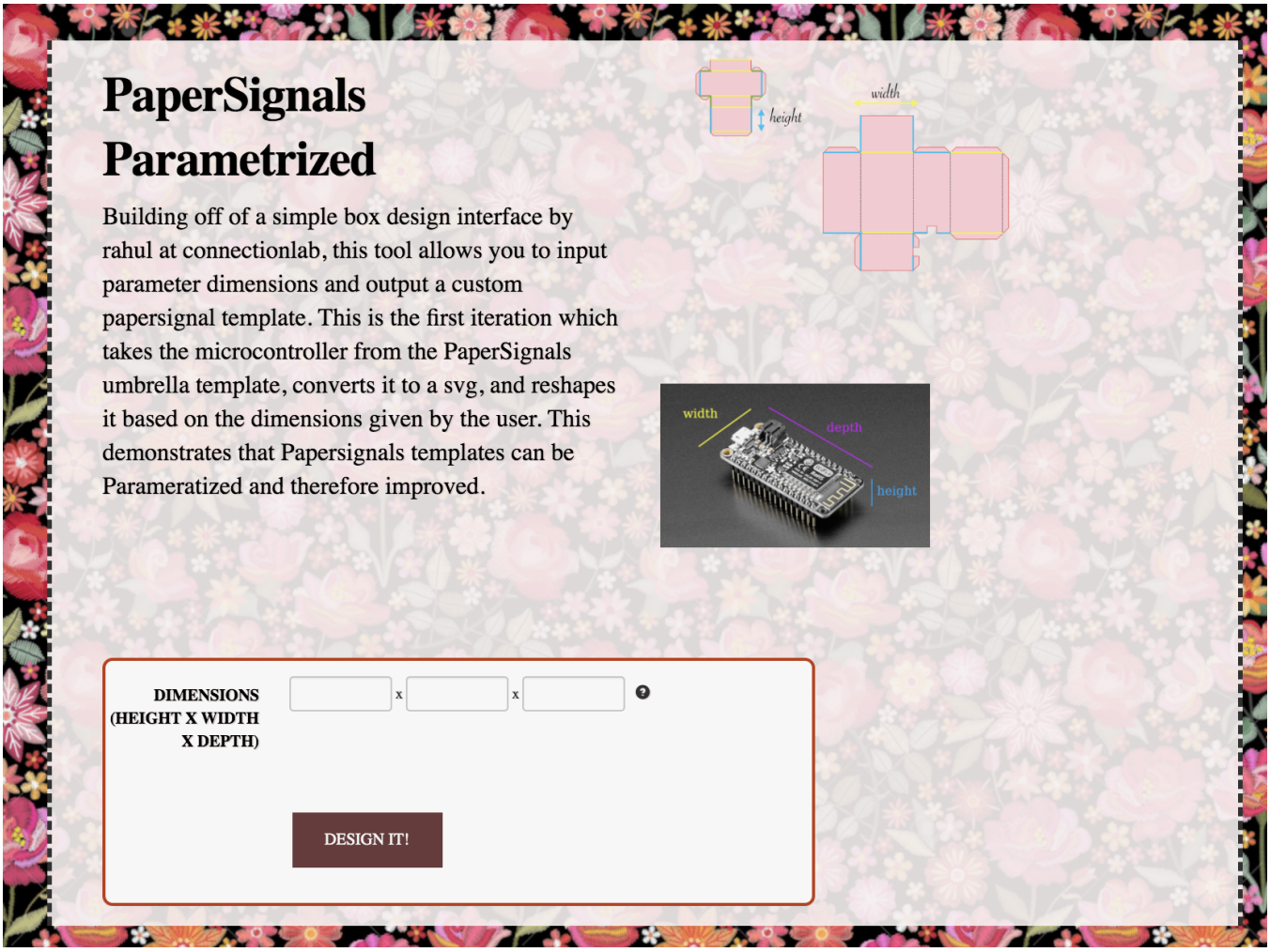


Figure 7: Custom PaperSignals Interface. http://customPaperSignals.herokuapp.com).

terms of dimensions. And for complicated designs, changing the dimensions of one part might warrant the update of another dimension. The interface must take these changes into account. Generated designs should be easy to modify and adapt to the user's needs.

The Parameterized PaperSignals interface is based on a collection of Python scripts that automate the changing of dimensions for two of the PaperSignals templates. The simplicity of the interface carries over many of the same benefits of similar open source software programs for paper craft objects to the robotic papercraft objects considered here. Non-expert users only need to input the desired dimension changes, while expert users can consider the design abstractions and work toward the development of fully parameretized printable-robot templates.

2.2 Implementation

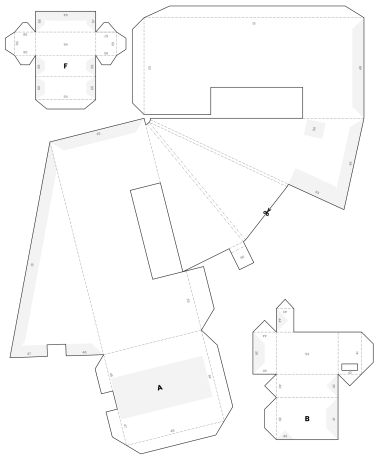
We developed a prototype of a website for generating customizable templates in the SVG format. The web application takes dimension parameters from the user and outputs

a set of templates for the microcontroller, the main box, and the other parts. All the parts have the right dimensions that perfectly fit each other, so that users do not have to adjust the dimensions manually.

The application consists of a HTTP server and a simple HTML user interface to interact with. In order to run the application, the code for the server is contained in a file called ‘server.py’.

We used Inkscape to trace the original Google templates, which are currently only available in a PDF format, and generate vector (SVG) files. This allowed us to nicely lasercut all the templates along with the dashed lines and provided more flexibility. Ever better, we could then easily modify the resulting SVG files. The process is as follows.

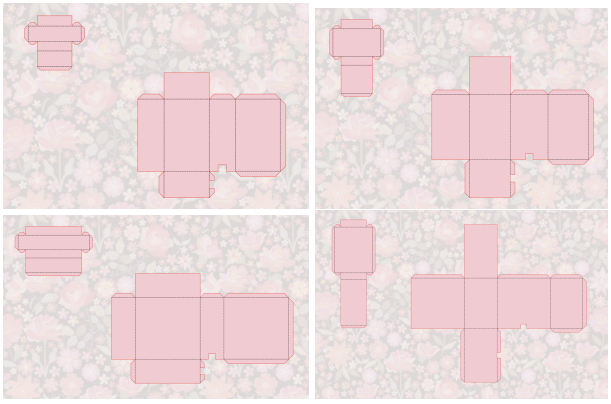
Each SVG template consists of several XML paths. An XML path is just a sequence of commands, starting with an “M” command specifying the beginning location, followed by commands for vertical, horizontal and diagonal movement. We then insert the user’s dimensions as horizontal and vertical commands, and output the resulting path in a new SVG file.



```
  <path
style="fill:#ffcfd9 ;stroke:#ff0000;stroke-width:0.5px;
 stroke-linecap:butt;
 stroke-linejoin:miter;
 stroke-opacity:1"
 d="M 35.075403,35.500069
 h 51.419828
 h ${w}
 v 15.827727
 ...
 V ${y}
 L ${p},${l}
 h -5.667727
 l -5.079998,-5.079997
 v 39.08037 v ${f}
 l -5.079998,5.08
 h -41.25983
 h ${ww}
 l -5.08,-5.08
 V ${vlu}
 h 5.667728
 l 5.079999,5.08 z"
 id="path3702"
 />
```

An SVG path derived from Google templates, with inserted user input dimensions as commands.

It is possible to modify different properties of the path and fill the shape with a user-defined pattern. One possible future direction for the project is to modify the interface to allow for template designers (separate from end-users, who are adjusting parameters on an existing template) to design parametric patterns.



Parameterized microcontroller & main box templates

3 DOMAIN SPECIFIC LANGUAGE FOR PAPERSIGNALS

The main challenge in parametrizing and designing PaperSignals is specifying both the design of the moving parts and the movement itself, while making sure that these specifications describe a feasible PaperSignal. The most trivial example is when we change the microcontroller box, we need to change the larger box, and if the moving parts depend on the size of the larger box, those need to be customized as well. It is a significant challenge to design a DSL that makes it possible for users to specify the entire PaperSignal. However, we can define a simpler DSL that has, for example, primitives box and `signalShape`, where box holds information about the dimensions and design of the main box, and `signalShape` holds information about the moving component of the PaperSignal.

Thus, if we are given coordinates for a pivot point, three dimensions, and some other parameters, we can "drill" a hole in our box, lasercut it and insert a wheel. For example, a potential PaperSignal designer would write

```
p = hMovement(pivot = (5,5),
   initPosition = -1,
   finalPosition = 3,
   box = b)
```

to declare that box B has a pivot at position 5,5, and it must move back and forth 5 units horizontally. More precisely, our proposed DSL looks as follows.

We have the essential syntax for performing basic operations that might be useful for designing the PaperSignal. These include arithmetic, loops, assignments, function definitions etc. Then, we have domain specific primitives such as box and `signalShape` that, depending on their arguments, keeps a string of the form `<path ... d =" ...z"id="path0000"/>`, and whenever there is a new component to the object, we generate a new string describing it, and concatenate it with the path.

Finally, the DSL will generate symbolic constraints responsible for "feasibility" of the PaperSignal and use a solver to maintain that various dimensions and specifications don't contradict each other. The building blocks for these constraints are symbolic values X_{int} and X_{bool}.

3.1 Syntax

Variable x

Constant c

$$c ::= X_{\text{int}} \mid X_{\text{bool}} \mid n \mid b \mid s \mid \mathsf{NA}$$

Parameter p

$$p ::= x \mid x = e \mid \mathsf{varparam}$$

Argument a

$$a ::= e \mid x = e \mid \mathsf{vararg}$$

Expression e

$$e ::= x \mid c \mid x \leftarrow e_1 \mid \{e_1; e_2\}$$

$$\text{if } e_1 \text{ then } e_2 \text{ else } e_3 \mid \text{while } e_1 \text{ do } e_2$$

$$\lambda \overline{p}.e \mid e\,(\overline{a})$$

$$\ldots$$

$$\mathsf{box}(\mathsf{signalShape} = t, \mathsf{dims} = (x_1, x_2, x_3),$$

$$\mathsf{pivot} = (x_1, x_2, x_3))$$

$$\mathsf{microcontroller}(a_1, a_2, a_3)$$

$$\mathsf{vmove}(a_1, a_2, a_3)$$

$$\mathsf{hmove}(a_1, a_2, a_3)$$

signalShape

$$
\begin{aligned}
t ::= \ & \mathsf{Umbrella}(e) \\
\mid \ & \mathsf{vMovement}(\mathsf{pivot} = e_1, \\
& \qquad \mathsf{initPosition} = e_2, \\
& \qquad \mathsf{finalPosition} = e_3) \\
\mid \ & \mathsf{hMovement}(\mathsf{pivot} = e_1, \\
& \qquad \mathsf{initPosition} = e_2, \\
& \qquad \mathsf{finalPosition} = e_3) \\
\mid \ & \mathsf{Clock}(\mathsf{pivot} = e_1, \\
& \qquad \mathsf{initPosition} = e_2, \\
& \qquad \mathsf{finalPosition} = e_3)
\end{aligned}
$$

Path

$$p ::= [] \mid \mathsf{Point}(e_1, e_2) :: p$$

3.2 Data Structures

Heap

$$H : m \to (v, A)$$

Value

$$v ::= (X, Y) \mid \overline{n} \mid \overline{b} \mid \overline{m} \mid (\lambda \overline{p}.e, m) \mid \Gamma \mid \mathsf{NA}$$

Attribute

$$A : s \to v$$

Environment

$$\Gamma : x \to m$$

Stack

$$S ::= \emptyset \mid (e, m_\Gamma) : S$$

Constraint

$$C ::= e_1 = e_2 \mid e_1 < e_2$$

3.3 Reduction Semantics

$$\textsc{IdentLookup}\ \frac{H(m_\Gamma) = (\Gamma, A_\Gamma) \qquad \Gamma(x) = m_v}{\langle (x, m_\Gamma); H\rangle \hookrightarrow \langle m_v; H\rangle}$$

$$\textsc{Constant}\ \frac{m_v \text{ fresh} \qquad H' = H[m_v \mapsto (c, \emptyset)]}{\langle (c, m_\Gamma); H\rangle \hookrightarrow \langle m_v; H'\rangle}$$

$$\textsc{Assign}\ \frac{H(m_\Gamma) = (\Gamma, A_\Gamma) \qquad \Gamma' = \Gamma[x \mapsto m_v]}{\langle (x \leftarrow m_v, m_\Gamma); H\rangle \hookrightarrow \langle m_v; H'\rangle}$$

$$\textsc{LamAbs}\ \frac{m_f \text{ fresh} \qquad H' = H[m_f \mapsto ((\lambda \overline{p}.e, m_\Gamma), \emptyset)]}{\langle (\lambda \overline{p}.e, m_\Gamma); H\rangle \hookrightarrow \langle m_f; H'\rangle}$$

...

$$\textsc{Box}\ \frac{\begin{array}{c} H(m_1) = (v_1, A_1) \qquad H(m_2) = (v_2, A_2) \\ s = \text{“} < path \ldots > \text{”}[shape = v_1, dims = v_2] \qquad H' = H[m_v \mapsto s] \end{array}}{\langle (\mathsf{box}(m_1, m_2); H\rangle \hookrightarrow \langle m_v; H'\rangle}$$

$$\textsc{Umbr}\ \frac{\begin{array}{c} H(m_1) = (v_1, A_1) \quad H(m_2) = (v_2, A_2) \quad H(m_3) = (v_3, A_{v_3}) \\ s = \mathsf{umbrPath}(v_1, v_2, v_3) \qquad H' = H[m_v \mapsto s] \end{array}}{\langle (\mathsf{umbrella}(m_1, m_2, m_3); H\rangle \hookrightarrow \langle m_v; H'\rangle}$$

$$\textsc{Path}\ \frac{\begin{array}{c} H(m_1) = (v_1, A_1) \quad H(m_2) = (v_2, A_2) \quad H' = H[m_v \mapsto s] \\ s = s' + \text{“L ”} + str(v_1) + str(v_1) \quad \langle p; H\rangle \hookrightarrow \langle m_{s'}; H'\rangle \end{array}}{\langle (\mathsf{Point}(m_1, m_2) :: p; H\rangle \hookrightarrow \langle m_v; H'\rangle}$$

$$\textsc{hMove}\ \frac{\begin{array}{c} H(m_1) = (v_1, A_1) \quad H(m_2) = (v_2, A_2) \quad H(m_3) = (v_3, A_{v_3}) \\ s = \mathsf{hMovePath}(v_1, v_2, v_3) \qquad H' = H[m_v \mapsto s] \end{array}}{\langle (\mathsf{hMovement}(m_1, m_2, m_3); H\rangle \hookrightarrow \langle m_v; H'\rangle}$$

$$\textsc{vMove}\ \frac{\begin{array}{c} H(m_1) = (v_1, A_1) \quad H(m_2) = (v_2, A_2) \quad H(m_3) = (v_3, A_{v_3}) \\ s = \mathsf{vMovePath}(v_1, v_2, v_3) \qquad H' = H[m_v \mapsto s] \end{array}}{\langle (\mathsf{vMovement}(m_1, m_2, m_3); H\rangle \hookrightarrow \langle m_v; H'\rangle}$$

$$\textsc{Clock}\ \frac{\begin{array}{c} H(m_1) = (v_1, A_1) \quad H(m_2) = (v_2, A_2) \quad H(m_3) = (v_3, A_{v_3}) \\ s = \mathsf{clockPath}(v_1, v_2, v_3) \qquad H' = H[m_v \mapsto s] \end{array}}{\langle (\mathsf{Clock}(m_1, m_2, m_3); H\rangle \hookrightarrow \langle m_v; H'\rangle}$$

3.4 Future Work

(1) Functional specification: The presented parameterization allows for adjusting dimensions based on geometric specification of previously defined elements. Another component of fabrication could involve consideration for the mechanical properties and incorporating functionality into the flexibility of the design. Long term, the printable robotics community could aim to generate designs directly from functional specifications.

(2) Domain Specific Language: The presented abstractions are a first attempt at defining primitives for parameterizing of PaperSignals templates. There already exist robotics experts who have achieved parameterizable and customizable robots with high precision and great functionality. It would be interesting to consider the design components that can be extracted from these models and develop a more sophisticated DSL based on this.

(3) Fabrication techniques: It is worth considering how this fabrication process scales with larger or smaller functional robots and not simply PaperSignals robots, which have only a single degree-of-freedom (rotation from the servo motor).

The fabrication process may have to be adapted in order to scale. For example, the features that can be cut out by hand are limited and therefore complex laser cutting tools would be needed for smaller objects/features. Future work would involve considering improving or automating certain parts of the fabrication process, while continuing to be accessible to the average user. This would present quite the challenge.

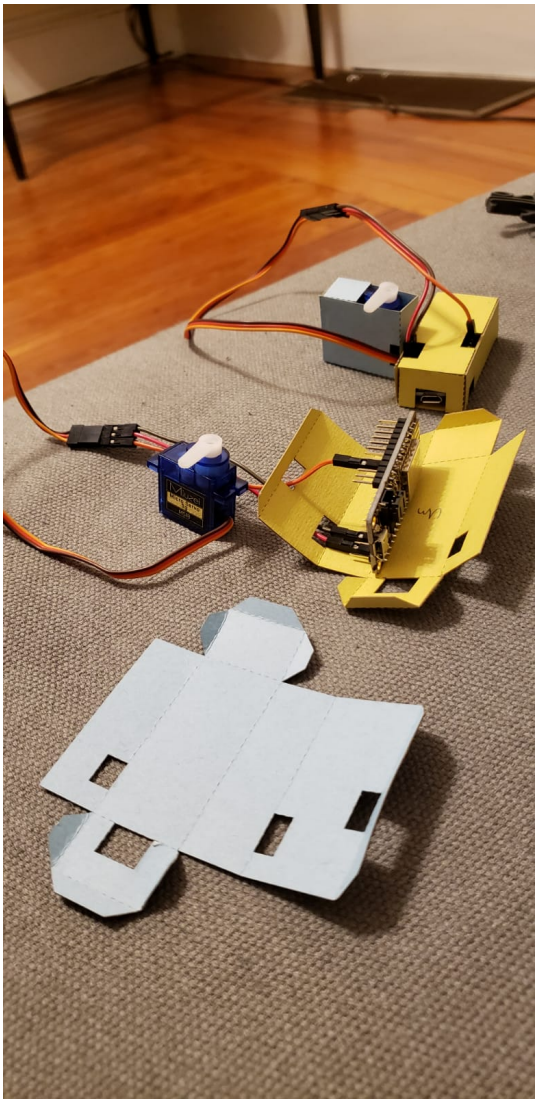


Figure 8: The servo is in blue, with wires connecting to the microcontroller (shown with USB port facing us). Ebay microcontrollers are servos are several times cheaper than Google's ($5 vs $25), however, Google's templates have dimensions that are incompatible with the cheaper version. Note also the functional specifications induced on the paper template: holes must exist for the wires to connect the microcontorller to the servo, as well as for the USB port to program the papercraft.

4 CONCLUSION

The primary contributions of this work is a method for developing an interface to change the dimensions of existing PaperSignals templates and an abstraction of a few

design primitives for further parameterizing the templates. The interface was implemented in Python and used to generate design files for the PaperSignals umbrella microcontroller and main box components. The design primitives are a first attempt at a set of abstractions that would could specify different PaperSignals templates. This work furthers the goals of printable-robot design in empowering and encouraging creative interactions with our physical world.

ACKNOWLEDGEMENTS

This project required much guidance and assistance from Prof. Elena Glassman and Prof. Nada Amin and we are extremely grateful for their contributions to the completion of this project. The authors would like to acknowledge the contributions of the students in CS 252r and CS 279r (Harvard PL/HCI Graduate Seminar).

REFERENCES

[1] Michael T. Tolley Robert J. Wood Onal, Cagdas D. and Daniela Rus. Origami-inspired printed robots. *IEEE/ASME Transactions on Mechatronics*, 20(5), 2015.

[2] Ankur M. Mehta and Daniela Rus. An end-to-end system for designing mechanical structures for print-and-fold robots. *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[3] Ankur M. Mehta, Daniela Rus, Kartik Mohta, Yash Mulgaonkar, Matthew Piccoli, and Vijay Kumar. A scripted printable quadrotor: Rapid design and fabrication of a folded mav. In *ISRR*, 2013.

[4] Gaetano Caruana and Gordon Pace. Embedded languages for origami-based geometry. 01 2007.

[5] Roger C Alperin and Robert J Lang. One-, two-, and multi-fold origami axioms. *Origami*, 4:371–393, 2009.

[6] Amir Firouzeh and Jamie Paik. Robogami: A fully integrated low-profile robotic origami. *Journal of Mechanisms and Robotics*, 7(2):021009, 2015.

[7] Daniela Rus and Michael T Tolley. Design, fabrication and control of origami robots. *Nature Reviews Materials*, 3(6):101, 2018.