

INTERACTIVE DESIGN INTERFACES TO SUPPORT IDEATION AND RAPID PROTOTYPING

by

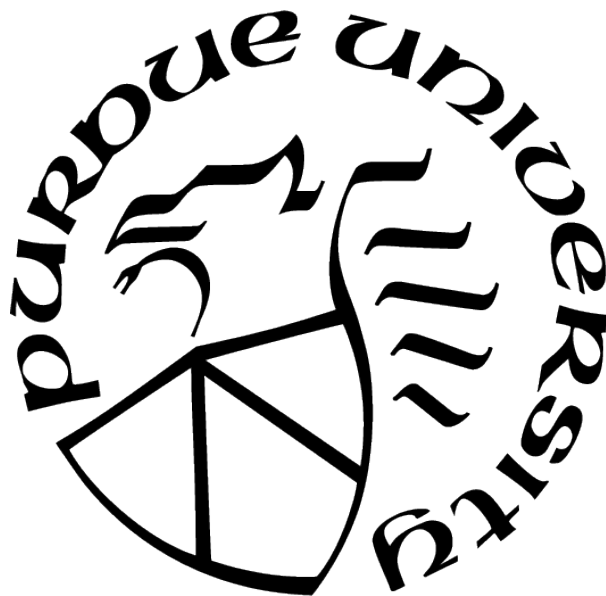
Devashri Vagholkar

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science



School of Mechanical Engineering

West Lafayette, Indiana

December 2021

**THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL**

Dr. Karthik Ramani, Chair

School of Mechanical Engineering

Dr. Bedrich Benes

School of Computer Graphics Technology

Dr. Tahira Reid Smith

School of Mechanical Engineering

Approved by:

Dr. Nicole Key

This work is dedicated to my family Sangita Vagholkar, Utpal Vagholkar, Kamakshi Vagholkar and my late grandfather Vaman Vagholkar for their unconditional love and support and for ingraining in me the values of hard work and honesty.

ACKNOWLEDGMENTS

In this acknowledgement, I would like to thank Dr. Karthik Ramani for his mentoring and advice throughout the entire premise of this thesis. Through his support and guidance, I have gained valuable experience both academically and professionally. I would also like to thank Professor Bedrich Benes and Professor Tahira Reid Smith for serving in my committee. I am thankful to Prof. Benes for introducing me to the world of graphics and geometry and also Dr. Will Yunbo and Dr. Min Liu for advising me in my work. I would also like to thank all the people from the Convergence Design Lab at Purdue that participated in user studies and shared their valuable feedback on my work. Special thanks for Terrell Glenn and Luis Paredes for being amazing project leads and mentors in my main projects. I would also like to acknowledge Subramanian Chidambaram and Ananya Ipsita for their valuable advise and feedback at different stages of my work. Lastly, I would like to thank my teammates Pashin Raja, Kiran Payne, Sai Reddy, Runlin Duan and Dr. Yangzi Dong for their collaboration and team work in the projects and sharing their knowledge and skills with me.

TABLE OF CONTENTS

LIST OF FIGURES	8
ABSTRACT	10
1 INTRODUCTION	11
1.1 Product Development and Design Thinking	11
1.2 Tools for Product Development and the Users	13
1.3 Motivation and Research Goals	14
1.4 Scope	14
1.5 Thesis Organization	15
2 RELATED WORK	16
2.1 Sketch-based Ideation Tools	16
2.2 Design tools for IoT Devices	17
2.2.1 Programming	17
2.2.2 Electronics Toolkits	18
2.3 Design tools for Hand Wearables	19
3 DESIGN INTERFACE FOR SKETCH-BASED IDEATION	20
3.1 Overview	20
3.2 System Design	21
3.2.1 Motivation	21
3.2.2 Design Principles	21
3.3 Implementation	23
3.3.1 The Problem	23
3.3.2 The Interface	23
3.3.3 Sketcher	24
3.3.4 Simulator	26
3.4 Testing & Results	26
3.4.1 Testing	26

3.4.2	Results	27
3.5	Limitations & Future Work	29
3.6	Conclusion	29
4	DESIGN INTERFACE FOR IOT DEVICES	31
4.1	Overview	31
4.2	System Design	32
4.2.1	Motivation	32
4.2.2	Design Principles	33
4.3	Implementation	33
4.3.1	Live Programming Simulator	33
4.3.2	Hardware Prototyping ToolKit	36
4.3.3	Controller Mobile App	37
4.4	Testing Results	38
4.4.1	Expert Study	38
4.4.2	Youth Study	40
4.4.3	Results	40
4.5	Limitations & Future Work	41
4.6	Conclusion	41
5	DESIGN INTERFACE FOR HAND WEARABLES	43
5.1	Overview	43
5.2	System Design	43
5.2.1	Motivation	43
5.2.2	Design Principles	44
5.3	Implementation	45
5.3.1	Hand Customization	45
5.3.2	Electronics Functionality Design	46
5.3.3	Design for Fabrication support	47
5.4	Testing & Results	50
5.4.1	Testing	50

5.4.2	Results	50
5.5	Limitations & Future work	51
5.6	Conclusion	51
6	CONCLUSION	53
	REFERENCES	54

LIST OF FIGURES

1.1	Double Diamond representation of Product Development	12
1.2	A Designer’s Journey through the Solution Space	12
3.1	Sketch2Sim: A Sketch-based Early Stage Design Exploration Tool	20
3.2	The Ideal Ideation Tool	22
3.3	Sketch2Sim Interface	23
3.4	Sketching System Architecture	24
3.5	Sketching Rules	25
3.6	Refining Freehand Sketch Strokes to fit Shapes	25
3.7	Generating Graph Model from Sketch	26
3.8	JSON file format for saving designs and transferring to other tools	27
3.9	Different designs that can be supported by the tool	28
3.10	Idea Timeline: Designs based on same ideas were created at different points of time	29
3.11	Ideation Bubble: Different ideas explored by a user to design a 2D vehicle that can overcome an obstacle	30
4.1	IoTMaker: A System for Ideation and Rapid Prototyping of IoT devices	31
4.2	Learning curve comparison between IoTMaker and other systems	32
4.3	IoTMaker Workflow	34
4.4	IoTMaker’s Live Programming Simulator Interface	35
4.5	Electronics Repository	36
4.6	IoTMaker Coding Challenges	39
5.1	FabHandWear- An end-to-end system for Customized Functional Hand Wearables	43
5.2	FabHandWear- Interface	45
5.3	Hand Parameterization: (a) User input 4 hand dimensions; (b) 3 angle values for each finger based on initial reference measurement; (c) 5 length values for each finger based on initial reference measurement, each length expressed as multiple of palm height; (d) generation of 30 skeletal points based on b and c ; (e) generation of 31 outer points based on skeletal points and multiples of finger width; (f) extruding the 2D outer points and smoothening the edges	46
5.4	FabHandWear- Interface, Hand Customization	46

5.5	FabHandWear- Interface, Electronics Placement	47
5.6	FabHandWear- Circuit routing	48
5.7	FabHandWear- Fabrication template and screen printed wearable	48
5.8	FabHandWear- Steps in Fabrications	49
5.9	FabHandWear- Example Applications	50
5.10	FabHandWear- Custom fit wearable design on different hands	51

ABSTRACT

Generating ideas and creating prototypes of physical products is a highly non-linear and iterative process. Current tools divide this process into multiple discrete steps with different tools to support each of these steps such as CAD modelling, simulation and fabrication. We believe, design interfaces that combine different steps of the process and create different layers of abstraction depending on the type of the user and where they are in the process can support users in generating more creative ideas and creating better functioning prototypes more efficiently. In order to validate this, we developed three interfaces- a sketch-based ideation tool, a live programming interface to create IoT devices and a design tool to support design and fabrication of hand wearables. The foundation of these design interfaces is the layer of abstraction that allows users to focus on idea generation and converting it into a tangible prototype with little or no technical knowledge, and a continuous visual feedback that guides the user to make necessary changes to improve their design. The three tools were evaluated through user testing for supporting creation of different ideas and converting them into functional prototypes.

1. INTRODUCTION

1.1 Product Development and Design Thinking

Product development is defined as the process of creating new products or modifying existing products so as to offer some kind of benefit to the user. Product development can involve changing an existing product or creating a completely new one to satisfy the newly defined wants of the user or the market [1]. There are many definitions of product development offered in literature. It is important to understand each of them has its validity based on the type of the product and the type of the problem the product is trying to solve.

In general, we can classify it into the following steps:

1. Idea generation
3. Concept/prototype development and testing
4. Business model development

In recent years the term "Design Thinking" has gained popularity and is often linked towards the human-centered aspect of product development. Design thinking is defined as a human-centered approach to create innovative solutions by integrating together the user needs with technology and business. Design thinking looks at product development more like a problem solving process and highlights the 5 main steps as follows:

1. Empathize
2. Define
3. Ideate
4. Prototype
5. Test

Figure 1.1 shows a graphical view of the listed 5 steps in the famous double diamond representation by Plattner et al. (2009) [2]. The first diamond is the problem space while the second diamond is the solutions space. Figure 1.2 shows the journey of a designer in the solution space. It first starts with an idea, followed by some quick low-fidelity ideation on a pen and paper. The next step is to translate the idea into a digital version such as a CAD tool. The CAD tool then creates supporting fabrication files that can be used to create physical prototypes. These physical prototypes are then tested in real world scenarios

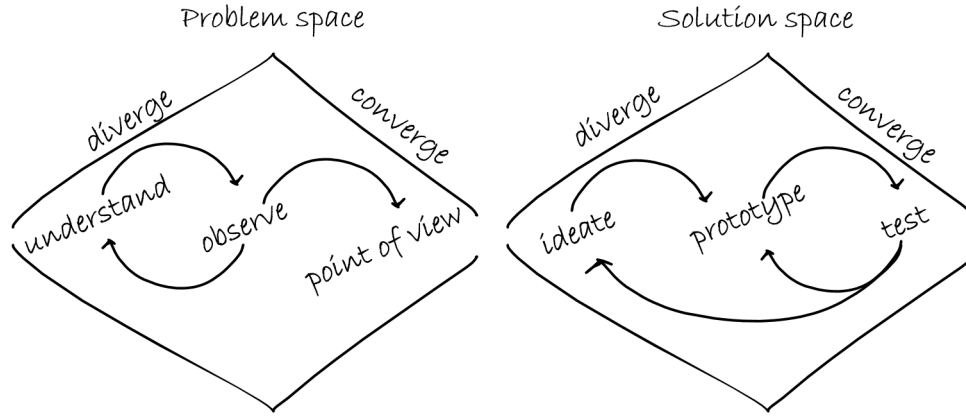


Figure 1.1. Double Diamond representation of Product Development

or user testing or simulation to evaluate their performance. Based on this feedback the designer decides whether to reiterate the process again and repeat the cycle until desired performance and functionality is achieved.

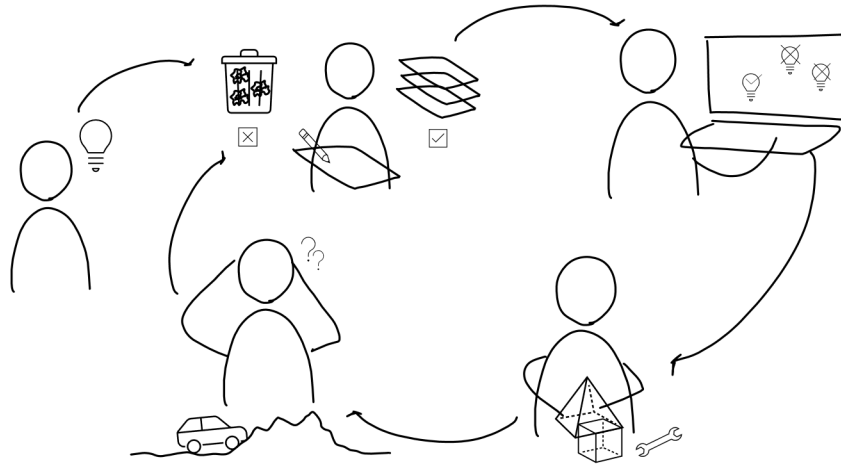


Figure 1.2. A Designer's Journey through the Solution Space

Humans naturally develop patterns of thinking modeled on repetitive activities and commonly accessed knowledge [3]. These patterns of thinking also known as schemas are organized sets of information and relationships between things, actions and thoughts that are stimulated and initiated in the human mind when they encounter some environmental stimuli. As these schemas are stimulated automatically it obstructs the users from thinking outside the box and prevents them from viewing the problem in a way that will enable

them to come up with new solutions. Design thinking frameworks encourage breaking these schemas and promotes creative thinking to come up with unconventional or out-of-the-box solutions.

1.2 Tools for Product Development and the Users

The emergence of the field of product design and industrial design is linked to the industrial revolution in mid 18th century with widespread progress in mechanization and industrialization. Almost all of the design work was done manually using sophisticated tools on paper. Over the years Computer Aided Design (CAD) was introduced and design started taking a digital form on computer screens using 2D parametric sketching and 3D solid modelling. The most popular tools in CAD are Solidworks [4], Creo [5] and AutoCAD [6]. With further advancement in computation and processing powers of computers, various analysis tools such as ANSYS [7] were released which could validate the performance of a design in given physics conditions. While these tools are extremely precise and fast, and have saved a lot time for humans, one needs to have substantial technical knowledge of geometry, design and engineering to be able to feed the correct input to these tools. This requires spending time in gaining the domain knowledge and learning the tools. Since these tools are specifically designed for engineers, the knowledge is already there and with training and practice they become proficient in using these tools. The challenge lies in making these tools accessible to individuals who do not have domain knowledge but have ideas and want to create them. This thesis focuses on this category of users who have ideas and want to quickly validate them or even create working physical prototypes. These users could be creatives, artists, hobbyists, DIYers or K-12 students. In order to make these tools accessible to this set of users it is important to understand what the user already knows and what the user needs to know in order to successfully create ideas and prototypes. Based on this information, one needs to create design interfaces by carefully identifying the "black boxes" or layers of abstraction. The purpose of these design interfaces is to provide a toolkit that allows them to be creative and translate their ideas into prototypes without having to worry about the technical details.

1.3 Motivation and Research Goals

The Maker movement, DIY culture and garage/backyard labs have inspired a lot of people with different backgrounds into building their own products from their ideas. This has created a need for design and prototyping tools for such users who can quickly translate their ideas to working models without spending too much time in learning those tools. Another motivation is one of the features of design thinking [8] that states encourages designers to make ideas tangible in early stages of product development. Prototypes must be created as quickly as possible. The maxim when creating or selecting a prototype is: as simple as possible, as meaningful as necessary. With this as the motivation, the research goals of this thesis is to create and validate interactive design interfaces that:

1. have a lower barrier of entry
2. promote freedom of thoughts and creativity
3. support rapid creation of functional prototypes

1.4 Scope

Referring to Double diamond representation of product development process in Figure 1.1 again, the Problem Space deals with understanding the problem and asking counter questions to reinforce the understanding around the problem. The stages include divergent thinking for enquiry based research, user interviews and observations to converge to a 'point of view' of the problem. Once the problem is clearly identified, the next step is to look for the solutions. This happens in the solution space where again divergent thinking is promoted to explore different ideas, followed by convergent thinking in prototyping and testing. This is an iterative process and has to be repeated multiple times before converging to one solution. The scope of this thesis lies in the solution space; developing design interfaces to promote idea generation and evaluation, and creating prototypes for testing. These proposed design interfaces combine different steps of idea generation along with active feedback on the idea's performance. The three design interfaces that will be discussed in the following chapters are:

1. Sketch2Sim: a sketch-based ideation tool
2. IoTMaker: a live programming interface for creating IoT devices
3. FabHandWear: a design tool to support design and fabrication of hand wearables

1.5 Thesis Organization

The thesis begins with literature review and related work for the design interfaces followed by detailed chapters on each of the three design interfaces- Sketch2Sim, IoTMaker and FabHandWear. In each chapter we discuss the motivation, development and testing for each of the design interfaces. Towards the end we conclude with our findings from development of the three interfaces and how they can help establish design rationale for similar design interfaces.

2. RELATED WORK

The two steps in the solution space of product development are Ideation and Rapid Prototyping. While ideation is more of an abstract process and revolves around the designer’s intuition, creativity and prior experience, Rapid Prototyping gives a structure to these abstract ideas and translates them into tangible models that can validate the ideas. Both the steps are necessary and critical to successfully come up with a solution. In the following sections we shall discuss some prior research on both these steps.

2.1 Sketch-based Ideation Tools

Ideation is the first step in problem solving. Engineers and designers often begin ideation with some rough sketches of their ideas on a pen and paper or white board followed by iterating on these sketches until they feel confident with the idea and then translate the sketch into some digital format or CAD model. Sketching promotes creativity by supporting exploration [9]. Sketching allows rapid idea generation without having to learn a new tool. In the early stage of product design it is very important to explore as many different ideas as possible before bolting down any one of them. Another advantage of ‘sketch’ is it creates a natural abstraction of the intricacies of the designs. Sketches are mostly 2D schematics that define shape, form factor, location, orientation, forces and motion directions. At sketch level one doesn’t worry about the performance, friction, wear and tear and tolerances. This kind of abstraction is necessary in the early stages so that it does not constrain the thinking of the designer. This very practice of sketching for ideation has been explored by the research community to augment these static sketches on a paper. Cobbie [10] is a sketching companion robot that collaborates with the user by generating new sketches based on user’s sketches. Another way to augment static sketches is to create them digitally and have smart tools that can understand the sketch intent or refine the sketch for higher fidelities [11]. Image recognition from freehand sketches has also been explored widely in supporting ideation and creativity through digital sketches. Juxtapoze [12], a clipart composition tool that uses freehand sketches and suggests different images of objects with similar shapes, Alchemy [13] on the other hand is a character design and concept art tool based on digital

freehand sketching. Both the tools promote creative expression and serendipitous discoveries. Freehand sketch to art is one way of creative ideation while there is also work that translates sketch to 3D models. Co3deator [14] is a collaborative tool that allows users to create solid 3D assemblies from a combination of their sketches to create different designs. SketchOpt [15] and DreamSketch [16] create generative 3D models from sketch based inputs.

2.2 Design tools for IoT Devices

With the tremendous technological innovation over the past decade today we are surrounded by internet connected devices all the time. These internet connected devices or Internet of Things (IoT) devices are supported by two main pillars - 1. Electronic hardware that allows the communication between input and output components and 3. the program that defines the behavior of these IoT devices. In order to create functional prototypes of these smart devices one needs to be able to understand and connect these three pillars which are discussed below in greater detail.

2.2.1 Programming

Creating a functional piece of program requires one to understand various complex aspects of computing such as compiling, interpretation and have sufficient knowledge of different programming languages. Programming is only a part of creating IoT devices and for one to be able to quickly create prototypes an in-depth knowledge may not be necessary. There are many higher level programming languages that are fairly easy to learn. These programming tools allow creators to focus on the functionality without having to worry about the internal workings of the code. Block-based programming languages are one such example [17]. Scratch [18] is a popular block based programming platform for beginner and expert level coders to create games and on-screen interactive prototypes by simply dragging and dropping code blocks like puzzle pieces. This approach converts an otherwise intimidating subject into a playful experience and promotes faster learning and creativity. Block programming has also been extended to robotics and STEM learning kits. One level below block programming are Integrated Development Environments (IDE). IDEs support higher

level programming languages for different applications. Many robotics kits come with their own IDE to support programming of input and output devices on their platform [19], [20]. These IDEs support libraries of different components such as motors and sensors and contain functions to code quickly without having to understand the logic behind them and create fairly complex to even commercial grade prototypes.

2.2.2 Electronics Toolkits

The second aspect of IoT devices involves putting together various input and output devices such as actuators, LEDs and sensors. With the Maker movement and DIY culture, hardware development is now democratized and the term 'tinkering' has gained popularity. Such 'tinkering' kits have reduced the barrier of entry for hardware development. Arduino [19] and Raspberry Pi [20] are two such hardware prototyping platforms that are accessible to beginner and expert users. At the heart of these kits is a pre-soldered PCB that allows plug and play of various input/output devices. The kits also support extensible modules in form of shields that allows more complex or specific functionality based on user's needs. For example, a wifi or bluetooth shield can enable communication between multiple devices located at different locations or access data from cloud. More such kits mainly addressed to K-12 learning include Sphero [21], micro-bit [22], Lego Mindstorms [23], VEX Robotics [24]. All of these kits are based on the principle of fastening the creation of functional prototypes. One very interesting example is microBit [22] that combines block programming with live simulation of electronics on the screen. Based on the block code of different logical and input/output device blocks the interface renders an animation of the electronics circuit. This allows users to view their code's behavior before they can actually put together the connections physically. In this way the user can debug the code earlier in the design process and save time, effort and frustration that would occur if the device does not behave as expected. Although microBit Live programming interface offers tremendous advantage in early debugging of code through the live simulator it still comes with a small set of peripheral devices that limit the users from exploring wider range of ideas. Additionally, the microBit board requires the users to have some basic understanding of circuit connections and basic

bread boarding skills thus increasing the barrier of entry. In order to address these limitations it is necessary to add more input/output devices that can support more functionalities, thus expanding the scope for creativity. Secondly there needs to be a layer of abstraction with respect to pin connections and power systems. With these two additional features we propose IoTMaker a live programming interface with a wide electronic repository that can be connected by simple plug and play interaction. This interface is discussed in Chapter 4.

2.3 Design tools for Hand Wearables

Hands are the most versatile human body parts. One can accomplish a variety of actions using hands such as waving, grabbing, touching, pointing. This makes hands the most suitable part for wearable design. While hands offer this flexibility in interaction they are different for different individuals. This makes hand wearable design challenging because of the anatomical variability in hand forms, multiple degrees of freedom, and absence of standard manufacturing techniques. These challenges influence the accuracy and comfort of hand wearables [25], [26], [27], [28], [29]. This creates a need for design tools that support creation of custom-fit hand wearables. One such design tool that supports creation of custom-fit wearables are ElectroDermis [30]. Electrodermis allows users to scan their body parts that is converted into a 3D mesh. The software tool then flattens the 3D mesh and allows users to place sensors and generate fabrication files. While Electrodermis supports creation of custom-fit wearable patches for different individuals it requires sophisticated 3D scanning tools and is more suitable for simpler geometries. Another such design tool LASEC [31] supports design for manufacturing of custom shapes of wearables. The design tool is based on a parametric stretchable pattern of wearable patch that can be laser cut. The system supports ease of fabrication but does not offer much support for custom-fit hand wearable design as it requires the user to manually measure the dimensions of the hand and recreate the desired shape in the interface. In order to address these gaps in hand wearable design tools, we propose FabHandWear [32] an end to end system for creation of customized and functional hand wearables. Chapter 5 in this thesis will cover the design interface workflow and hand parameterization feature of this tool.

3. DESIGN INTERFACE FOR SKETCH-BASED IDEATION

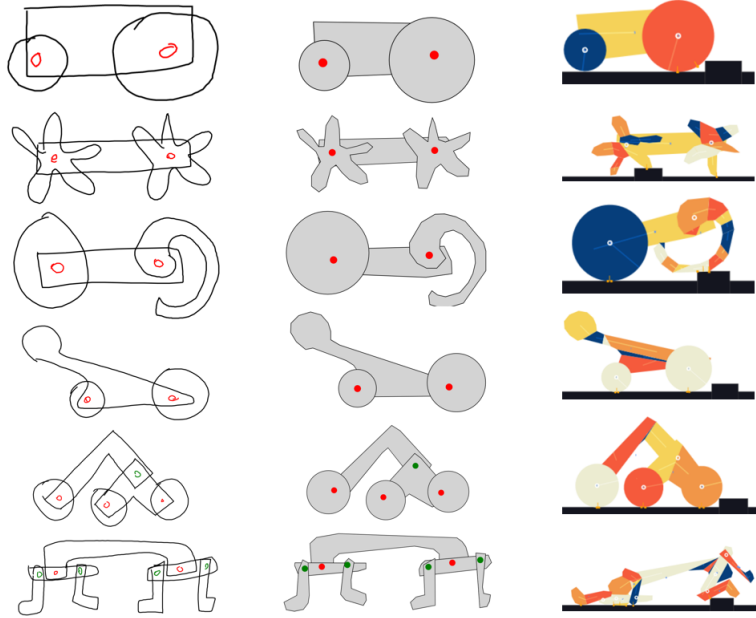


Figure 3.1. Sketch2Sim: A Sketch-based Early Stage Design Exploration Tool

3.1 Overview

Sketch2Sim is an early stage exploratory ideation tool that promotes creative thinking to support users to come up with different design ideas. It is meant for different types of users from beginners to experts. A simple use case of designing a terrestrial vehicle that can go from one end of the terrain to another has been considered for the study of this interface. The common approach to solve such a design problem would be to start with some sketches on a paper and then re-create them on a CAD package. In S2S we offer a digital version of freehand sketcher that can provide instantaneous feedback to the user and support exploratory ideation. S2S is a web based application that is based on creativity support design principles established on prior research.

3.2 System Design

3.2.1 Motivation

Engineering design process begins with an engineer creating sketches of his ideas. After a few sketches they may select a design that they think will 'do the job' based on their intuition or prior experience. This design is then translated into a CAD package [4], [5], [6]. The CAD files are then validated by running simulations or developing low fidelity prototypes. It is only after this point that the engineer is able to get some kind of feedback and find out whether the design successfully does the desired job. One has to invest a lot of time and resources before he/she can get any kind of feedback about their design idea. Another problem specially among non-expert users, would be when one is not very familiar to the CAD packages or simulation and prototyping tools. One has to spend time learning these tools and this might also prevent them from trying more complex ideas that would be difficult to model or prototype. These type of users are important in ideation process because they are not domain experts and can offer 'out of the box' ideas unlike experienced engineers who tend to follow 'ingrained patterns of thinking' [3]. With this motivation one can identify a need for a tool that can provide some kind of feedback on the design's behavior at the sketch level while still encouraging 'breadth of exploration'.

3.2.2 Design Principles

Sketch2Sim is based on principles of design thinking [8] and principles for creativity support [33]. As an early stage ideation tool the following features were considered when developing S2S:

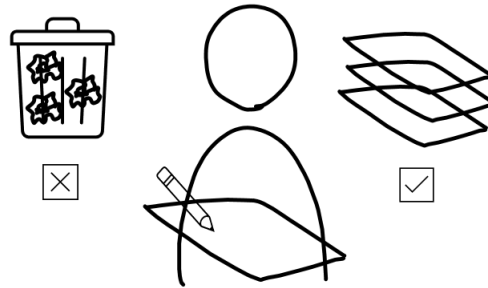
Design Thinking Principles:

1. Encourage wild ideas
2. Go for Quantity
3. Be visual and make it tangible
4. Fail early and often

5. Stay focused
6. Let's have fun

Principles of creativity support:

1. Support exploration
2. Low Threshold, High Ceiling, and Wide Walls
3. Support data Interchange with other tools
4. Make It As Simple As Possible - and Maybe Even Simple
5. Choose Black Boxes Carefully
6. Enable users to invent things



The Ideal Ideation Technique

Creativity X Rationality

Figure 3.2. The Ideal Ideation Tool

It is also important to understand that early stage ideation tools should also support rationality along with creativity. Creative ideas are not only the ones that are different from the obvious solutions but also have a strong rationale for why they are different. An ideal ideation tool would be one that combines creativity and Rationality [34]. One should be able to combine their conscious and unconscious minds.

3.3 Implementation

3.3.1 The Problem

A simple use case of a terrestrial vehicle that needs to go from one end to other end of the terrain is considered. The problem here is to design a 2D vehicle that can overcome three different types of terrains- 1. Flat terrain with an obstacle, 2. Uneven terrain, 3. Upwards slope terrain. This use case was considered since it does not require one to have any specific domain knowledge and is easily accessible to users with different backgrounds. The problem only requires knowledge of gravity, friction, collision, space, area and motion, which everyone understands even if they may not know the technical terms. The 2D vehicle can be made of 1. Parts that can have any shape and size and 2. Pin joints that can be powered (motors) or non-powered (free pin joints). The simplicity of problem is aimed to make users feel confident about solving it and encourage them to think freely.

3.3.2 The Interface

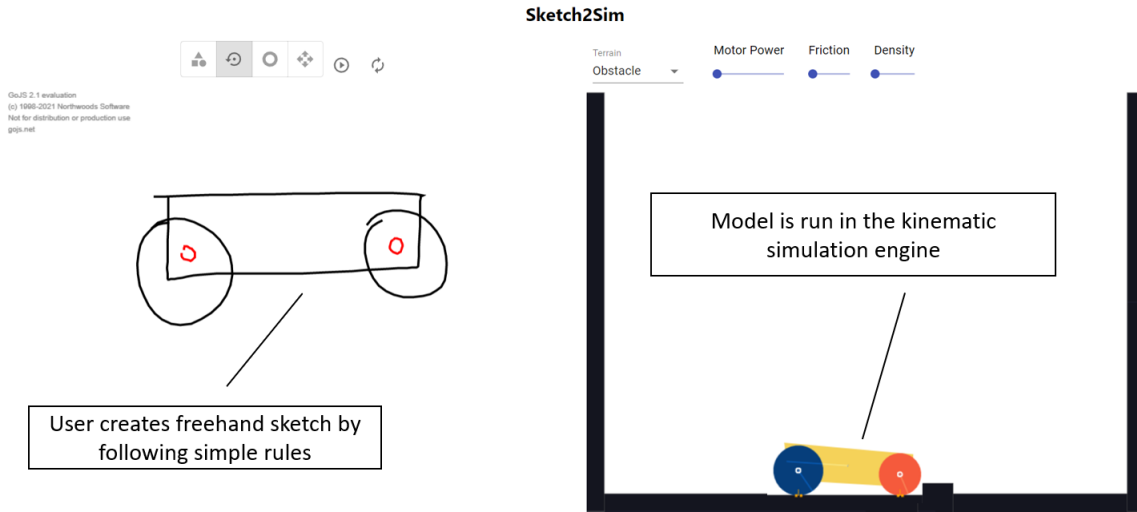


Figure 3.3. Sketch2Sim Interface

S2S is a web based application developed using React.js [35]. The GUI is divided into two parts- 1. Sketcher and 2. Simulator. The Sketcher is the creativity side while the simulator is the rationality side of this ideation tool. The sketcher takes in freehand sketch input from

the user which is then interpreted into a design graph that creates the kinematic model of the different sketch entities and their interrelations. This graph is then passed to the physics engine Matter.js [36] which renders the simulation of the 2D vehicle design in the predefined terrain.

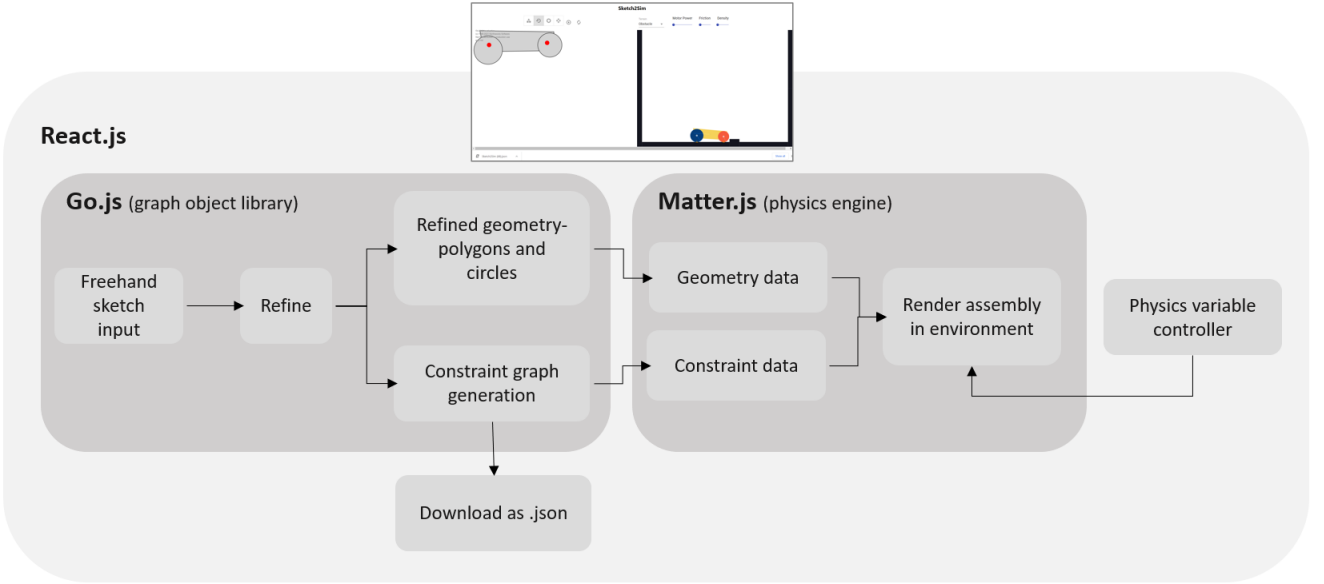


Figure 3.4. Sketching System Architecture

3.3.3 Sketcher

The Sketcher is developed on a graphing object javascript library Go.js [37]. The Sketcher is based on some simple rules that are motivated from natural sketching habits. The sketcher rules are listed below:

1. Create a part in a single stroke
2. Each part has to be a closed shape
3. Use the correct brushes to differentiate between parts (black), motors (red) and free pin joints (green)
4. Two parts that are connected by a joint should have an overlapping and corresponding joint drawn in the overlapping region

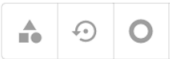

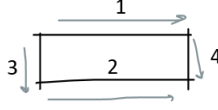


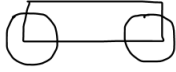

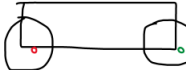
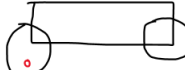
<div> <div>  </div> <div> Component: black stroke </div> <div> Motor: red stroke </div> <div> Pin joint: green stroke </div> </div>	
	
	
	
	
✓	✗

Figure 3.5. Sketching Rules

The two main features of the sketcher are refining the sketch entities and interpreting the design into the design graph. The first feature of refining simply fits circles and polygons (concave and convex) into the freehand sketch. It also recognizes the joints based on brush

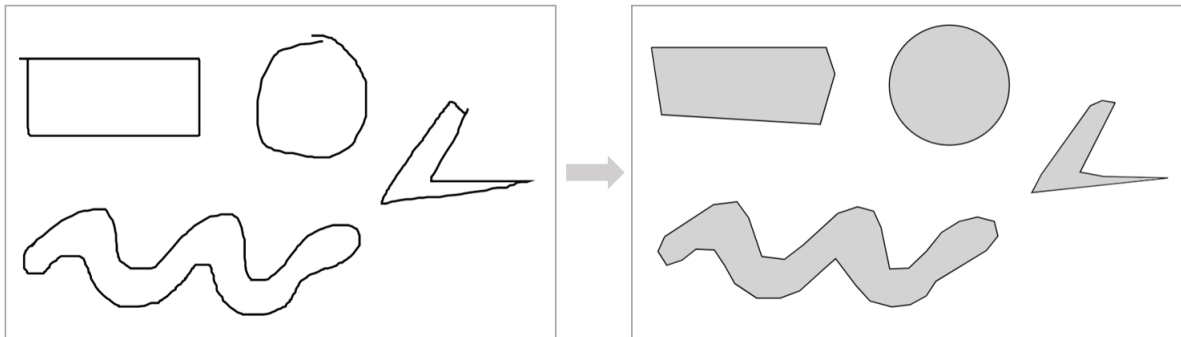


Figure 3.6. Refining Freehand Sketch Strokes to fit Shapes

colors. The second feature of interpreting the sketch into a design graph work together with the sketcher rules. The sketcher assumes a layered drawing of different entities and any two entities that have a connection between them must overlap with the type of joint drawn in the overlapping region. This rule is very intuitive and natural for one to overlap two

connected parts with the joint in the overlapping region. The backend interprets the parts and joints based on the overlaps and brush colors and then converts it into a graph structure.

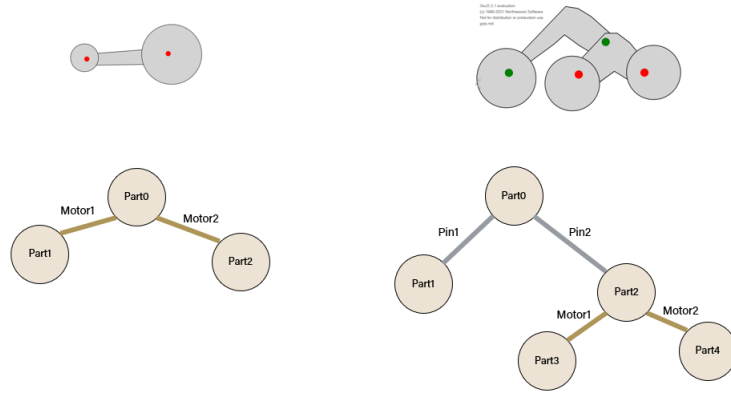


Figure 3.7. Generating Graph Model from Sketch

3.3.4 Simulator

Once the graph is generated by the sketcher backend it is then transferred to the physics engine which then applies physical properties such as mass, gravity, friction and constraints to the parts so that the entire 2D car assembly can be simulated in the preloaded terrains. The simulator acts like a instant feedback to the designer on how their design performs. The visual simulation also gives out hints to the designer on what changes would make the design perform better and help in building intuition. Now the user can create another design and iterate through the process until they are satisfied. The graph generated by the Sketcher can be downloaded as a .json file to save the design and promote file interchange with other tools.

3.4 Testing & Results

3.4.1 Testing

The system was tested for its robustness and support for different possible shape and joint combinations. Figure ?? shows the breadth of designs that the system was able to

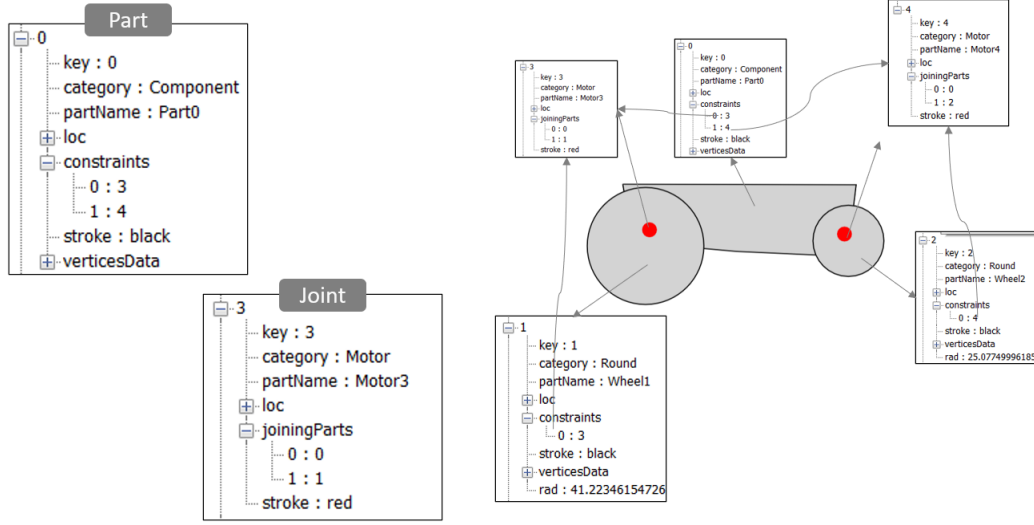


Figure 3.8. JSON file format for saving designs and transferring to other tools

successfully translate in the physics engine from the freehand sketches without losing the user’s intent. A preliminary user study was conducted with three users. The study steps involved a short practice session followed by a free play. The users were asked to ‘think aloud’ while they sketched and share their intentions. The feedback gained from the preliminary user study was used to determine the next directions of Sketch2Sim. Figure 3.11 shows the different ideas explored by one of the users.

3.4.2 Results

From the preliminary study the system’s sketching usability was validated. The Sketcher was easy to learn and allowed users to sketch their ideas freely. From the observational notes it was clear that the users were enjoying the tool and found it engaging. The instant feedback helped users identify why their design works or does not work and nudged them on what would be the possible changes that can help them improve their design. Figure 3.10 shows the timeline of different ideas generated by one of the user. It can be noted that not all designs of the same idea origin were created back-to-back but are distributed randomly. This verifies the non-linear exploratory support of the tool.

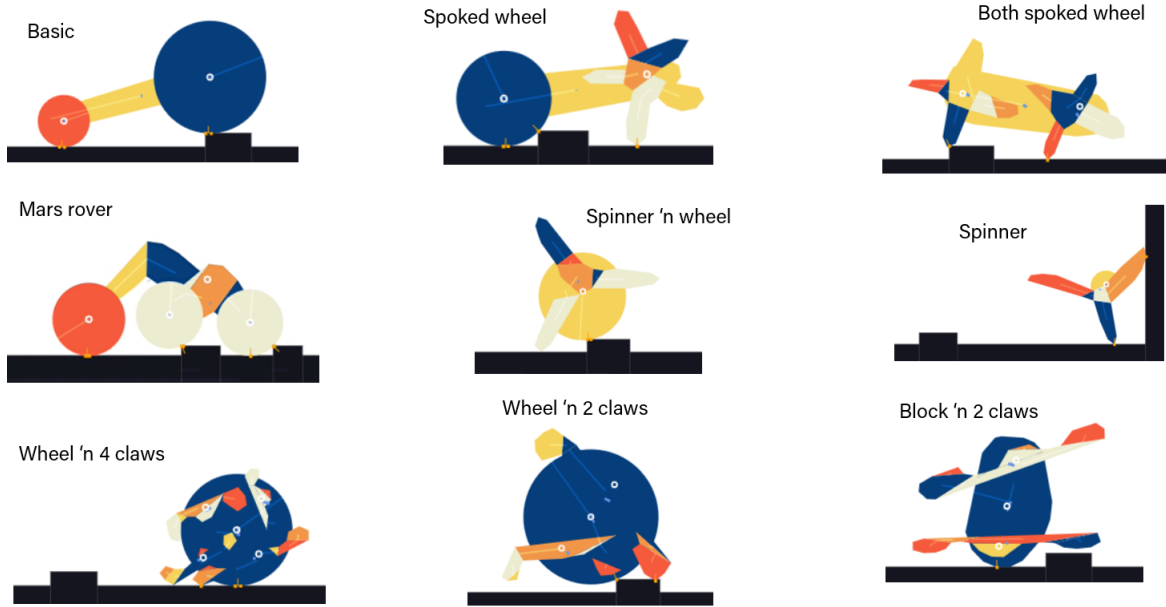


Figure 3.9. Different designs that can be supported by the tool

The designs generated by the users were clustered together into bubble of different ideas as shown in Figure 3.11. From the Idea Bubble it is evident that the user explored a large variety of ideas and even combined different ideas. Looking at the four design ideas in the orange ellipse it can be noted that the user was making incremental changes to their design based on the feedback from the simulator. The user was determined to make their design work by small tweaks in the shape of the the chassis. The two ideas in the green ellipse show how the user combined the idea of "Dead weight bhind" with "Chassis shape change to clear obstacle". Finally the design in blue ellipse shows the most optimized design after combining the two ideas and dropping the dead weight and achieving the same effect of moving the center of gravity behind by changing the shape of the chassis. One of the users also made a remark 'I like the fact that I can quickly start over if my design does not work' which is similar to throwing away a sketch and grabbing a new sheet to start afresh. The study also validated the SIAM (Search for Ideas in Associative Memory) [38] model's two cognitive states during ideation - productive ideation and impasse. During the productive ideation state users created many different ideas but after some time they exhausted their ideas and reached an impasse state.

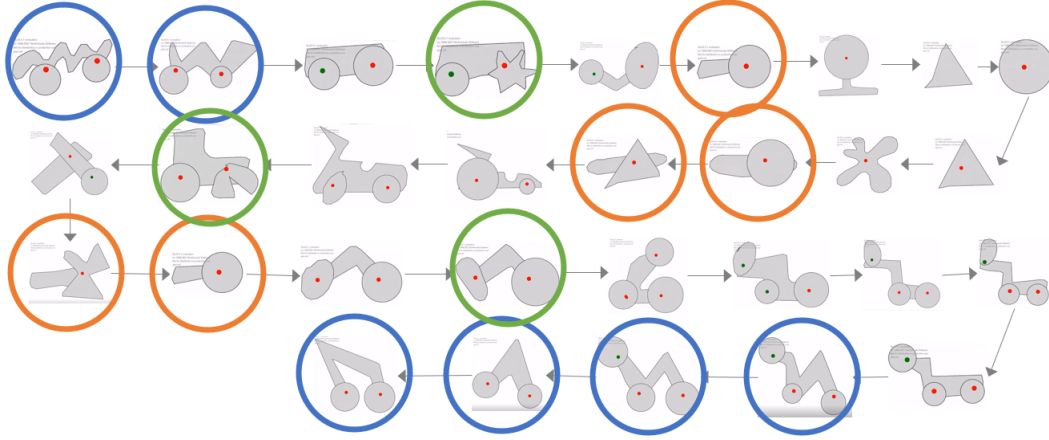


Figure 3.10. Idea Timeline: Designs based on same ideas were created at different points of time

3.5 Limitations & Future Work

The preliminary study also uncovered some limitations of the tool. Once the user ran out of ideas reached the state of cognitive impasse they did not know what to do. The system can have a feature to help the user overcome this impasse and again enter the productive state by nudging. Nudging could be done by offering some indirect hints or suggestions or inspirations to the user on detecting a long period of inactivity in the system. One of the users with expertise in sketching also expressed some frustration with the sketcher for not supporting multi-stroke sketching for the same part. In the next version the system can support this feature by grouping nearby strokes together and enclosing them into a single shape.

3.6 Conclusion

In this chapter we discussed Sketch2Sim, a sketch-based design exploration tool for 2D terrestrial vehicles. We established design principles for ideation tools based on prior work and developed a system that allows users to create freehand sketches of 2D vehicles and provides instant feedback on the design's behaviour in predefined terrains. Finally, the tool was evaluated for its usability and design exploration abilities by performing a preliminary

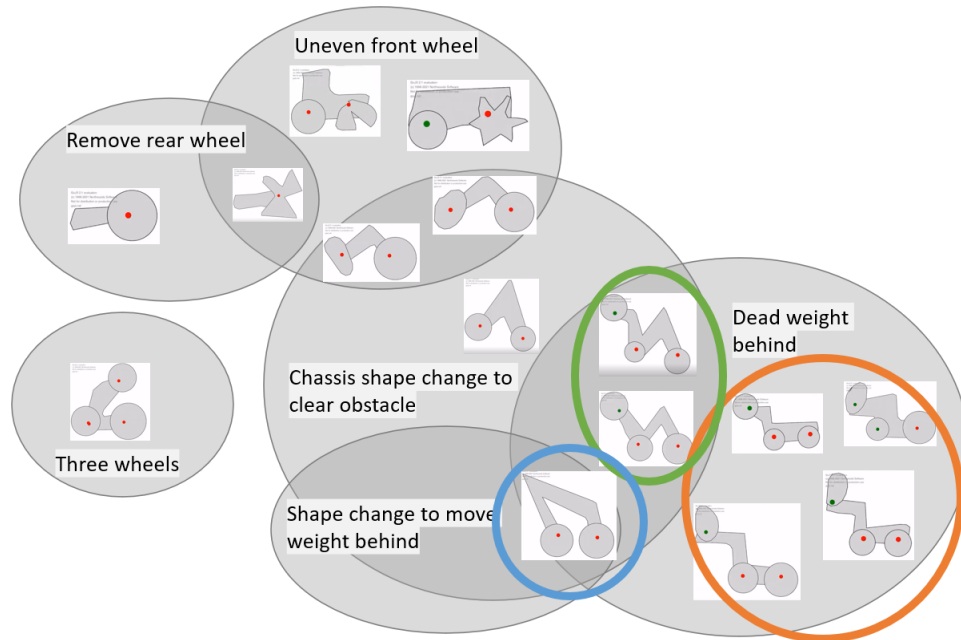


Figure 3.11. Ideation Bubble: Different ideas explored by a user to design a 2D vehicle that can overcome an obstacle

user study. The study results allowed the team to find the next directions and features to improve the tool and its ideation capabilities.

4. DESIGN INTERFACE FOR IOT DEVICES

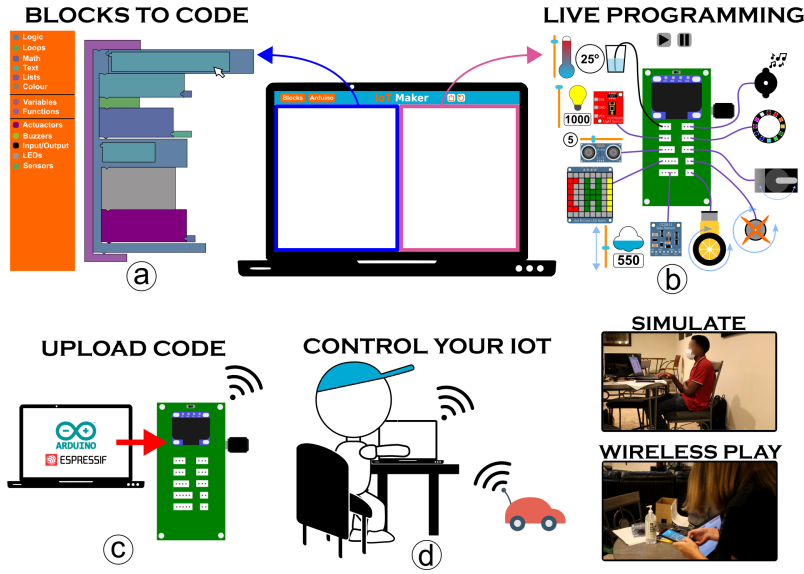


Figure 4.1. IoTMaker: A System for Ideation and Rapid Prototyping of IoT devices

4.1 Overview

IoTMaker is a system that allows users to design, build and interact with internet connected electro-mechanical devices or IoT devices. IoTMaker encourages users to 'think wild' with their ideas without constraints and provides simple user-friendly tools to translate their ideas into working prototypes of IoT devices. It is mainly targetted towards the K-12 community but it can extend to the DIY and maker community and even to expert users. The system is made of a live programming simulator that uses block-based programming language to create a simulation of the electronic components, a plug-n-play electronic board that supports a wide variety of Input/Output components, and lastly a mobile app that allows users to control their devices wirelessly and play with them. Each of the three components support the ideation, development and control aspect of the IoT devices. The system was evaluated by expert users and 15 students, and was found to be helpful in reducing the barrier of entry for creation of IoT devices with advanced functionalities. In this chapter we focus on the design, development and testing of IoTMaker's workflow.

4.2 System Design

4.2.1 Motivation

The maker movement has led to the development of simplified toolkits that make creating technological prototypes an engaging and playful experience. This has increased the accessibility of tech to individuals who are driven with creativity and curiosity but not necessarily having a technical background. Technology is also becoming a part of art and design community due to these toolkits. The underlying philosophy of these toolkits lies in the creation of 'black boxes'. How well are the complex aspects of technology abstracted and only limited information is made available to the users. This kind of abstraction is an important feature that makes these tools have lower barriers of entry. Knowing just enough about the components so as to be able to put them together to create a functional prototype gives the users the freedom to think creatively and make their imagination tangible. Creative exploration and learning the science are the two main goals of these toolkits. Although many such toolkits are available as discussed in section 2.2, there is still a lot of scope to develop and improve such toolkits to bring in a balance of creativity, learning and user experience. Having identified block-based programming coupled with a live simulation of electronics as an interesting approach to make IoT device creation an engaging and efficient in microBit [22] our team expanded on this idea to propose IoTMaker a live programming interface with an extended electronics repository of plug and play devices.

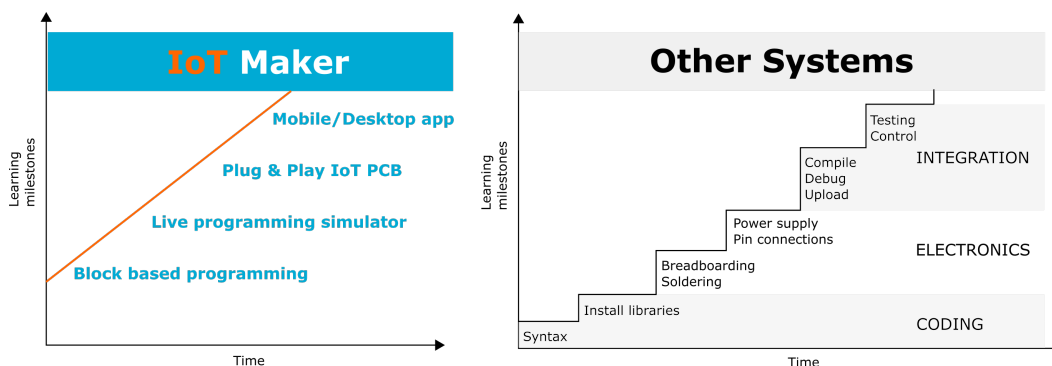


Figure 4.2. Learning curve comparison between IoTMaker and other systems

4.2.2 Design Principles

IoTMaker as a system makes transition from programming to electronics and controls continuous by: the block-based programming interface has a visual simulation of the MakAR board and the various electronics components. This familiarizes users with the components of electronics repository before they physically assemble them. User already has a visual memory of how to connect, what each component looks like and how each component functions. User is more aware of what to expect and what could possibly go wrong, thus reducing the errors and frustration. The system has carefully placed black boxes to allow more 'tinkering' and reduce frustration. The main design principles IoTMaker is based on are:

1. Low ceilings, wide walls
2. Flow
3. Play
4. Visualization
5. Open Ended

Figure 4.2 compares IoTMaker with other tools that have discrete learning milestones in each step and requires longer time until the final device is created.

4.3 Implementation

The complete system is made of three sub-systems- the live programming interface, the MakAR board and the electronics repository and a companion mobile app. This thesis shall only focus on the development of the live programming simulator and its testing.

4.3.1 Live Programming Simulator

The live programming interface is the entry point of the IoTMaker workflow. The first step of creating an IoT device is to decide which components to use and how to use them.

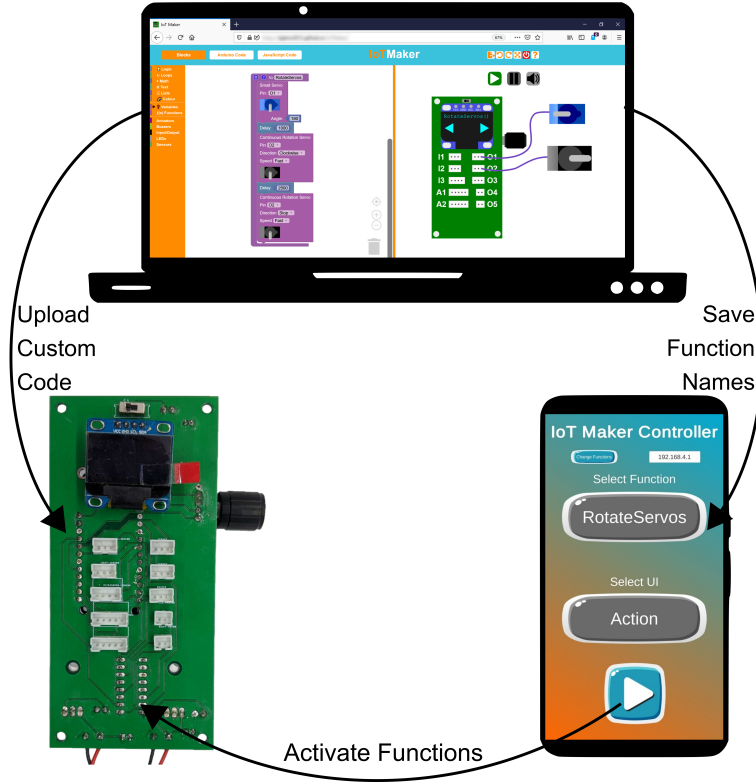


Figure 4.3. IoTMaker Workflow

The Live Programming Simulator (LPS) is a web-based interface that is divided into two parts as shown in Figure 4.4.

First the input side (left screen) which is comprised of block based programming panel that utilizes Google’s Blockly [17] project to allow users to drag and drop different component and logical blocks to create a code and then generate a JavaScript equivalent of this code. Each component of IoT device is represented by a block. The team modified a version of Blocklyduino [39] to reuse the logical blocks along with some pre-existing electronics blocks. Since we expanded the electronics repository some new blocks had to be created. This was done by using Google Blockly’s developer tools [40]. Creation of blocks is a way of abstraction of the electronic components. Instead of looking at a completely unfamiliar electronic device the user will be looking at a puzzle piece that has simple buttons or dropdowns to control the functionality of that component. It was also important to identify the functionality of the component and how it behaves in the code. For example, a motor or a servo is an output



Figure 4.4. IoTMaker’s Live Programming Simulator Interface

device and a light sensor is an input device. Thus the shape and connections of the blocks had to be carefully designed to prevent the user from creating a syntactically wrong code. The blocks design also inherently prevented the users from attaching the devices to wrong pins based on the type of the device. These inherent features in the block design allow users to focus on ideation and creativity instead of worrying about the internal details. Once the blocks are placed in the desired order the user simply hits the "Run" button which translates the block code into a javascript equivalent. This JavaScript code generated from the blocks is then passed on to the Simulator (right screen). The Simulator translates the JavaScript code into a series of visual animations that simulate how the input/output components would function if they were programmed that way. The LPS gives instant feedback on the behavior of the IoT devices and allows user to make changes and view the updated behavior. The simulation panel populates the graphics of the electronic components that are present in the block code and also supports user interactions on these graphics for input devices. For example, the temperature sensor has a slider that allows the user to change the temperature value and see real time behavior of the other components based on their code logic. This reduces the time and resources that would be spent in getting the physical prototype built and tested. The LPS allows users to see and fix potential errors that might occur in their devices early on in the design process. Thus, saving time and effort and preventing frustration. The

LPS interface is also based on the principle of supporting creativity through the block based programming panel and verifying the rationality of the idea through the visual simulation panel as discussed in Figure 3.2.

4.3.2 Hardware Prototyping ToolKit

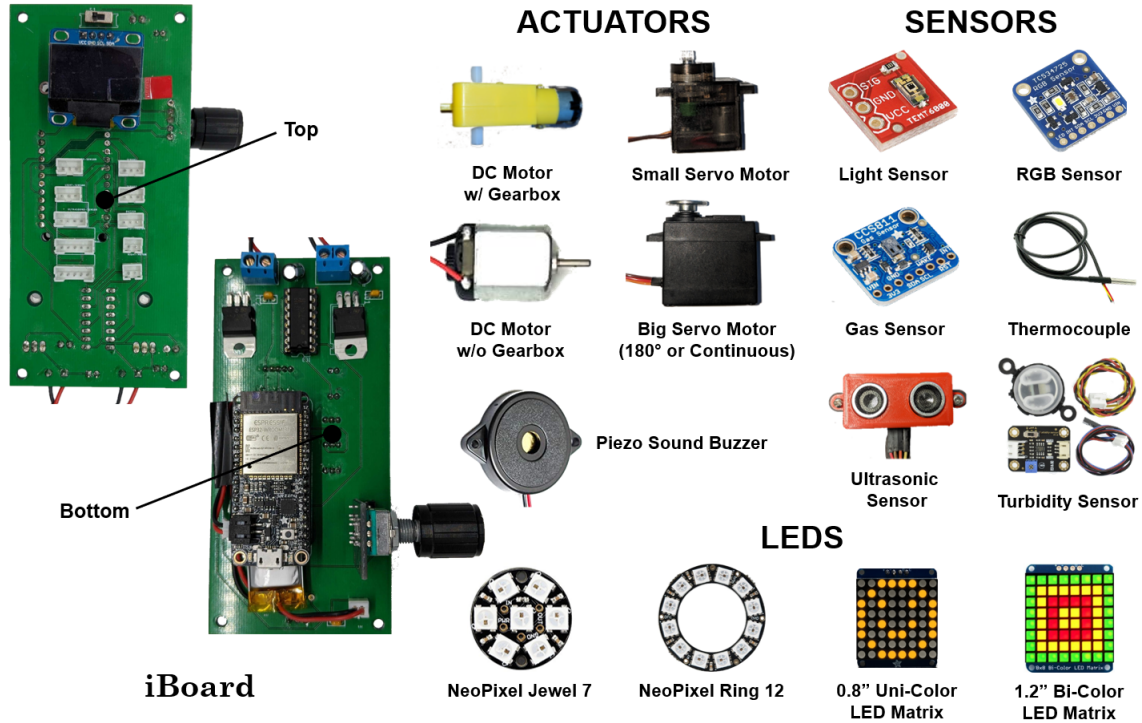


Figure 4.5. Electronics Repository

Once the user is satisfied with their code and the behavior of the IoT device in the LPS the next step is to physically put together the components. Hardware assembly can be a difficult task specially for novice users. The long datasheets, the pin connection diagrams and the power requirements can all be overwhelming. In order to address this problem, IoTMaker offers a modular hardware toolkit. The toolkit is comprised of the MakAR Board and set of input/output components. The MakAR board has an inherent PCB design that ensures correctness of pin connection and power supply while allowing users to simply plug-n-play the desired components. At the heart of the MakAR Board is the Wifi-enabled microcontroller unit (MCU), Adafruit HUZZAH32 ESP32 Feather Board [41]. The board

layout is carried out in alignment to usability principles. The various components on the board were divided into user interaction and non-user interaction components. The former were placed on the front side on the MakAR board while the later were placed on the rear side to reduce visual clutter and improve focus. The front side consists of a small OLED display and the input/output ports along with a rotary encoder knob on the side to navigate the screen. The design takes into account the Gestalt's Principles of perceptual organization [42]. The principle of proximity states that objects which are close together in space tend to be grouped together when visually inspected; whereas, the principle of continuity states that objects that are arranged in straight or curved lines tend to be grouped together. In accordance with these principles, the input/output ports were grouped separately in straight lines to help distinguish between similar input/output male pin headers on the MakAR board. The front of the MakAR board gives access to the 5 for input and 5 for output ports. In order to support a wide range of input components the MakAR board gives access to three different types of input pin headers. The 3-pin and 4-pin inputs headers connect to devices that supply simple analog/digital data to the MCU, the 5-pin input header supports I2C communication protocol thus expanding the variety of possible input/output components. Along with plug and play support to a wide range of input/output components the MakAR board also optimizes power distribution by splitting the power across three different power sources. The MCU, OLED display, and rotary encoder are powered by a 3.7-volt Lithium-Polymer battery. The input/output components are powered by two 9-volt Lithium-Ion batteries, thus allowing the MakAR board to operate with all pins and components being used simultaneously at maximum load condition. Apart from the input/output components listed in Figure 4.5 intermediate/expert users can also add more components to the list and expand the system.

4.3.3 Controller Mobile App

After the user has written the program using the Live programming simulator and assembled their hardware components onto the MakAR board, now they need to control and interact with their physical IoT devices. To enable this interaction we developed a mobile

application that acts like a remote control to trigger different functions in the devices. The app was made using Unity's 3D application development suite [43] that allows apps to be deployed on multiple platforms - smartphones and PCs. Upon startup, the mobile app interprets a text file generated by the IoTMaker Web application. This text file is comprised of all the user-created function names and this list is loaded into the app. The app allows four different types of control interactions

1. The "Action" option that is a simple button press to trigger a function
2. The "Toggle" option that activates or deactivates a function
3. The "Slider" that allows continuous increment or decrement of a value
4. The "Directional Pad" that allows four-direction control to a particular function

. The communication between the mobile app and the MakAR board is carried out using the User Datagram Protocol (UDP) packets. The UDP utilizes a Wi-Fi network and the IP address of the MakAR board to transmit commands and data between the app and the IoT device.

4.4 Testing Results

IoTMaker was evaluated through two user study procedures - Expert study and Youth study. Expert study was conducted to gain insights on the system from users that were familiar with developing and using electro-mechanical devices. Youth study was conducted with high school students most of which had none to little experience with creating electro-mechanical devices. Both the studies were followed by survey questions to evaluate the usability and learning support of the system.

4.4.1 Expert Study

The goal of the Expert study was to gain a perspective of IoTMaker's performance from the point of view of users who are experts in developing systems and tools for creating for

similar electromechanical devices. The study was based on a qualitative analysis by the experts to uncover:

1. Design considerations to support learning
2. User Experience mainly addressed towards kids
3. Usefulness of the Live Programming Simulator

4 expert users began with a pre-user study survey that involved questions related to their background and experience in working with electro-mechanical devices. Based on the responses all users were familiar and confident with creating and using electro-mechanical devices.

Followed by the pre-user study survey, the users were asked to perform the 5 coding challenges (Figure 4.6). A pseudo-code document along with instructions on how to progress from one step to the next was used as a form of instruction manual for the Coding Challenges. Following the pseudo code, users recreated the code in the IoTMaker's Live Programming Interface. Between two Coding Challenges, users had to upload their code to the MakAR board and use the Controller app to control their IoT devices. The last coding challenge was an open-ended one that allowed the users to come up with their own ideas to generate a code for a car.

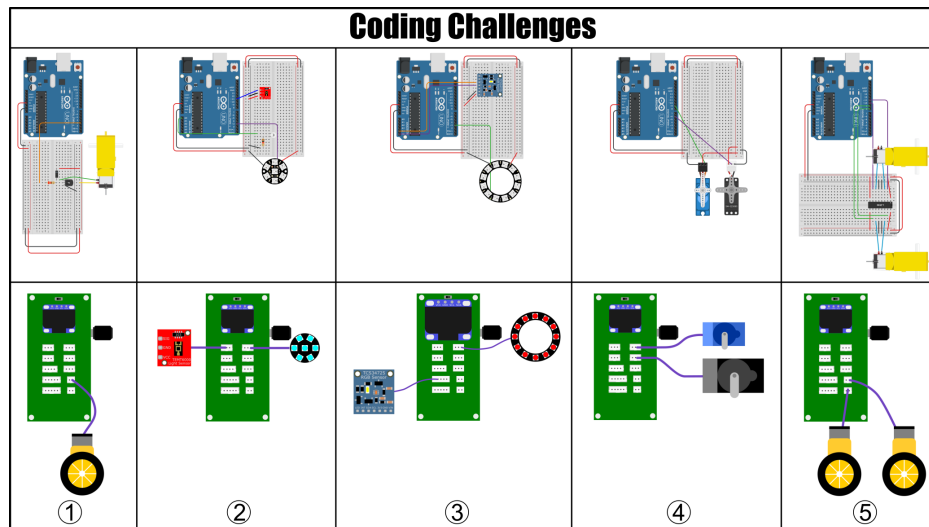


Figure 4.6. IoTMaker Coding Challenges

All the experts were able to complete all 5 coding challenges with little or no issues within the allotted time. The table below summarizes the responses to the post-user study survey, it is clear the system met the expectations of the experts and they had a smooth experience interacting with IoTMaker.

4.4.2 Youth Study

The team conducted a youth study with 15 students between the age group of 11 to 18 to evaluate IoTMaker. The aim of this study was to answer the following two research questions: RQ1 How does IoTMaker support youth in creating higher level programming concepts in a simple and robust way? RQ2 What does the MakAR board do to open the capacity for live programming complex electro-mechanical devices? The flow of the study was as follows- 1. users were asked to fill out a pre-survey that gathered data about their background and experience with EMDs, 2. Users were introduced to the system and given a brief explanation. 3. After this users were asked to perform the 4 coding challenges, skipping the challenge 4. 4. Students were asked to share their feedback via the post-user study survey and interview.

4.4.3 Results

Based on the pre-survey most students were not familiar to live programming tools ($M = 2.133$, $m = 2$, $\sigma = 0.247$, scale of 1 to 7). All the teams except one successfully completed all 4 coding challenges in 90 minutes. While most students had issues following the pseudo code they verbally mentioned it was mainly because they were too impatient about trying this new toolkit. The system was positively rated for usability and user experience. Overall the results indicated that the tool was useful to help novices learn how to navigate and manipulate electronics, that it reduces the time to build IoT devices from scratch while offering time to improve upon iterations, and significantly reduces the barrier to entry for novice users. From these Youth studies, it can be inferred that (1) IoTMaker is an attractive and usable tool for young people who are interested in learning how to program physical computing devices, (2) The plug-and-play MakAR board makes programming physical computing devices easier

and more enjoyable, and (3) Live Programming is a great tool to assist novice users in creating complex devices that would require more skill than they otherwise would possess; thus, lowering the barrier of entry and providing a higher entry point to this level of physical computing.

4.5 Limitations & Future Work

The present version of the Live Programming Simulator only generates visual feedback for the electronic components. A lot of IoT devices have mechanical components such as gears, joints and links attached to the actuators in order to perform a particular task. The present version of the interface does not account for these mechanical components and the user can run into problems of misalignment and interference only when the physical device is assembled. The interface needs to include the mechanical components into the simulator for early detection of such mechanical issues. In the next version of the Live programming interface the team plans to support a set of modular mechanical components that can be assembled by simple drag and drop operations and programmed using the same block based interface.

4.6 Conclusion

In this chapter we discussed the motivation, design, development and user testing for IoTMaker system that allows K-12 students to learn and build IoT devices. The system is made of three integral components - Live Programming Interfaces that supports ideation and creativity, the Hardware Prototyping Kit that supports simple hands-on interaction with the physical components and the mobile controller that allows remote wireless control of the IoT devices. From the user studies we evaluated the usability of the Live Programming Interface and found the overall user experience to be positively scored by the expert users and the High schoolers. It can be summarized that IoTMaker system:

1. Supports in teaching non-expert users how to create and program physical computing devices

2. Makes programming and building IoT devices an enjoyable experience
3. Allows users to be creative and helps focus on the functionality of the device over the internal technical details

Thus IoTMaker system helps in lowering the barrier of entry and provides wide walls by supporting creation of a variety of devices of varying complexity.

5. DESIGN INTERFACE FOR HAND WEARABLES

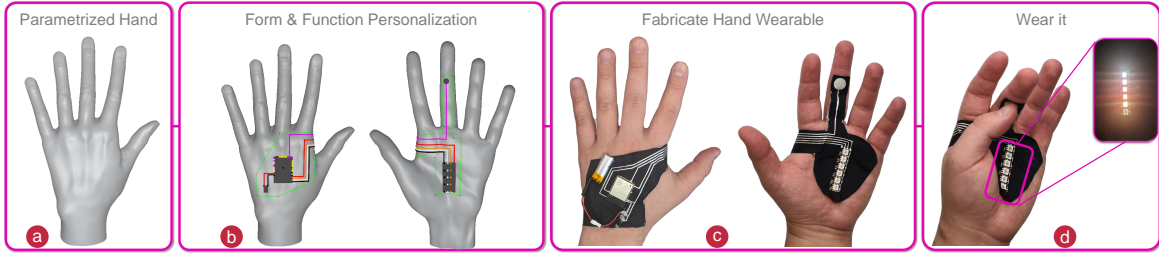


Figure 5.1. FabHandWear- An end-to-end system for Customized Functional Hand Wearables

5.1 Overview

FabHandWear(FHW) [32] is an end-to-end system to support ideation, design and fabrication of hand wearables. FabHandWear allows users with different expertise levels to create their own hand wearables. The system has a web-based application that has a preloaded parametric hand model which can be customised to the right shape and size of the user. The app then allows users to place different electronic components on the hand by simple drag and drop operation followed by creating wiring connections. In the end the app generates a 2D layout which can be printed for fabrication. Fabrication of the wearable is done using screen printing and conductive ink paint. FHW makes wearable design and fabrication accessible to the maker community and lowers the barrier of entry for users who do not have strong background in electronics or fabrication. FHW is motivated by current limitations in wearable development and is an attempt to address these limitations by offering an accessible and user-friendly workflow.

5.2 System Design

5.2.1 Motivation

Creation of electro-mechanical devices (EMD) is becoming more and more accessible with the development of open-source technologies and plug-n-play modular kits [19], [20]. Development of EMDs and relevant research in three pillars of EMDs - programming and

electronics have been discussed in section 2.2. Along with EMDs recently there has been an increasing research done in wearing these electronic devices on bodies through wearable design. Despite these tools that enable EMD development for different types of users, wearable design still stand out of these generic EMD development toolkits and assume a 'high-tech' or 'futuristic' category of EMDs that still remains limited to domain experts. FHW breaks these technical barriers towards wearable tech and offers an easy, user-friendly workflow from design to fabrication of hand wearables. FHW focusses on hand wearables and builds upon prior work discussed in section 2.3. The design objectives and system development are discussed in detail in the next section.

5.2.2 Design Principles

FanHandWear is a hand wearable tool that can enable users to translate their ideas into functional wearable prototypes. In order to achieve this goal it was important to carefully choose the 'black boxes'. User should be able to control the customizable aspects of the design but need not be exposed to the inner workings of the electronics. For example, when designing a thermal sensing interaction it is important to know what is a temperature sensor capable of doing - detect hot and cold attributes. Knowing how the sensor detects temperature, what are its power ratings and what are the different connecting terminals is not relevant at this stage. This information will only block the creative thinking of the designer and add to their cognitive load. Secondly, the fabrication process should not require sophisticated machinery and should be doable using accessible tools and processes. With these motivations the following design objectives were identified:

Design Objectives:

1. Functionality
2. Customizability
3. Fabricability
4. Visualization
5. Open Ended

5.3 Implementation

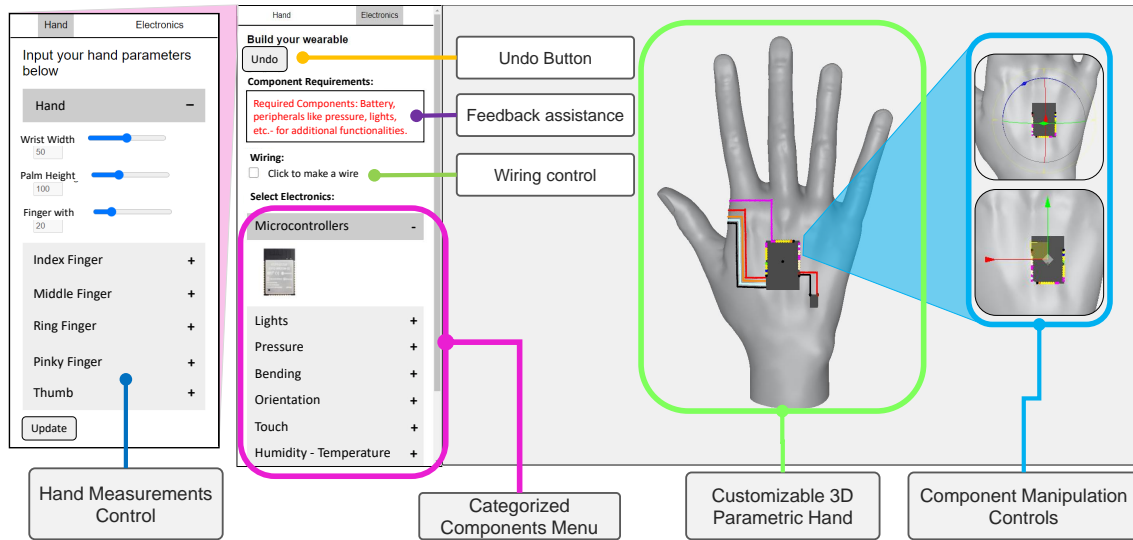


Figure 5.2. FabHandWear- Interface

The aim of the design tool is to facilitate and assist the creation of hand wearable templates which can then be used for fabrication. The design tool interface is structured into three steps - 1. Hand Customization, 2. Electronics Functionality Design, 3. Design for Fabrication support

5.3.1 Hand Customization

Hand customization is a feature that allows the users to reconfigure a default 3D hand mesh to match their hand's shape and size. This is a particularly important aspect of the workflow because there is wide range of variability in hand anatomies and it is difficult to have a 'one size fits all' kind of a hand wearable. The fit of the wearable directly impacts the comfort and ergonomics and thus is a deciding factor of user experience. Another problem with a 'one size fits all' kind of design will be the reachability of user input components such as buttons. Thus, the first step is to measure some reference dimensions of the hand and input them. The hand mesh updates to reflect the size and shape of the user's hand.

In the backend of this feature is a parameterized hand model that approximates the dimension of each section of the hand using only 4 hand dimensions. Figure 5.3 shows the

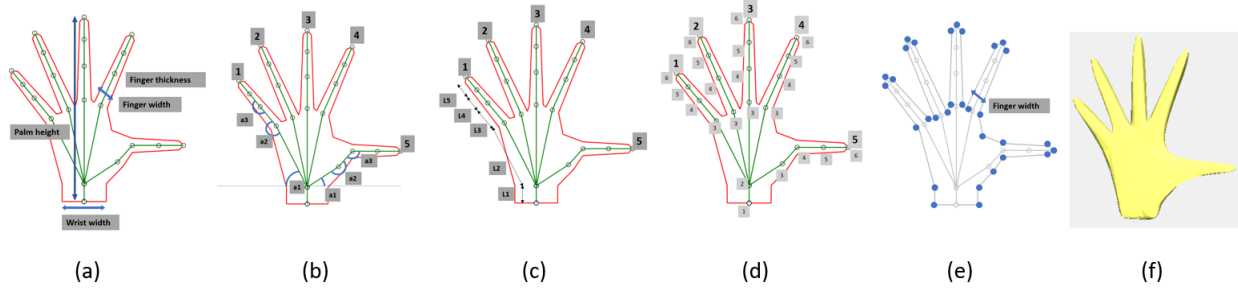


Figure 5.3. Hand Parameterization: (a) User input 4 hand dimensions; (b) 3 angle values for each finger based on initial reference measurement; (c) 5 length values for each finger based on initial reference measurement, each length expressed as multiple of palm height; (d) generation of 30 skeletal points based on b and c ; (e) generation of 31 outer points based on skeletal points and multiples of finger width; (f) extruding the 2D outer points and smoothening the edges

various steps involved in creating a parameterized hand model by starting with just 4 hand dimensions.

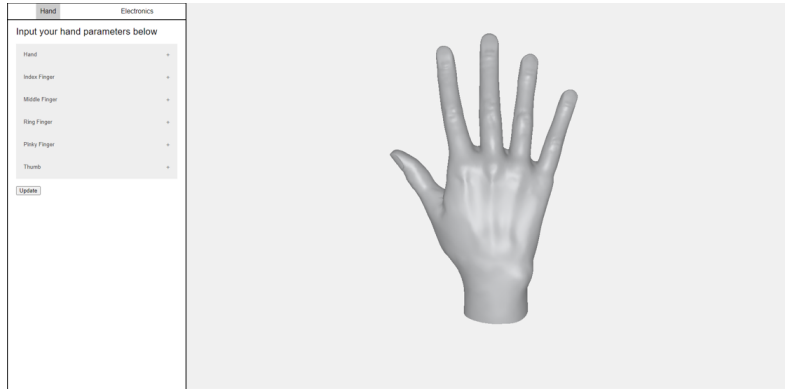


Figure 5.4. FabHandWear- Interface, Hand Customization

5.3.2 Electronics Functionality Design

The next step, Electronic Functionality Design allows users to place various electronic components on the parametric hand mesh. The interaction is such that the user drags and drops the electronic components from the side bar and on leaving the mouse the component snaps itself to the surface of the hand where it was left. At this point, the user can zoom,

pan and rotate the hand and access any area of the hand mesh to place the components. The component list is arranged in such a way that it groups them based on the functionality and makes it easier for the user to find a particular component they are looking for instead of scrolling through a list of technical names of sensors. Another feature of the EFD is to guide the user while designing by suggesting components that must go together to The electronic

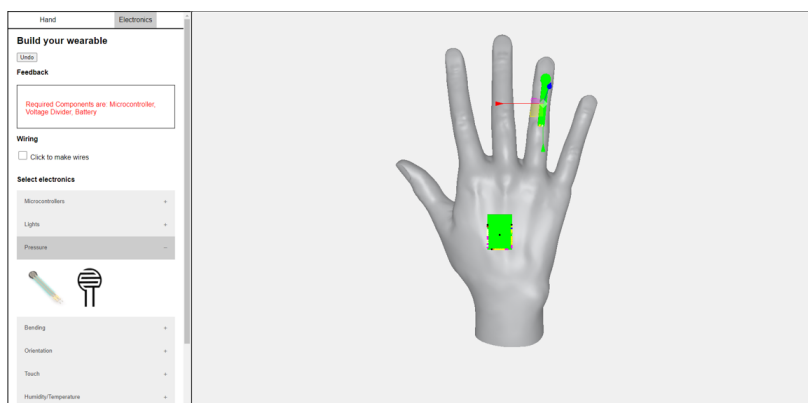


Figure 5.5. FabHandWear- Interface, Electronics Placement

components are internally stored in a XML format. Each component has the attributes that define the name, 3D model data, connecting pin location and names. These attributes are useful in creating correct connection wiring. The next step in EFD is to generate wiring diagrams. Based on the xml data of the components pins are rendered in a color coded format. The system makes use of this information to generate routing between components. Users can also create routing patterns based on their design and click on a pin of same color on two components to create a connection between them. The system does not allow connection between two pins of different color and prevents any errors.

5.3.3 Design for Fabrication support

Once the electronic components are laid out and wiring diagram generated it is time to convert the design into manufacturable file format. For this step the system translates the 3D hand model with components and wiring laid out to a flatened 2D layout. In order to generate a manufacturable wearable we need to create a substrate on which all the components shall stick. This is achieved by letting the user click on outer vertices of of the

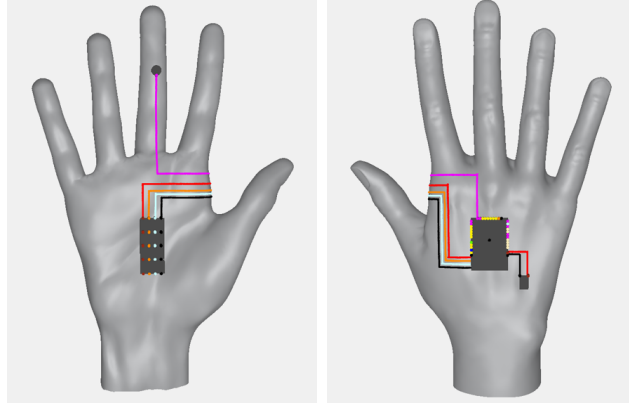


Figure 5.6. FabHandWear- Circuit routing

substrate contour. The system tracks the users consecutive clicks and renders the edges of the substrate. Following creation of the substrate in 3D it is then mapped against the 2D hand grid and opened flat using ray-casting. Now FHW can generate a 2D template that can be printed out. This template consists of the flattened substrate with electronic components retaining their relative locations and pins and the circuit lines.

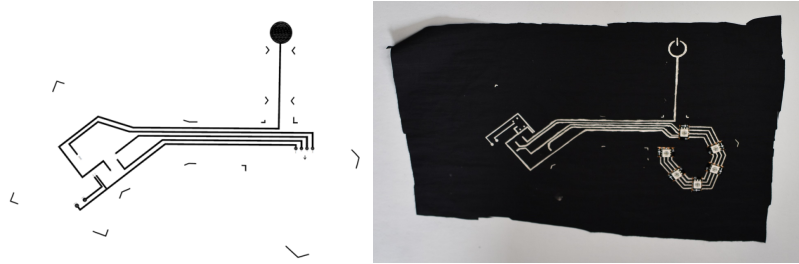


Figure 5.7. FabHandWear- Fabrication template and screen printed wearable

Fabrication of the wearable is carried out by simple processes that are accessible to designers with different backgrounds. The proposed fabrication process does not require any special tools or sophisticated machines. The steps in fabrication of the wearable involve:

1. screen print the circuit path on a piece of textile that will act as the substrate
2. cure for 5 minutes at 65 deg. C
3. stick the components and connecting pins in their marked locations using electrically conductive double-sided-tape

4. paint the connecting pins of the components with conductive ink to ensure connection with the traces
5. cure the wearable for 5 minutes at 65 deg. C
6. (optional) apply a layer of liquid bandage for durability and wear protection
7. cut the fabric shape as per the markings
8. stick the substrate fabric on the hand using skin friendly adhesive tape

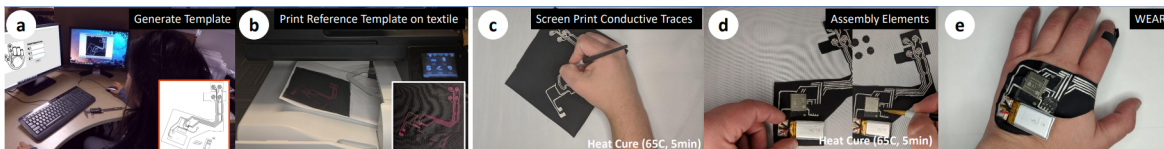


Figure 5.8. FabHandWear- Steps in Fabrications

The substrate fabric was selected based on the following properties

1. skin friendly
2. non-conductive
3. retain shape over time
4. thick enough to not let the ink seep through
5. easily available

Based on the above properties a prior research, Nylon 100% water proof (#6059 water repellent T Nylon) commercially available from good Rich Textiles International was selected. The conductive ink used was CI-1036 silver ink from Engineered Conductive Materials because it provides high conductivity, flexibility and short curing time.

5.4 Testing & Results

5.4.1 Testing

FHW was tested by demonstrating the entire workflow for 4 different applications as shown in Figure 5.9. It is clear that FHW design interface offers the ability to create a wide variety of wearables with different functionalities and a custom hand fit. The system was also evaluated for usability with individuals from non-technical backgrounds. The users were given three tasks of varying complexity and required the users to place different components on different parts of the hand.



Figure 5.9. FabHandWear- Example Applications

5.4.2 Results

All users were successfully able to design the wearables. The post study survey outcome scored the system a mean raw NASA Task Load Index (NASA TLX) of 47.5 (SD=15.083) and a System Usability Scale (SUS) score of 70.42 (SD=16.61). Figure 5.10 shows custom-fit for same wearable design on hands of different users. It is clear the variation in hand dimensions was captured by the tools and corresponding measurements were translated to the final fabrication template. The created wearable is different for each user and ensures comfort and reachability for each individual user.



Figure 5.10. FabHandWear- Custom fit wearable design on different hands

5.5 Limitations & Future work

The FHW's parametric hand model only supports a 5 finger hand structure and is not inclusive for designing for individuals with more or less than 5 fingers. In order to address this the parametric hand model must be redesigned to support a user input of number of fingers. Second limitation being, the interface does not provide any feedback on the functionality and ergonomics of the hand wearable. It is only after the user fabricates the final wearable that he is able to uncover such problems. The system can add another panel that shows the wearable in function by allowing the user to interact with the different components on screen and viewing the feedback in realtime.

5.6 Conclusion

FabHandWear is a complete end to end design and fabrication system to develop self-contained functional hand wearables, enabling multiple levels of customization of shape and functionality. FabHandWear enables accessibility to makers to encourage exploration and applications of new forms of hand wearables. We envision an emerging generation of customizable hand wearables that support hand differences and various functionalities. The design of a 3D parametrized hand enables us to control hand measurements and the precisely locating electronic components, which benefits the fit of the resultant wearable. FHW's design interface supports a guided workflow for users with little or no experience in electronics to create custom-fit functional hand wearables. From the user study we evaluated how well the

systems supports differences in hand dimensions and shapes. We were also able to verify the scope for creativity and design exploration through the example applications.

6. CONCLUSION

In this thesis we discussed the design, development and user evaluation of three design interface workflows. The first one, Sketch2Sim, a sketch-based ideation tool. The significance of "sketching" as a preferred mode of ideation was justified by creating a design interface that translates freehand sketches into physics models for 2D terrestrial vehicle. The feedback from user study verified how freehand sketching with a physics simulation engine can allow user to explore design space in breadth and depth. Active simulation from the physics engine helped user understand the problem deeply and make necessary incremental changes to their design. The second interface was the Live Programming Simulator for creating IoT devices. Inspired by microBit's [22] live simulator we redesigned a visual simulator with more number of peripheral components. The youth study results showed how IoTMaker's Live Programming Simulator supports layers of abstraction necessary to encourage creative idea generation with robust visual feedback to ensure functionality of the code before it can be run on the physical prototypes. Lastly, we looked at the third design interface, FabHandWear [32], that supports creation of custom fit hand wearables. The workflow of this interface supports unique hand 3D model generation based on only 3 hand dimensions to finally creating a 2D hand template that can be fabricated using DIY desktop fabrication tools. By developing and testing all the three design interfaces the main takeaways were a set of design rationales to consider when making such interfaces to support ideation and rapid prototyping:

1. Understand the user's background - what he knows, what doesn't know
2. Identify themes that the user is familiar with and use them as a layer of abstraction - such as freehand sketching in Sketch2Sim and puzzle pieces in the IoTMaker's Live Programming Simulator
3. Carefully place black boxes to prevent errors, to encourage focus on creation and functionality and to eliminate the need of technical knowledge or internal functioning of the final product
4. Support continuous, visual, instantaneous feedback to any changes in the user-created ideas

REFERENCES

- [1] E. Lutters, “Product development,” in *CIRP Encyclopedia of Production Engineering*, L. Laperrière and G. Reinhart, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 991–992, ISBN: 978-3-642-20617-7. DOI: [10.1007/978-3-642-20617-7_6464](https://doi.org/10.1007/978-3-642-20617-7_6464). [Online]. Available: https://doi.org/10.1007/978-3-642-20617-7_6464.
- [2] H. Plattner, C. Meinel, and U. Weinberg, *Design Thinking: Innovation lernen – Ideenwelten öffnen*. Jan. 2009, ISBN: 9783868800135.
- [3] R. F. Dam and T. Y. Siang, “What is design thinking and why is it so popular?” *The Interaction Design Foundation*, 2021. [Online]. Available: <https://www.interaction-design.org/literature/article/what-is-design-thinking-and-why-is-it-so-popular>.
- [4] D. Systemes, *3d cad design software*, 2021. [Online]. Available: <https://www.solidworks.com/>.
- [5] P. Creo, *Creo cad software: Enable the latest in design*, Aug. 2021. [Online]. Available: <https://www.ptc.com/en/products/creo>.
- [6] A. autodesk, *Autodesk empowers innovators everywhere to make the new possible*, Oct. 2021. [Online]. Available: <https://www.autodesk.com/>.
- [7] I. Ansys Inc., *Ansys / engineering simulation software*, 2021. [Online]. Available: <https://www.ansys.com/>.
- [8] C. Müller-Roterberg, *Handbook of Design Thinking*. Nov. 2018, ISBN: 978-1790435371.
- [9] M. Resnick, B. Myers, K. Nakakoji, B. Shneiderman, R. Pausch, and M. Eisenberg, “Design principles for tools to support creative thinking,” *Report of Workshop on Creativity Support Tools*, vol. 20, Jan. 2005.
- [10] Y. Lin, J. Guo, Y. Chen, C. Yao, and F. Ying, “It is your turn: Collaborative ideation with a co-creative robot through sketch,” in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI ’20, Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–14, ISBN: 9781450367080. DOI: [10.1145/3313831.3376258](https://doi.org/10.1145/3313831.3376258). [Online]. Available: <https://doi.org/10.1145/3313831.3376258>.
- [11] R. Bellamy, M. Desmond, J. Martino, P. Matchen, H. Ossher, J. Richards, and C. Swart, “Sketching tools for ideation (niet track),” in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE ’11, Waikiki, Honolulu, HI, USA: Association for Computing Machinery, 2011, pp. 808–811, ISBN: 9781450304450. DOI: [10.1145/1985793.1985909](https://doi.org/10.1145/1985793.1985909). [Online]. Available: <https://doi.org/10.1145/1985793.1985909>.

- [12] W. Benjamin, S. Chandrasegaran, D. Ramanujan, N. Elmqvist, S. Vishwanathan, and K. Ramani, “Juxtapoze: Supporting serendipity and creative expression in clipart compositions,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’14, Toronto, Ontario, Canada: Association for Computing Machinery, 2014, pp. 341–350, ISBN: 9781450324731. DOI: [10.1145/2556288.2557327](https://doi.org/10.1145/2556288.2557327). [Online]. Available: <https://doi.org/10.1145/2556288.2557327>.
- [13] K. D. Willis and J. Hina, “Alchemy: Experiments in interactive drawing, creativity, and serendipity,” in *Proceedings of the Seventh ACM Conference on Creativity and Cognition*, ser. Camp;C ’09, Berkeley, California, USA: Association for Computing Machinery, 2009, pp. 441–442, ISBN: 9781605588650. DOI: [10.1145/1640233.1640342](https://doi.org/10.1145/1640233.1640342). [Online]. Available: <https://doi.org/10.1145/1640233.1640342>.
- [14] C. Piya, V. -, S. Chandrasegaran, N. Elmqvist, and K. Ramani, “Co-3deator: A team-first collaborative 3d design ideation tool,” in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI ’17, Denver, Colorado, USA: Association for Computing Machinery, 2017, pp. 6581–6592, ISBN: 9781450346559. DOI: [10.1145/3025453.3025825](https://doi.org/10.1145/3025453.3025825). [Online]. Available: <https://doi.org/10.1145/3025453.3025825>.
- [15] M. Keshavarzi, C. Hotson, C.-Y. Cheng, M. Nourbakhsh, M. Bergin, and M. Rahmani Asl, “Sketchopt: Sketch-based parametric model retrieval for generative design,” in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2021, ISBN: 9781450380959. [Online]. Available: <https://doi.org/10.1145/3411763.3451620>.
- [16] R. H. Kazi, T. Grossman, H. Cheong, A. Hashemi, and G. Fitzmaurice, “Dreamsketch: Early stage 3d design explorations with sketching and generative design,” in *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST ’17, Québec City, QC, Canada: Association for Computing Machinery, 2017, pp. 401–414, ISBN: 9781450349819. DOI: [10.1145/3126594.3126662](https://doi.org/10.1145/3126594.3126662). [Online]. Available: <https://doi.org/10.1145/3126594.3126662>.
- [17] Google, *Blockly*, en-us, 2020. [Online]. Available: <https://developers.google.com/blockly/>.
- [18] S. Scratch, *Imagine, program, share*, 2021. [Online]. Available: <https://scratch.mit.edu/>.
- [19] Arduino, *Arduino - home*, 2020. [Online]. Available: <https://www.arduino.cc/>.
- [20] R. Pi, *Teach, learn, and make with raspberry pi – raspberry pi*, 2019. [Online]. Available: <https://www.raspberrypi.org>.

- [21] littleBits, *littleBits / electronic building blocks for the 21st century*, 2021. [Online]. Available: <https://sphero.com/>.
- [22] (). “Micro:bit educational foundation,” [Online]. Available: <https://microbit.org/>.
- [23] T. L. Group, *Lego mindstorms*, <https://www.lego.com/en-us/mindstorms>, 2020.
- [24] VEX, *VEX Robotics*, <https://www.vexrobotics.com/>, 2020.
- [25] K. Katsuragawa, J. Wang, Z. Shan, N. Ouyang, O. Abari, and D. Vogel, “Tip-tap: Battery-free discrete 2d fingertip input,” in *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, 2019, pp. 1045–1057.
- [26] D.-Y. Huang, L. Chan, S. Yang, F. Wang, R.-H. Liang, D.-N. Yang, Y.-P. Hung, and B.-Y. Chen, “Digitspace: Designing thumb-to-fingers touch interfaces for one-handed and eyes-free interactions,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 1526–1537.
- [27] C. Zhang, X. Wang, A. Waghmare, S. Jain, T. Ploetz, O. T. Inan, T. E. Starner, and G. D. Abowd, “Fingorbits: Interaction with wearables using synchronized thumb movements,” in *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, 2017, pp. 62–65.
- [28] P. C. Wong, K. Zhu, and H. Fu, “Fingert9: Leveraging thumb-to-finger interaction for same-side-hand text entry on smartwatches,” in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–10.
- [29] C. Zhang, Q. Xue, A. Waghmare, R. Meng, S. Jain, Y. Han, X. Li, K. Cunefare, T. Ploetz, T. Starner, *et al.*, “Fingerping: Recognizing fine-grained hand poses using active acoustic on-body sensing,” in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–10.
- [30] E. Markvicka, G. Wang, Y.-C. Lee, G. Laput, C. Majidi, and L. Yao, “Electrodermis: Fully untethered, stretchable, and highly-customizable electronic bandages,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ACM, 2019, p. 632.
- [31] D. Groeger and J. Steimle, “Lasec: Instant fabrication of stretchable circuits using a laser cutter,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–14.

- [32] L. Paredes, S. S. Reddy, S. Chidambaram, D. Vagholkar, Y. Zhang, B. Benes, and K. Ramani, “Fabhandwear: An end-to-end pipeline from design to fabrication of customized functional hand wearables,” *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 5, no. 2, Jun. 2021. DOI: [10.1145/3463518](https://doi.org/10.1145/3463518). [Online]. Available: <https://doi.org/10.1145/3463518>.
- [33] M. Resnick, B. A. Myers, K. Nakakoji, B. Shneiderman, R. F. Pausch, T. Selker, and M. Eisenberg, “Design principles for tools to support creative thinking,” 2005.
- [34] i. design-org interaction, *What is ideation?* 2021. [Online]. Available: <https://www.interaction-design.org/literature/topics/ideation>.
- [35] R. React, *React – a javascript library for building user interfaces*, 2021. [Online]. Available: <https://reactjs.org/>.
- [36] m. matter matter, *Matter.js*, 2021. [Online]. Available: <https://brm.io/matter-js/>.
- [37] g. gojs gojs, *Gojs*, 2021. [Online]. Available: <https://gojs.net/latest/index.html>.
- [38] B. Nijstad and W. Stroebe, “How the group affects the mind: A cognitive model of idea generation in groups,” *Personality and social psychology review : an official journal of the Society for Personality and Social Psychology, Inc*, vol. 10, pp. 186–213, Feb. 2006. DOI: [10.1207/s15327957pspr1003_1](https://doi.org/10.1207/s15327957pspr1003_1).
- [39] BlocklyDuino, *BlocklyDuino/BlocklyDuino*, original-date: 2012-10-12T13:48:09Z, BlocklyDuino, 2019-09-10. [Online]. Available: <https://github.com/BlocklyDuino/BlocklyDuino>.
- [40] B.-d.-t. Blockly-developer-tools, *Blockly developer tools* *nbsp;/nbsp; google developers*, 2021. [Online]. Available: <https://developers.google.com/blockly/guides/create-custom-blocks/blockly-developer-tools>.
- [41] A. Adafruit. (). “Adafruit HUZZAH32 – ESP32 feather board (pre-soldered),” Adafruit.com, [Online]. Available: <https://www.adafruit.com/product/3591>.
- [42] S. Coren and J. S. Girgus, “Principles of perceptual organization and spatial distortion: The gestalt illusions,” *Journal of Experimental Psychology: Human Perception and Performance*, vol. 6, no. 3, p. 404, 1980.
- [43] Unity, *Unity real-time development platform / 3d, 2d vr & ar engine*, 2020. [Online]. Available: <https://unity.com/>.