2015-11-01

Building 3D-Printed Widgets to Incorporate into Prototypes

David E. Brandt
Brigham Young University - Provo

Building 3D-Printed Widgets to Incorporate into Prototypes

David E. Brandt

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Michael D. Jones, Chair
Kevin Darrell Seppi
Yiu-Kai Dennis Ng

Department of Computer Science

Brigham Young University

November 2015

ABSTRACT

Building 3D-Printed Widgets to Incorporate into Prototypes

David E. Brandt
Department of Computer Science, BYU
Master of Science

Creating interactive prototypes can be a long and difficult process. It requires expertise in various fields. Prior work in developing interactive prototypes minimize time required to make a prototype, but generally sacrifice fidelity for fluidity. Advances in 3D printing create new opportunities to prototype with greater fidelity and fluidity. We investigate the use of several kinds of sensors, including IR photo interrupters, IR photo reflectors, push button switches, and potentiometers, to create interactive prototypes. We first design a library of 3D printable interaction components, buttons, sliders, and knobs using those sensors then we develop software to transform interaction events into events in computer programs. The combinations of interaction components and sensing devices are evaluated based on their durability and ability to be printed into prototypes and used as human-computer interface devices.

Keywords: PhysiComps, Widgets, 3D-Printed

ACKNOWLEDGEMENTS

I would like to express my thanks and appreciation first and foremost to my advisor Dr. Michael Jones for believing in me, providing me with support, and guiding me through the process of completing this thesis. I would also like to thank Dr. Dan Olsen and Dr. Kevin Seppi for providing me with valuable feedback on my work, and Dr. Dennis Ng for supporting me.

I thank my fellow lab mates for their support. I especially want to mention Hunter Hofstrand and Casey Walker for helping me with some of the designs for the widgets and prototypes and for their help creating images for my thesis. I would like to thank Tim Price and Kristian Sims for answering any questions I had with hardware. I thank Kyler Tolman from the compliant mechanisms lab for helping me understand the bistable mechanism model, which led to the creation of the cantilever button. Finally, I would like to thank my family for their support and for always believing in me.

Table of Contents

Table of Contents....iv

Table of Figures....v

Table of Tables....vii

Chapter 1: Introduction....1

1.1 Thesis Statement....5

1.2 Organization....6

Chapter 2: Related Work....7

2.1 Widget toolkits for interaction with physical devices....7

2.2 Cameras for interaction with physical devices....10

Chapter 3: Component Designs....12

3.1 Evaluation Criteria....12

3.2 Buttons....14

3.2.1 Cantilever Button....15

3.2.2 Compliant Mechanism Button....16

3.2.3 Pop Cap Button....20

3.2.4 Spring Button....21

3.2.5 Magnetic button....22

3.3 Slider....23

3.4 Knob....24

3.5 Summary Table....25

Chapter 4: Widgets....26

4.1 Mechanical Sensors as Widgets....27

4.1.1 Buttons....28

4.1.2 Slider and Knob....29

4.2 IR Photo Reflectors as Widgets....30

4.2.1 Buttons....32

4.2.2 Slider....33

4.2.3 Knob 34
4.3 IR Photo Interrupters as Widgets 38
4.3.1 Buttons 39
4.3.2 Gray Code Slider 40
4.3.3 Gray Code Knob 44
4.4 Modular Widgets 46
4.5 Grid Array 47
4.5.1 Button 48
4.5.2 Slider 48
4.5.3 Knob 50
Chapter 5: Prototypes 52
5.1 Software Architecture 52
5.2 Ski Pole Handle 54
5.3 Etch-a-Sketch Controller 56
5.4 3D Mouse 58
5.5 Grid Array Prototype 59
Chapter 6: Conclusion and Future Work 61
6.1 Future Work 62
References 64

Table of Figures

Figure 1.1 Widget 2
Figure 1.2 Sensors and Prototype 5
Figure 3.1 Print Orientation 14
Figure 3.2 Cantilever Button 15
Figure 3.3: Compliant Mechanism Button 16
Figure 3.4: PRBM for one half of compliant mechanism 17
Figure 3.5: Monostable Model 19
Figure 3.6 Pop Cap Button 20

Figure 3.7 Spring Button 21
Figure 3.8 Magnetic Button 22
Figure 3.9 Base Slider 23
Figure 3.10 Base Knob 24
Figure 4.1 Front to back: Push button switch, Rotary potentiometer, Slide Potentiometer 27
Figure 4.2 Button Interaction 28
Figure 4.3 Slide Potentiometer Widget 29
Figure 4.4 Slide Potentiometer Trace 29
Figure 4.5 IR Photo Reflector 30
Figure 4.6 Reflectance Test Device 31
Figure 4.7 IR Photo Reflector with Compliant Mechanism Button 32
Figure 4.8 Slider with IR Photo Reflector 33
Figure 4.9 Knob with IR Photo Reflector 34
Figure 4.10 Knob States and Traces 35
Figure 4.11 Three Sensors Trace 37
Figure 4.12 IR Photo Interrupter 38
Figure 4.13 IR Photo Interrupter with Cantilever Button 39
Figure 4.14 Gray Code Slider 40
Figure 4.15 Custom Gray Code PCBs 41
Figure 4.16 4-Bit Gray Code Sequence 42
Figure 4.17 Gray Code Knob 44
Figure 4.18 4-Bit Circular Gray Code 45
Figure 4.19 Modular Widget 46
Figure 4.20 Grid Array of Proximity Sensors 47
Figure 4.21 Grid Slider Example 49
Figure 5.1 Overview of Software Architecture 53
Figure 5.2 Ski Pole Handle 54
Figure 5.3 Etch-a-Sketch Controller 56

Figure 5.4 3D Mouse 58

Figure 5.5 Grid Array Prototype 59

Table of Tables

Table 3.1 Units for Compliant Mechanism Model 18

Table 3.2 Example Parameters 19

Table 3.3 Features of Component Designs 25

Table 4.1 Color Reflectance Values 32

Chapter 1: Introduction

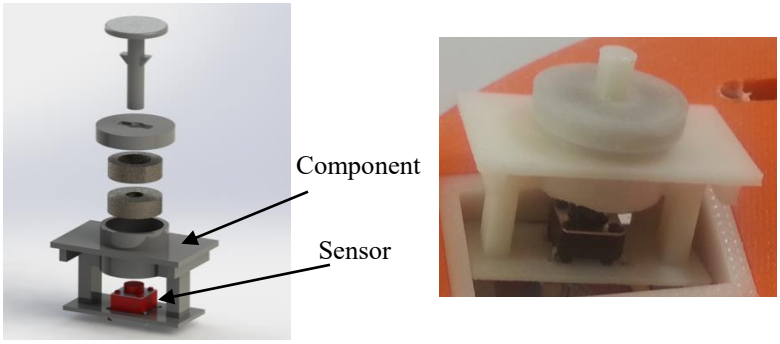
A physical interactive computing device (PhysiComp) is a small handheld device that includes sensors and computing. A defining feature of a PhysiComp is that the physical shape is driven by the intended purpose and human factors. PhysiComps rarely take the form of a rectangular box. They are designed for usability and comfort.

Creating 3D prototypes of PhysiComps requires a significant amount of time and expertise. The process often involves designers, engineers and modelers. The designers create the look and feel of the device, but have to wait until the engineers are done creating the electronics to have a fully working prototype. 3D CAD modelers then come in and add any mounting brackets necessary to enable the device to contain the electronics. The design may require revisions to work with the electronics. This process can take weeks and involve the expertise of several people, reducing the number of iterations a prototype can feasibly go through.

3D printers have opened the door for iterating through prototypes much more quickly. Once a device is designed and modeled, it can be printed and tested in a short amount of time. The model can then be changed and reprinted efficiently many times with little time or cost overhead. Unfortunately, making a PhysiComp interactive still requires a significant amount of expertise.

It is not enough to just model a device in 3D modeling software. Doing so would neglect the importance of finding a form factor for the device that fits well and feels comfortable in a user's hand. For this purpose, we have developed a design flow in which designers mold their prototypes with clay or foam, so that they can easily make small adjustments until the device feels right. After the device is sculpted, interactive widgets need to be added at the right places

on the surface to create a PhysiComp. My thesis will lay the ground work for adding interactive widgets to PhysiComps.



A widget is constructed from a component and a sensor. A component (gray in left image) is a 3D-printed part, and any non-printed parts (like magnets or springs) if necessary, used to interface with both human user and sensor. A sensor (red in left image) interfaces with a component and outputs data.

Figure 1.1 Widget

A widget is a part designed for human interaction that can be incorporated into PhysiComps. It is made up of a component that interfaces with one or more sensors. A component is a 3D-printed part, and any non-printed parts (like magnets or springs) if necessary, that interface with both the human user and a sensor. A sensor is a piece of hardware with which the component interfaces and outputs data. The component interfaces with the sensor by changing its state. The way that a component changes sensor state depends on the sensor (see chapter 4 for more details). As an example, Figure 1.1 shows a magnetic button widget. The component consists of the mounting geometry for the sensor, the magnetic button above the sensor, the magnets that are inside the magnetic button, and the plunger that interacts with the sensor. The sensor is a push button switch mounted on the component. When the magnetic button is pushed, a plunger pushes the push button switch down causing the sensor to change state.

Adding interactive widgets to PhysiComps is important to designers involved in the prototyping process. It is a challenge for them to add interactivity to a PhysiComp without help from experts in other fields. Further adding interactivity to PhysiComps is a challenge to them because it requires knowledge in programming and hardware. A toolkit that enables designers to add interaction to their device while still giving them the freedom to iterate over designs will allow them to more efficiently hone in on a preferred design.

The following scenario gives a broad overview of how the widgets can be used to create a PhysiComp in order to show how this work contributes to the future work. Imagine that a designer named Sally is creating a simple device that can control an mp3 player. It might have a skip backward, play/pause, and skip forward button widget, and a slider widget to turn the volume up and down. The device is in the form of a ski pole handle and Sally wants to make it comfortable to hold and use. She envisions the shape of the device and begins to sculpt it out of clay. As she is sculpting, she also thinks about where the component parts of the widgets can be placed to fit into her design. She gets it to the point where it feels comfortable to use with one hand and lets the sculpture dry.

The next day Sally comes back and she is not quite satisfied with the shape, so she files down some of the curves to create a better fit for her hand. Finally, the sculpture is to the point where she is ready to create a digital model and make it interactive. She marks the places on the sculpture where she wants to place the button and slider components.

Sally creates the digital model by scanning the sculpture. She opens the model inside of a CAD program. She then runs a tool that detects where she has placed the markings for her components. It finds those markings and replaces them with the corresponding components. The

tool enables Sally to modify the size and shape of the button components, and the size of the slider component.

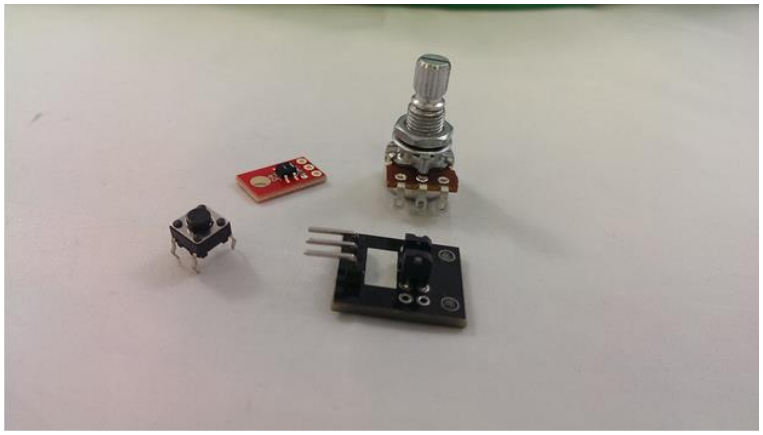
She decides that she is satisfied with the model and sends it off to a 3D printer to be printed. She pulls the 3D-printed model out of the printer and attaches the sensors that she needs to make it interactive and functional. The sensors are attached to a programmable chip that can read the sensor data, and include a push button switch for each button component, and a slide potentiometer for the slider component. With the sensors attached to their corresponding components (see Figure 1.1), Sally now has a PhysiComp with fully assembled widgets. She uses a program to link the button and slider widgets with their functions. For example, the slider widget would be linked to controlling the volume of the MP3 player.

Sally has created a fully functioning prototype that allows her to test her design. She decides that she does not like the position of the play/pause button component or the slider component. So, she opens up the CAD program again to edit the placement of the components. She uses the tool to tweak the position of the components and then reprints the prototype. She removes the hardware from her first prototype and places it into the modified version to test it.

This process of modifying and tweaking the position of components can continue until Sally is satisfied. It requires little effort to change the prototype and new designs can be iterated over quickly and efficiently.

In this thesis, we support the realization of this scenario by exploring ideas for the assembly and placement of 3D-printed widgets. The combination of the sensors and components should make it easier to prototype interactive PhysiComps and give designers more fluidity when creating prototypes. The widgets should be durable for repeated use in a prototype, printable in any orientation to support a fluid design exploration and provide realistic pleasing haptic

feedback to users. We study how push button switches, potentiometers, infrared (IR) photo interrupters, and IR photo reflectors can be used to transform mechanical motion from 3D-printed components into computing events. The components are most often the elements that the user interacts with so it is important that they are durable and provide haptic feedback. The components are printed and ideally should be printable in any orientation on the surface. The sensors are used to implement interaction in the widgets in PhysiComps so they will be evaluated based on their usability in a widget implementation. This includes the size and shape of their interaction region, sensitivity to interaction events, and programmability). Figure 1.2a shows some of the sensors that are used inside of PhysiComps. Figure 1.2b shows a ski pole handle that controls an mp3 player using the sensors and components created in this project.



(a) Sensors: from left to right: Push Button Switch, IR Photo Reflector, IR Photo Interrupter, Knob Potentiometer

(b) Ski Pole Handle Prototype

Figure 1.2 Sensors and Prototype

1.1 Thesis Statement

3D printing is an effective means by which to build durable interactive widgets that can be used in any orientation and which are pleasing human-computer interface devices that can be incorporated into PhysiComps.

1.2 Organization

The rest of the thesis is organized as follows. Chapter 2 presents related work. Chapter 3 discusses the base component designs for the buttons, slider, and knobs. Chapter 4 describes the sensors and how they were incorporated into the components to create functioning widgets. Chapter 5 shows examples of PhysiComps using the widgets. Chapter 6 discusses conclusions and future work.

Chapter 2: Related Work

Much work has been done to make it easier for designers to prototype physical devices. Some develop specialized hardware toolkits that are easy to set up and incorporate into a prototype. Others use sensors to detect interaction in order to eliminate the need for hardware toolkits, which can be bulky and make it difficult for designers to form their device in the way they want. The following sections describe some of the work that has been done and how it relates to this work. Compared to prior work, this work presents 3D printable components for use with common sensors that are simple use.

2.1 Widget toolkits for interaction with physical devices

D.tools was developed with the idea of creating a tool based on how designers think in the prototyping phase (Hartmann et al., 2006). It uses pre-defined hardware, such as buttons, sliders, switches, LCD screens, and LEDs, that are customizable and can be programmed with a simple-to-use visual language that is extendable. This toolkit assists designers with hardware, but they have to figure out how to incorporate the hardware into their prototype. We seek to create widgets where sensor interaction is built in.

Hudson created the BOXES system (Hudson and Manko, 2006). It essentially consists of thumbtacks, wires and hardware boards. When they are touched, the boards sense capacitance and register an event. One difficulty with the system is that it only works well with thin materials like cardboard. This results in it being difficult to prototype 3D shapes. We were inspired by the physicality of the method and the fluidity of the prototype system, but aim for a higher fidelity prototype of the physical form while retaining similar fluidity.

Midas is similar to the BOXES system in that it uses capacitive touch sensors (Savage et al., 2012). It is more customizable as the touch interface can be attached to many different types

of materials and can easily be molded to fit most shapes. Midas can send touch events by simulating the mouse and keyboard or by sending instructions to a device via a WebSocket. While Midas is customizable, it requires the touch sensors be fabricated for each device. We created a toolkit that has readymade electronics that are quicker and easier to incorporate into a prototype. The toolkit also has a wider range of interactive experiences because we use moving physical widgets rather than just tape.

Gadgeteer is a system designed with the idea of making the creation of new physical computing prototypes accessible, flexible, and versatile (Villar et al., 2012). A number of components were created that can be incorporated and programmed with little experience. Additionally, the library can be easily incorporated into a 3D-printed model, or laser-cut housing. We build on these ideas by making it easier for designers to incorporate interactive widgets into their 3D models with little modeling experience. This is achieved by creating a library of 3D printable components that are designed around interaction with a sensor. The component designs can be printed and incorporated into a prototype to add interaction quickly. Any special mounting geometry for the sensor is included in the component.

Switcharoo gives designers the ability to quickly create interactive prototypes while still focusing on form and feel (Avrahami and Hudson, 2002). Designers are able to create models out of soft materials like foam. They can then take components from Switcharoo and pin them on to their models to make them interactive. The Calder toolkit is similar to Switcharoo. It gives designers a set of tools, similar to GUI toolkits for software developers that can be used to make their prototypes interactive with little technical background (Lee et al., 2004). The components are also attached to foam or clay models. Rather than using just using soft modeling materials like foam, we extend this work to use 3D printers, which give designers the ability to incorporate

interactive components directly into their prototypes. 3D printers enable higher fidelity prototypes with lower fluidity.

Lamello uses acoustics for sensing interaction (Savage et al., 2015). 3D printable interactive components such as buttons, sliders, and dials were made with tines. A moving part strikes the tines as it moves generating sound that is detected by an audio sensor. The sound can then be processed and used to understand the interaction. A limitation of Lamello is that it can only detect interaction when the position of a moving part changes. It also is not yet able to distinguish between multiple components that are moved at the same time. Our work also uses 3D printable components, but with simpler sensors that are able to detect static position as well as change in position. Multiple components can be used distinguished and used simultaneously as well.

Vázquez developed a method of detecting interaction using pneumatics (Vázquez et al., 2015). The interactive controls were 3D-printed with air chambers, and use pressure sensors to detect user input. Interestingly, they use elastic material for their controls. For example, the pneumatic button contains a chamber made out of elastic material. As air is pumped in, the chamber inflates and can be used to simulate a button when a user presses it. The pneumatic controls are customizable in terms of shape and feel. Different shapes can be 3D-printed and the amount of air pumped into the chambers can change the feel of the controls. The method is limited in that it requires that an air compressor be attached to each control. The controls are also complicated to print as they often require components to be added mid print. We seek to simplify the process of making interactive controls by using simpler sensors and designs that can be printed and used without a complicated assembly process.

Numerous other design toolkits have been created to make the prototyping process faster and easier by making the electronics modular including MetaCricket (Martin et al., 2000), Phidgets (Greenberg and Fitchett, 2001), Smart-Its (Gellersen et al., 2004), littleBits (Bdeir, 2009), and xTel (Tokuhisa et al., 2009). All these toolkits integrate well with design prototypes and help designers, with little to no engineering or programming experience, develop their own prototypes. We seek to more fully integrate an easy to use toolkit into 3D printing and focus on interaction widgets.

2.2 Cameras for interaction with physical devices

We compare our work with work that uses optical sensors because they relate to our use of IR sensors. Cameras are used to detect color and texture data, which can be processed and understood. IR sensors are much simpler as they can only detect light levels, but both sensors are optical and can be used to detect interaction for more than one type of widget.

Akaoka created DisplayObjects, a system that projects a virtual model of a device prototype onto a physical model designed by the user (Akaoka et al., 2010). Interactive elements such as buttons can be created and placed onto the device. Interaction is picked up by sensors and cameras. While this allows more freedom for designers to explore new devices as well as test their physical prototypes as if they were functional, the tools are limited to use inside a motion capture arena. They also lack physical versions of the interactive elements.

Savage created a tool called Sauron that uses a single camera and image processing techniques to create an interactive 3D printable device (Savage et al., 2013). The inside of the object has to be illuminated and each interactive component needs to be registered so that the camera can distinguish between the components. All components must be visible within the field of view of the camera. Rather than using a camera with optical flow implemented in software we

use a variety of light sensing and processing tools. Using different technology to sense interaction events creates a new set of possibilities in prototyping. The sensors we use are more self-contained. They do not require the use of a camera attached to a computer running an image processing algorithm to generate interaction data. Instead, the sensors collect data and send it to a computer wirelessly.

Magic finger is a motion detecting device that allows users to have touch capabilities on many surfaces (Yang et al., 2012). The device consists of an optical camera, like those found in computer mice, for sensing motion parallel to the camera plane, and an RGB camera that can identify textures. A similar idea is used in this project for sensing input from the 3D-printed components, but simpler sensors are used.

Chapter 3: Component Designs

The following chapter discusses the different component designs. A component is either a button, slider, or knob that is 3D-printed with exception of the slider and knob potentiometers, which are used as is. (They are sensors that do not require a component. See chapter 4.) We have constructed each kind of component with different sensors. In this chapter we discuss the design and construction of each component/sensor combination and then evaluate their durability, haptics and print orientation. Table 3.3 at the end of the chapter summarizes the features of the component designs.

3.1 Evaluation Criteria

Durability is measured by the strength of the component. For buttons, strength means the approximate number of times they can be pushed down without breaking. For sliders, strength is defined by the number of times the slider can slide back and forth. For knobs, strength is defined by the number of times the knob can be rotated. Each component was printed and used 100 times. Very durable (good durability) means that it did not break, fairly durable (fair durability) means that it broke in 50 uses or more, and not very durable (bad durability) means it broke in less than 50 uses.

Haptics is a description of how the component feels to a human user. For example, pleasing feedback to a human user in terms of the button is that it is clear that the button has been pushed down. Otherwise, the button could be pushed until it breaks and/or it could be unclear to the user when it is in the down state. For the slider and knob, it is measured by whether or not the motion is smooth.

Print orientation refers to the way components are printed. The 3D printer used in this thesis lays down material one horizontal layer at a time. A component can be oriented at any angle relative to the layer plane. That orientation is the print orientation.

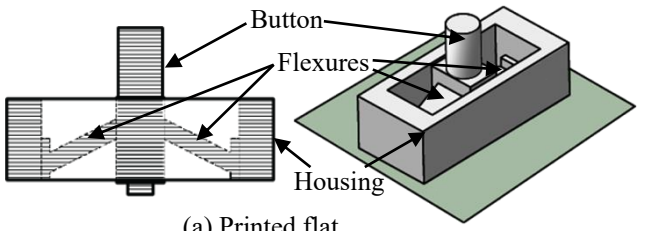
Print orientation can affect the strength and functionality of some of the components. We illustrate this with the compliant mechanism button design. Print orientation is determined by the printer used. We used a Stratasys Dimension Elite printer, which is a fused deposition modeling (FDM) printer that prints in ABS plastic. FDM essentially means that the printer builds models layer by layer. Using an FDM printer can limit the possible print orientations of a component if the component breaks easily when the layers are not placed in a certain way.

Figure 3.1 shows the compliant mechanism button printed in three different orientations. Figure 3.1a shows it printed flat. The first image is a cross section of the button, and the lines show how the 3D printer layers the plastic. The second image shows how the button was oriented on the print bed. This orientation does not work well for the compliant mechanism button. The flexures are very thin, so the layers just separate when the button is pushed.

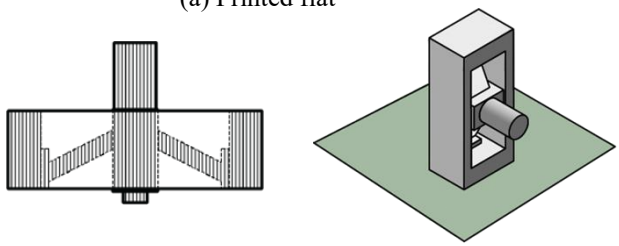
Figure 3.1b shows the button printed on its side widthwise. Printing the button in this orientation does not work because the layers are parallel to the direction that the button is pushed down. There is also only a thin layer holding each flexure to the button and to the housing. When the button is pressed, the layers will separate from each other and the button will break.

Figure 3.1c shows the button printed on its side lengthwise. This is the required orientation for the compliant mechanism button. The layers follow the angle of the flexures and continue into the button and the housing, creating a strong connection between them. As the flexures bow, the layers hold together, making the button much stronger.

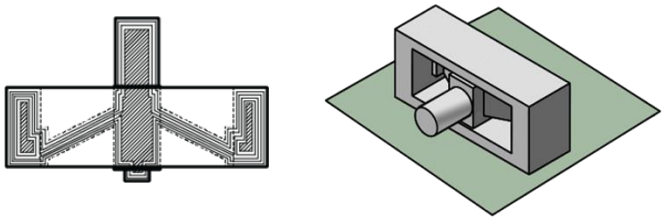
These criteria measure how effective a component is in terms of printing it and using it as a pleasing human interface. Durability is important because if a component were to break on its first few uses, it would take longer to test out a device as a result of spending time printing and reprinting components every time they break. Haptic feedback is important to give designers a better idea of how the device might feel before it gets to its final iterations. Print orientation is important because it shows the ease of use of each component. Printing a component right into a prototype requires less assembly and is thus easier than printing it separately and creating a way to mount it into a prototype.



(a) Printed flat



(b) Printed on side widthwise



(c) Printed on side lengthwise

Figure 3.1 Print Orientation

3.2 Buttons

Five different button designs are discussed and evaluated here. The designs differ based on how they work mechanically. The first three designs are completely 3D-printed and are

designed to move and provide haptic feedback without any assembly. The idea was to design a button that worked by printing it right on a prototype, but each one relies on print orientation so the motion, durability, and feedback are not consistent. As a result, we made two other designs that can be printed in any orientation which require assembly using other parts, such as springs, to allow motion and haptic feedback. These can be printed right into a prototype in any orientation.

3.2.1 Cantilever Button

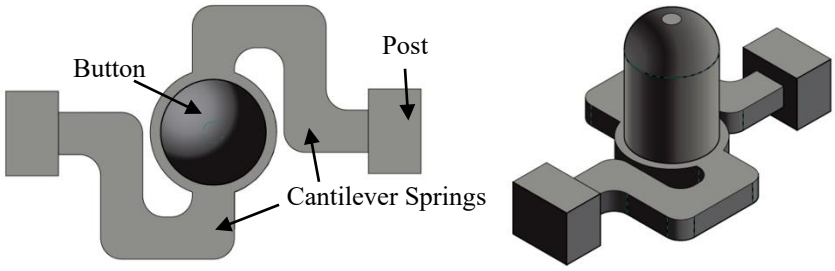


Figure 3.2 Cantilever Button

The cantilever button (Figure 3.2) contains two cantilever springs. The springs are connected to the button. Each spring also connects to its own post. The posts are used to attach the button to a prototype. When the button is pressed, the springs flex and allow the button to bow downward. The springs move the button back to its original state when released.

The button is fairly durable. It breaks after about 100 clicks by reasonable finger force of around 1.20 to 1.40 N or if a single force of about 17.35 N or more is used to push it. The button starts to sag from its original resting state after about 50 clicks. It has little haptic feedback. In other words, it is difficult to feel whether or not it is in the down state. As a result, it is possible to push the button too far and snap the springs. Print orientation matters. The springs are best

printed flat, orthogonal to the print head. That way the layers are also orthogonal to the direction of motion, strengthening the springs.

3.2.2 Compliant Mechanism Button

The compliant mechanism button (Figure 3.3) is described by equations that came from BYU's Compliant Mechanisms Lab[1].

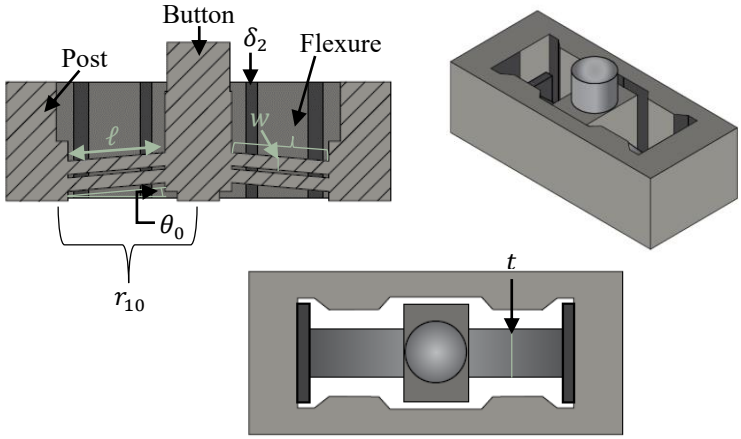


Figure 3.3: Compliant Mechanism Button

There are six constants (see Figure 3.3 for reference): The width of the flexures (w) is the width of a single flexure. The length of the flexures (ℓ) is the length of a flexure from the post to the button. The variable t is the out-of-plane thickness. The modulus of elasticity (E) is a constant that changes depending on the material that is used. The initial angle (θ_0) is the starting angle of the flexures before any force is applied to move the button. The length (r_{10}) is the horizontal width from the middle of the post to the middle of the button. The independent variable is δ_2, which is the displacement or the amount the button has moved down based on the force applied to it.

[1] Adapted from the first lab of ECCE 550/ME550 Micro-Electro-Mechanical Systems (MEMS), Fall 2012

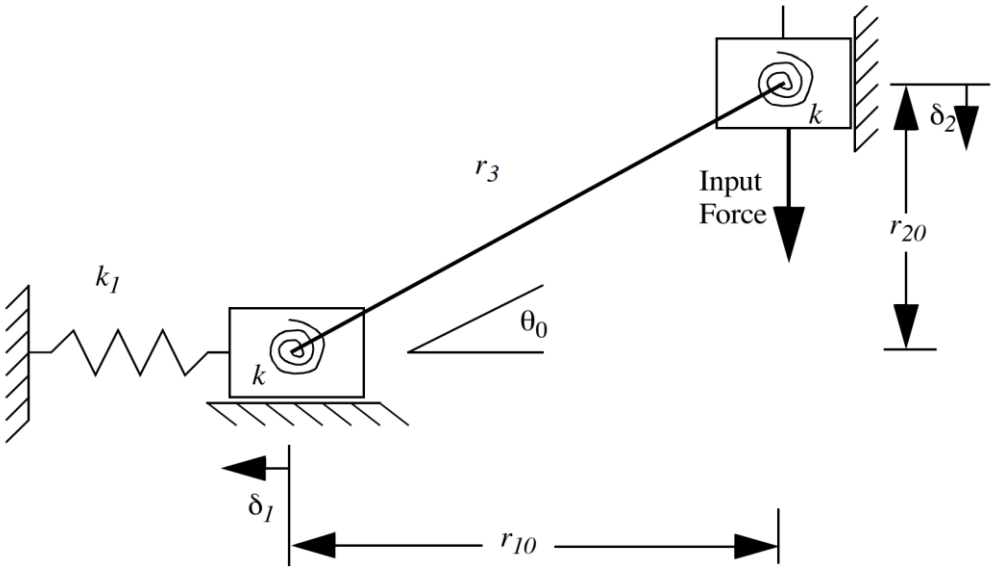


Figure 3.4: PRBM for one half of compliant mechanism[2]

The dependent parameters are calculated by the following equations (Figure 3.4):

$$k = \frac{tw^3E}{12\ell}$$

$$k_1 = \frac{Etw}{2\ell}$$

k is the stress constant for the button, and k_1 is the stress constant for the post.

$$r_3 = \frac{r_{10}}{\cos(\theta_0)}$$

r_3 is the length of the flexure plus the length of half of the post and half of the button.

$$r_{20} = r_{10}\tan(\theta_0)$$

r_{20} is the vertical length starting from the center of the button down to the center of the post.

$$r_2 = r_{20} - \delta_2$$

r_2 is the current vertical position of the button.

[2] Image taken from first lab of ECCE 550/ME550 Micro-Electro-Mechanical Systems (MEMS), Fall 2012

$$\theta = \sin^{-1}\left(\frac{r_2}{r_3}\right)$$

θ is the current angle of the flexure.

$$\delta_1 = r_3 \cos\theta - r_{10}$$

δ_1 is the horizontal displacement of the post.

$$\psi = \theta_0 - \theta$$

ψ is the angle that the flexure rotates from the initial angle θ_0.

The above parameters are used to calculate the approximate downward force required to push the button down to the given displacement δ_2. The force is calculated as follows:

$$F = \frac{k_1 r_2 (r_3 \cos(\theta) - r_{10})}{\sqrt{r_3^2 - r_2^2}} + \frac{2k\psi}{\sqrt{r_3^2 - r_2^2}}$$

Table 3.1 Units for Compliant Mechanism Model

Constants	Units
Width of flexures (w)	μm
Length of flexures (ℓ)	μm
out-of-plane thickness (t)	μm
modulus of elasticity (E)	GPa
initial angle (θ_0)	$radians$
length (r_{10})	μm

Table 3.1 shows common units that are used. When using these parameters, the calculated force (F) is in μN.

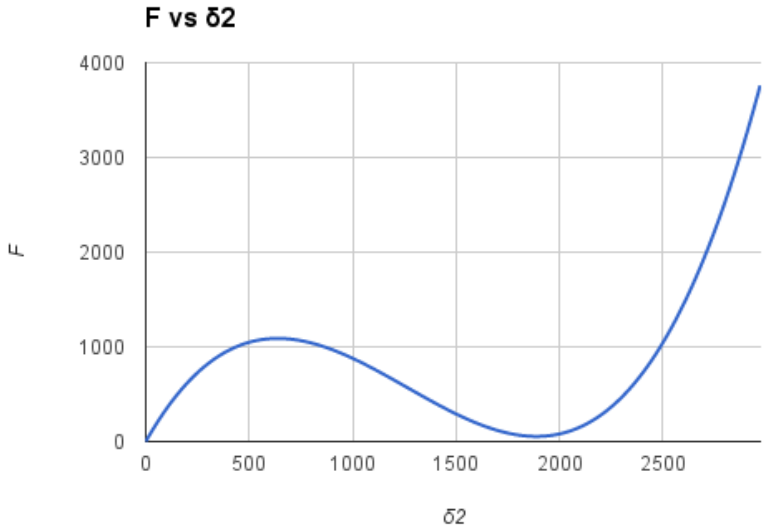


Figure 3.5: Monostable Model

Figure 3.5 shows the relationship between F and δ_2. The button moves to its down position when a force of just above 1000 (μN) is applied. A force graph like this shows that the button is stable in only one state. That means, when it is released, it will return to its up state. Had the force graph crossed the origin, it would mean that the button has 2 stable states (bi-stable). This means that the button would stay in the button down state until pushed back to the button up state. The parameters used to generate Figure 3.5 are as follows in Table 3.2:

Table 3.2 Example Parameters

Constants	Value
Width of flexures (w)	$800\ \mu m$
Length of flexures (ℓ)	$6000\ \mu m$
out-of-plane thickness (t)	$3000\ \mu m$
modulus of elasticity (E)	$2.09\ GPa$
initial angle (θ_0)	$0.09\ radians$
dimension (r_{10})	$14000\ \mu m$

The modulus of elasticity was taken from MatWeb[3] using the average value for the Acrylonitrile Butadiene Styrene (ABS), Extruded. The material was a good match for the material we used with the Dimension Elite printer.

The compliant mechanism button is very durable since it does not break with reasonable finger force of around 1.20 to 1.40 N after 100 clicks. It breaks when a single force of about 30.25 N is applied to it. When the button is first printed there is little haptic feedback besides some resistance when the button is in the down state. After being pushed about 100 times, the button starts forming a slight popping feel, which gives haptic feedback that is a pleasing to human interaction when the button is pushed down. The button print orientation is on its side, with the flexures being built up in the z direction for it to work (as shown in Figure 3.1c).

3.2.3 Pop Cap Button

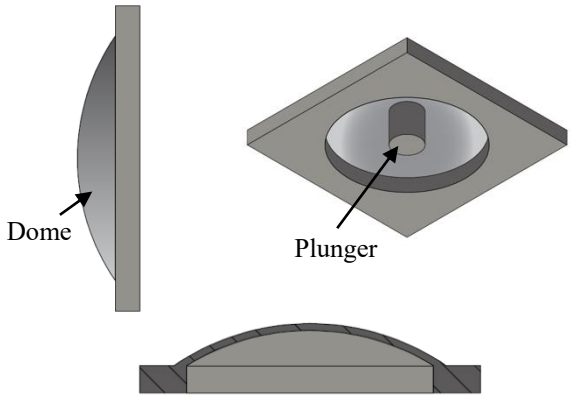


Figure 3.6 Pop Cap Button

The pop cap button (Figure 3.6) is a thin dome that when pressed, gives under the force. When released, the dome returns to its original state.

The button is very durable as it does not break with reasonable finger force of around 1.20 to 1.40 N after 100 clicks. It can break if a force of about 80.95 N or more is exerted on it.

[3] http://www.matweb.com

Cracks can appear after clicking more than 100 times. The button still works, but the plunger can also fall off of it if clicked enough. The plunger is attached on the underside of the dome in the center. When the dome is pressed enough, the plunger can start to shear off because of the strain caused from the dome deforming. The button gives haptic feedback that is pleasing to human interaction when pressed, so it is easy to tell when the button is in the down state. The best print orientation is when it is printed flat. If it is printed on its side, the layers separate from each other when the button is pressed.

3.2.4 Spring Button

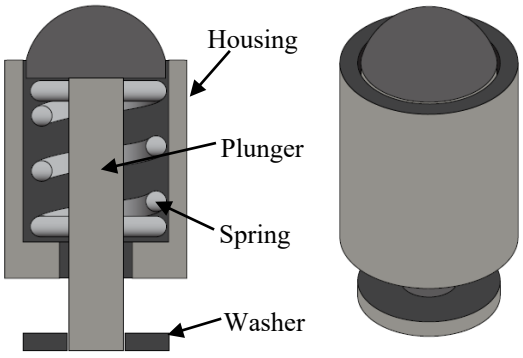


Figure 3.7 Spring Button

The spring button (Figure 3.7) contains a housing, a plunger, and a washer. These pieces are printed separately. A metal spring is then inserted into the housing, and the plunger goes through the center of the spring and through a hole in the bottom of the housing. The washer is attached to the plunger and keeps the plunger from being pushed out of the housing by the spring when the button is released.

The button is durable because it does not noticeably break down after 100 or more clicks. It has haptic feedback that is pleasing to human interaction because the button down state can be felt when the spring compresses fully. Print orientation does not matter for this button.

3.2.5 Magnetic button

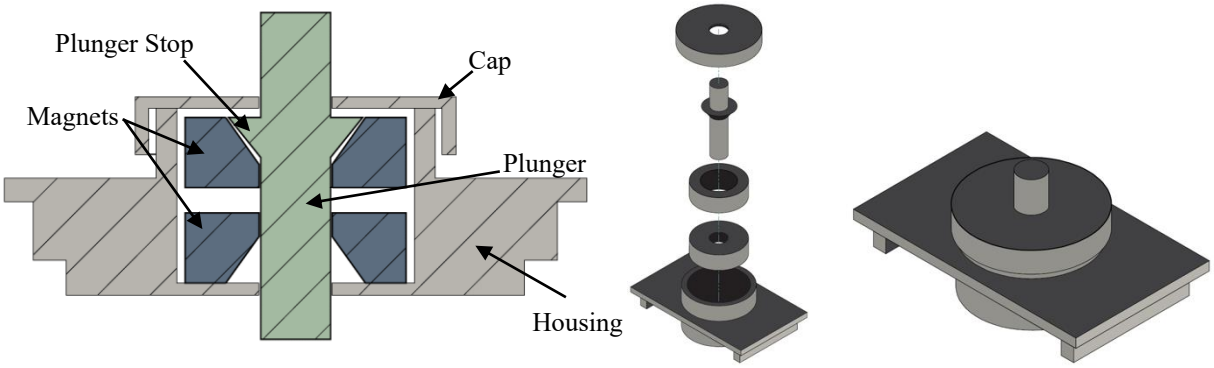


Figure 3.8 Magnetic Button

The magnetic button (Figure 3.8) is similar to the spring button. It contains a housing, a plunger, and a cap instead of a washer. Two magnets with donut holes are placed inside the housing. The plunger is inserted down the donut holes of the magnets and through a hole in the bottom of the housing. The cap is then placed around the plunger and snapped on to keep everything together. The plunger has a stop to keep it from coming out of the housing.

The repulsive force of the magnets is used as the mechanism for the button. As the button is pressed, the magnets move closer to each other and their repulsive force grows stronger. Upon release, the magnets push the button back up to its original state.

The button does not noticeably break down after 100 clicks and so is very durable. It has haptic feedback that is pleasing to human interaction from the force of the repulsion increasing as the button is pressed down. Print orientation does not matter.

3.3 Slider

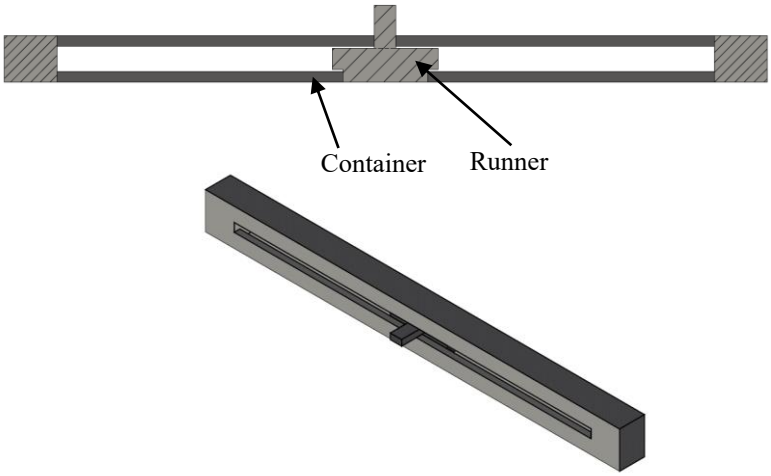


Figure 3.9 Base Slider

The basic 3D-printed slider (Figure 3.9) consists of a container and a runner that slides inside of it. The runner is printed at resolution distance from the container so that it does not fuse. Resolution distance in this case is the minimum distance in which two objects can be printed without being fused into one object. For example, the Dimension Elite prints at 0.178 mm resolution, so the runner is placed at 0.178 mm distance on all sides of the container. The slider is printed in place to reduce assembly time.

The slider is very durable as it can be slid back and forth at least 100 times without anything breaking. In terms of haptics, having to place the runner at resolution distance from the container causes the slider motion to be a little sloppy. There is a little up and down play inside the container. The runner also moves without being pushed if the slider is angled so that the runner moves at a downhill angle. Each print is slightly different and sometimes the runner does not run very smoothly in the container. A simple fix to make the slider feel smoother and stay in place is to use Super Lube® Synthetic Grease with Syncolon® (PTFE) Multi-Purpose Lubricant.

However, the up and down play is not fixed by the grease, and may be best fixed through printing the slider in separate pieces or using a higher resolution printer. This would allow for tighter tolerances which would reduce the up and down play. Printing in separate pieces enables tighter tolerances because the pieces can be printed closer (within the resolution of the printer) to the correct size without fusing.

The base design for the slider varies a little depending on the sensor used (refer to Chapter 4). The print orientation of the slider depends on the sensor used. The container and runner design has been printed standing and sideways, and both orientations printed without any fusing or other issues.

3.4 Knob

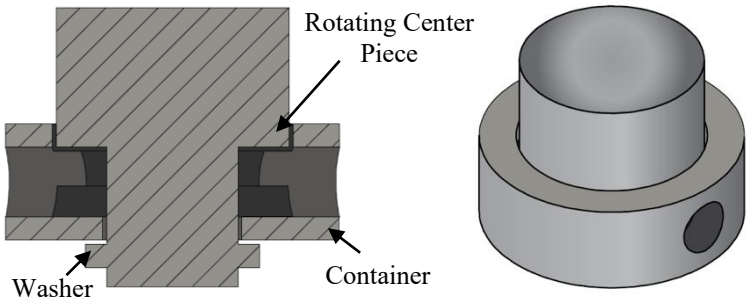


Figure 3.10 Base Knob

The 3D-printed knob (Figure 3.10) contains a rotating center piece inside of a container. There is a washer (also printed in place) below the container of the knob that attaches to the rotating center piece and keeps it from falling out of the container. The distance between the container and center piece is also done at resolution to keep them from fusing. The knob is printed in place to reduce assembly time.

The knob is very durable as it does not break after about 100 rotations. Haptics are similar to the slider. There is some slop because of the resolution of the printer. There is a little

up and down play as well as side to side play. The knob rotates well inside the container, but feels smoother when lubricant is added. The up and down and side to side play may be best fixed through using a higher resolution printer or through printing the pieces separately. This would allow for tighter tolerances which would reduce the up and down and side to side play. Printing in separate pieces enables tighter tolerances because the pieces can be printed closer (within the resolution of the printer) to the correct size without fusing.

Its print orientation also depends on the sensor being used. The rotational center piece and container design has been printer flat (the washer is flush with the print bed) and on its side and both printed without any fusing or other issues.

3.5 Summary Table

Table 3.3 Features of Component Designs

Component Type	Durability	Haptics	Print Orientation Matters
Cantilever Button	Fair	None	Yes
Compliant Mechanism Button	Good	Good	Yes
Pop Cap Button	Good	Good	Yes
Spring Button	Good	Good	No
Magnetic Button	Good	Good	No
Slider	Good	Okay, Smoother with grease	Depends on sensor used, but base model no
Knob	Good	Okay, Smoother with grease	Depends on sensor used, but base model no

Chapter 4: Widgets

We explain and evaluate each of the sensors used to make widgets. The sensor types are mechanical sensors, IR photo reflectors, and IR photo interrupters. We then discuss the widgets built using these sensor types. The sensors are evaluated based on their interaction region, sensitivity, and noise. Interaction region is defined as the space around a sensor in which it is possible to create changes in the sensor's value by creating motion through that space. This includes the action or direction needed to create the change in the sensor value. Sensitivity is the amount of motion required from a component to interact with a sensor. Noise is the fluctuations in the data that come when a component interacts with a sensor. The source of noise associated with the sensor itself is not distinguished and is instead assumed to be noise associated with the component interacting with the sensor in this evaluation.

Evaluating the sensors with these three criteria helps in understanding how each sensor can be used to develop a widget. Evaluating the interaction region enables us to understand the direction a component must move and the action it must take to trigger a sensor. Sensitivity informs the amount of motion a component needs to create for the sensor to detect change. Noise shows how loose tolerances in components may affect the reliability of what the sensor collects.

Each component has a general design that was discussed in Chapter 3. The button designs remain the same for all of the sensors except the plunger changes a little depending on the sensor it is interacting with. The slider and knob designs are modified for the different sensors.

The widgets are evaluated based on assembly and feasibility of widget placement. Assembly is defined as the necessary number of steps to assemble a functioning widget by combining its component and sensor and placing it on a prototype. More assembly can complicate the process of making a prototype and require more time. Feasibility of widget placement is defined as the area on the surface of a prototype on which it is possible to place a

specific widget. This shows how restrictive a widget is when designing a prototype. One other evaluation criteria, print orientation, is used for the sliders and knobs as in Chapter 3. The criteria could not be used until this chapter because the designs are dependent on the sensor.

4.1 Mechanical Sensors as Widgets

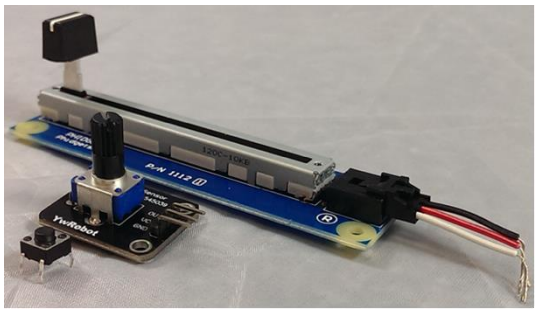


Figure 4.1 Front to back: Push button switch, Rotary potentiometer, Slide Potentiometer

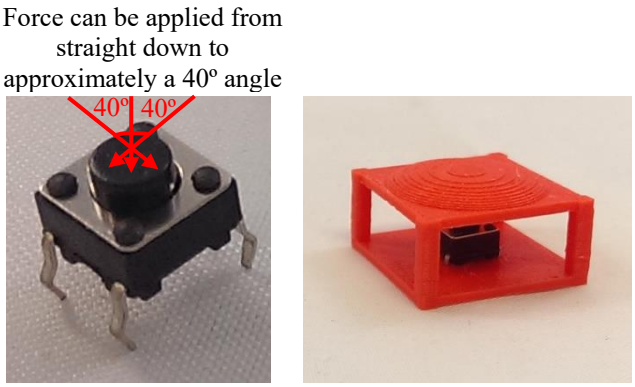
Mechanical sensors (Figure 4.1) are used for a comparison to the other two sensors (IR photo reflectors and IR photo interrupters) as the functionality for a button, slider, and knob already come built into the sensors. The term mechanical is used to signify that the sensors contain their own moving parts to generate changes in their readings.

The mechanical sensor used for buttons is commonly known as the push button switch. The push button switch either completes or breaks a circuit when pressed. The sensors we use contain a little piece of domed copper. When the button is pressed, the dome collapses from the force of the press and makes contact with the pins on either side of the button, completing the circuit. The dome reforms when the button is released, once again breaking the circuit.

Potentiometers can function as either a slider or a knob. The basic idea behind a potentiometer is that it creates fractional changes in resistance. These fractional changes are caused as a contact moves along a resistive strip. As the contact moves along the resistive strip,

resistance is either increased or decreased. Potentiometers can be found and purchased in either linear (slider) or rotary (knob) form.

4.1.1 Buttons



(a) Push Button Switch

(b) Push Button Switch with Pop

Figure 4.2 Button Interaction

The push button switch is rigid in terms of its interaction region. A force between straight down and approximately a 40º angle (Figure 4.2a) from vertical must be applied to the sensor in order for the button to go down. The button is sensitive in that as soon as it is pushed, the signal will change immediately. There is no noticeable noise from the senor when a component interacts with it. We did not observe any noise associated with bouncing, so no measures were taken to debounce the signal. It has two states high and low, and only changes when pressed and released.

Figure 4.2b shows the push button switch mounted below a pop cap button. To assemble the widget, the push button switch must be mounted below the button. It then needs to be wired up so that it can be attached to a processor that can detect its signal changes. The component is limited to being placed right above the sensor in order to interact with it. So the feasible widget

placement would be anywhere on the surface of a prototype where the component and sensor fit in a similar configuration as seen in Figure 4.2b.

4.1.2 Slider and Knob

Figure 4.3 Slide Potentiometer Widget

The component for the slide and knob potentiometers is a hole through the surface of the prototype and any mounting geometry if necessary. The slide and knob potentiometers that we used have handles that protrude far enough from the prototype surface that they can work for a human to interface with the device, but handles can be printed and attached if necessary or desired. The Potentiometers can be mounted so that the handle sticks through the surface of a prototype. Interaction occurs directly with the user as there is no component in between.

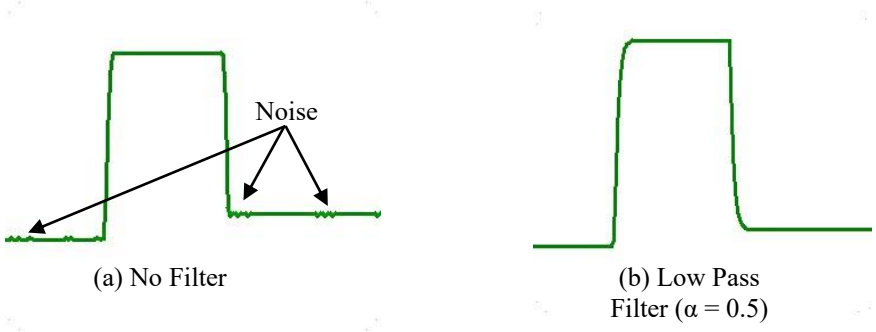


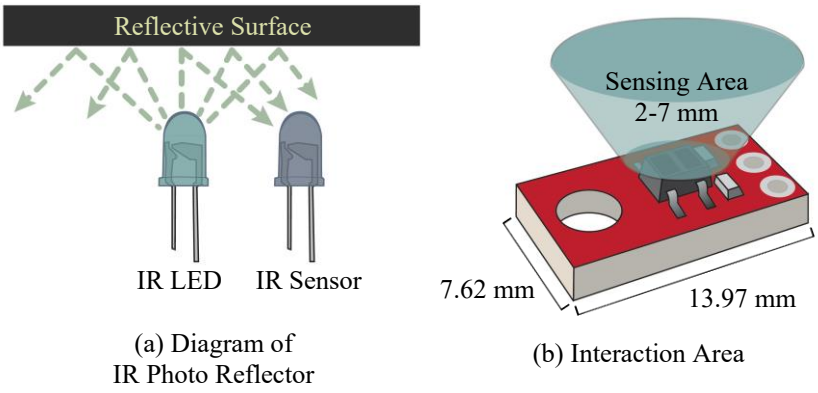
Figure 4.4 Slide Potentiometer Trace

The Interaction region for the slider is just a side-to-side force that moves the slider back and forth. The knob requires a rotational force. Potentiometers are sensitive. They can detect small movements. There is a slight bit of noise from the potentiometers, but it can easily be

filtered out using a low pass filter. Figure 4.4 shows two similar graphs of the signal coming from a slide potentiometer. Figure 4.4a shows the signal before a low pass filter is applied. There is not a lot of noise, but the signal has a tendency to attenuate a little. Figure 4.4b shows the signal after a low pass filter is applied. It smooths out the small attenuations. The print orientation does not apply in this case.

Assembly for both the slide and rotary potentiometer is just a matter of mounting it through the surface of a prototype (Figure 4.3) and attaching wires to it. The feasible widget placement is dependent on the size of the potentiometer. Their size and shape is dependent on what is available on the market.

4.2 IR Photo Reflectors as Widgets



(a) Diagram of IR Photo Reflector

(b) Interaction Area

Figure 4.5 IR Photo Reflector

IR photo reflectors (Figure 4.5a), also known as proximity or line sensors, can detect varying levels of reflected IR light. Generally, they are configured so that an IR LED and an IR sensor (phototransistor) are placed side-by-side, facing the same direction. When an IR reflective surface, such as paper, is placed in front of the sensor, greater degrees of IR light are reflected

back to the sensor as the surface gets closer. Depending on the implementation, this can either cause fractional increases or decreases in voltage.

The IR photo reflector has the largest range of interaction compared to the other sensors used. Figure 4.5b shows a blue cone that represents the interaction area of the sensor, which is approximately 30º angling away from one of the edges of the sensor. Imagine that the height of that cone is about 5 mm. A widget can interact with the sensor at about any angle above the sensor, and at about a 30º angle off of one of the edges. The optimal sensing distance is at about 3 mm, but the sensor can sense objects that are about 5 mm away as well as objects almost touching the sensor. The sensing distance depends on how reflective the material is. A more reflective material gives a wider range of values as well as can be detected from a slightly further distance (out to about 7mm).

Figure 4.6 Reflectance Test Device

The Dimension Elite has 9 materials. A cube was printed in each material and placed about 3 mm away from an IR photo reflector (Figure 4.6). The initial value of the sensor, without anything in front of it was about 1018. Higher values mean less reflectance. Table 4.1 shows the approximate reflective value for each color used with the Dimension Elite.

Table 4.1 Color Reflectance Values

Color	Approximate Value
White	76
Yellow	112
Red	242
Orange	295
Blue	500
Ivory	522
Grey	916
Green	934
Black	987

White is the most reflective surface and black is the least reflective. It is important that the surface is reflective when using an IR photo reflector. Otherwise, the sensor is not able to detect movement from a widget. Depending on the distance from the sensor, as well as the color, the levels of reflectance change. As a result, whenever this sensor is used, a registration step is required. Each widget is registered differently, but generally it means calibrating the widget to the sensor so that the widget functions properly.

The sensor is very sensitive to any change in light. Out of all of the sensors used, the IR photo reflector has the most noise as a result of difficulty controlling light levels. A low pass filter and time average can be used to help filter the signal.

4.2.1 Buttons

Figure 4.7 IR Photo Reflector with Compliant Mechanism Button

All that is required to get the IR photo reflector to interact with a button is to place it under the plunger (Figure 4.7). The button widgets are not complex to print and assemble, but do require an extra step in software. They have to be registered in order to detect button up and down states properly. Buttons are registered by first detecting the reflectance in the button up state, then in the button down state. That way the computer listening to the sensor can send an event when the button changes state.

Button widgets that use the IR photo reflector have the largest range of feasible widget placement. The placement is limited on the size of the component as well as the size of the sensor, but the sensor can be placed anywhere around a button component as long as it is about 3-5mm (7mm with a more reflective color) and can detect changes in light level as the button is pushed down. This allows for more freedom in where to place a button widget on the surface of a prototype.

4.2.2 Slider

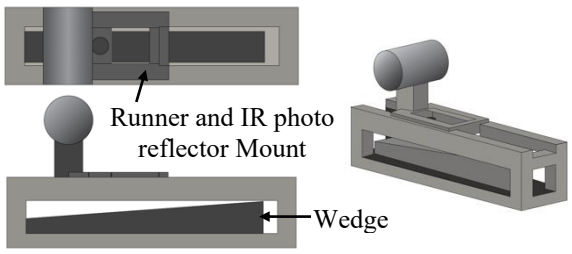


Figure 4.8 Slider with IR Photo Reflector

The sensor is placed on the runner of the slider facing a wedge. As the runner slides over the wedge, the amount of light the sensor detects increases or decreases as the wedge slopes closer or farther away from the sensor (Figure 4.8). The reflectance value coming from the sensor is used as the position of the slider.

A registration step is needed to know the minimum and maximum values of the slider. The minimum and maximum are used to normalize the values. This allows for the freedom to resize and reshape the slider to fit as desired into a prototype.

The assembly for this slider includes mounting the IR photo reflector, attaching the wires, and mounting the widget on the surface of a prototype. The slider can be printed directly into the prototype, but print orientation does matter. The runner is a lot weaker and buckles when the slider is printed with the runner facing sideways instead of up.

The size and shape of this slider can change as long as the wedge causes a steady increase/decrease in reflectance values. The feasible widget placement of the slider is limited by its size and the wires. The size can change giving designers more freedom in how they can place the widget on their prototype. The wires require enough room to move with the sensor so that they do not get snagged as the sensor is slides.

4.2.3 Knob

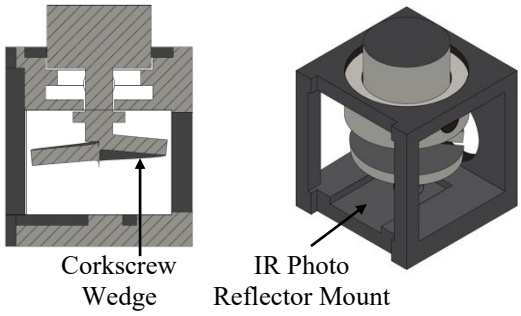
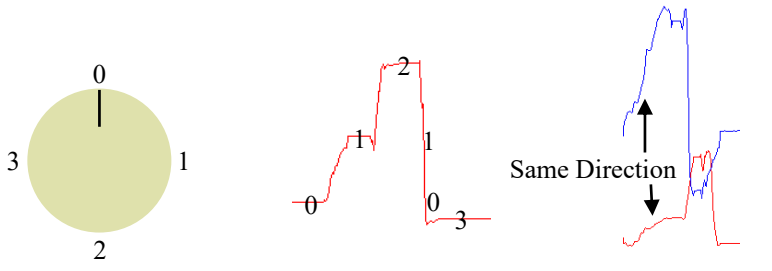


Figure 4.9 Knob with IR Photo Reflector

The sensor is placed below a corkscrew shaped wedge. As the knob rotates, the wedge changes slope, changing the amount of light reflected back to the sensor. There is a sharp change after the wedge rotates one full rotation. The sharp change signals that the knob is still rotating in

the same direction and that the knobs values continue to increase or decrease depending the direction (Figure 4.9).

The above process does not work very well because the corkscrew does not cause a sharp jump in the values of the sensor unless the user turns it fast enough. Since this design does not control the speed at which a user turns the knob, the sensor can read intermediate values. This is likely a result of the sensor being caught halfway (more or less) between the highest and lowest points in the corkscrew. It is difficult to know what the values will be at that stage with the result being the errant detection of multiple changes in direction.



(a) Knob States (b) Trace One Sensor (c) Trace Two Sensors

Figure 4.10 Knob States and Traces

In an attempt to clean up the data, a registration step was created for the knob. The registration consists of gathering *n* number of discrete states. The knob is marked with a line indicating where it is pointing and when it makes a full 360º revolution. For example, in one experiment 4 states were gathered (Figure 4.10a). State 0 was marked as the first value when the knob marker points up. State 1 was marked where the knob marker points right. State 2 was marked where the knob points down and state 3 where the knob points left. The light reflectance values were gathered at each of these states.

As the knob rotates, the distance of the value coming from the senor to the 4 states collected was calculated. The state that the value was closest to would be the current state of the

knob. If the state change was positive, then the knob increased. If it was negative, then it decreased. The only exception was if the knob did a full revolution. In this case the change was counted as positive if the state went from 3 to 0 and negative if it went from 0 to 3. It became even clearer that intermediate values were coming in during this jump (Figure 4.10b). Unless the knob was turned very quickly over the point where it goes from the lowest to highest state, intermediate values would pop up. In the case of the knob used to generate data for Figure 4.10, this point was between 2 and 3. So, instead of going from 2 to 3, the states (2, 1, 0) or a subset of them would come through. This made it difficult to know when the knob was continuing in the same direction, or reversing direction. It was also possible to stop at one of those intermediate values even though the state at that point should be either 2 or 3.

A second sensor was added in an attempt to be more consistent in detecting the direction that the knob was turning. The sensors were originally placed 180º apart. The knob would only be counted as increasing values or decreasing values if the reflectance from the two signals was changing in the same direction. This stabilized the knob, but unfortunately there was only a small window in which both sensor signals were changing in the same direction. As a result, the sensors were placed about 90º apart (Figure 4.10c) in an attempt to increase the amount of surface over the knob's corkscrew in which they change in the same direction. Doing so did increase the time in which both of the sensor's signals changed at the same time, but not enough to make the knob very responsive. This widget in its current state gives unreliable results as it is not responsive and is a little jumpy from the noise caused by the sloppiness from the print tolerances.

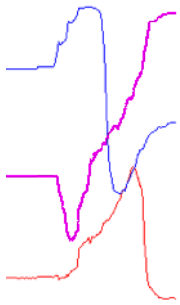


Figure 4.11 Three Sensors Trace

Finally, adding a third sensor gave reliable results. It was placed about 130-135 degrees offset from the first two sensors. Placement was driven by where it would fit. In this case, discrete knob states were no longer used. They helped in understanding the data, but are not necessary for detecting the direction that the knob is turning. Instead, the raw trace values were used as seen in Figure 4.11. When the knob is turned, the difference between the current read value and the previous read value of each sensor is calculated. The sign of each of the differences is checked and if the difference is greater than some arbitrary threshold, then a counter associated with the sign of that difference is incremented.

Once that is done for the three sensors, the larger absolute value of the two counters is selected. If it is greater than 1, it means that at least two of the sensors agree on the direction that the knob is turning. The result is the direction of knob rotation is consistently detected. Figure 4.11 shows the knob as it is rotated clockwise. There are always two sensor traces consistently changing in the upward direction. The limiting resolution of the 3D printer allowed movement of the knob both vertically and horizontally, which could cause false readings. Increasing the arbitrary threshold reduced the number of false readings, but made the knob less responsive.

The assembly for the knob widget requires that the sensor be mounted in the component, and that wires be attached to it. The widget can then be mounted or printed onto the surface of a

prototype. The feasible widget placement for the knob is constrained by the size of the knob and corkscrew as well as the size of the sensor. The sensor should be placed to detect the linear change in reflectance as the corkscrew rotates. With that requirement met, the knob widget can be placed wherever it can fit on the surface of a prototype. The diameter of the knob can change, but the diameter of the corkscrew that interacts with the sensor must be large enough to change the amount of light being reflected back to the sensor as it is turned.

4.3 IR Photo Interrupters as Widgets

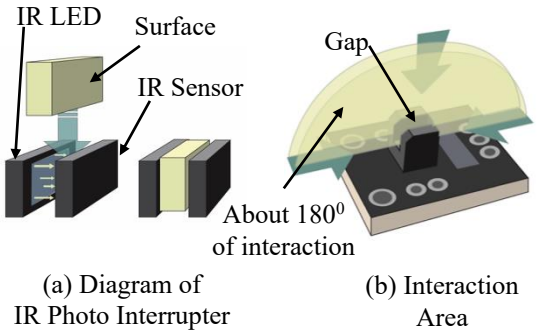


Figure 4.12 IR Photo Interrupter

IR photo interrupters (Figure 4.12a) contain an IR LED and an IR sensor that face each other across a gap. When a surface is inserted in between the gap, the light is blocked and the sensor detects the change. This can be used as a binary switch.

In terms of the interaction region, IR photo interrupters in general require more precision than the mechanical sensors and the IR photo reflectors when interacting with components. The part of the component that interacts with the sensor has to cross through the correct part of the gap to block out the IR light. Even though more precision is required, there are more ways to interact with it than the mechanical sensors. For example, a button plunger can move into the gap at any angle, as long as its plunger blocks the light in the button down state. This allows for

about 180º of interaction around the gap (Figure 4.12b). The sensor is very sensitive and changes state as soon as any IR light is blocked or detected. There is no noticeable noise when a component interacts with the sensor. The signal changes immediately when the IR light is blocked or unblocked. It is possible for the sensor to detect IR light from nearby components of nearby widgets when not desired if the sensors are not covered well enough to block out ambient IR light, in the case of the gray code slider (section 4.2.2) and knob (section 4.2.3).

4.3.1 Buttons

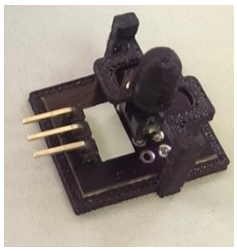


Figure 4.13 IR Photo Interrupter with Cantilever Button

All of the button designs work with the IR photo interrupter. Assembly is simple as the sensor is placed below the button so that the plunger can interact with it (Figure 4.13). The sensor also has to have wires soldered to it so that it can interact with a processor. The placement of the sensor is a little more complex depending on the width of the gap even though there are more ways to interact with it. The smaller the gap is, the more precise the movement from the button has to be to slide into the gap when the button is pressed down. The sensor allows a little more freedom for widget placement than the push button switch because the sensor could be mounted at different angles and still interact with its component.

The cantilever button is the most challenging design to use with this sensor, since the button gives no feedback when it goes to the button down state. As a result, it is easy to snap the button by pushing it too far. This can be fixed by mounting the sensor closer to the button so that

the plunger runs into the sensor when pushed down. Another challenge is that the button cantilever sags after about 100 clicks. The button can sag into the gap and always block the light, causing the button to always register in the down state.

4.3.2 Gray Code Slider

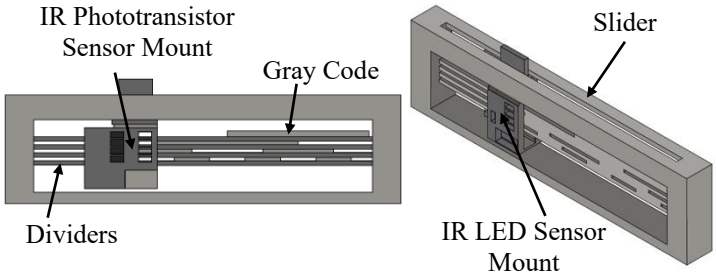
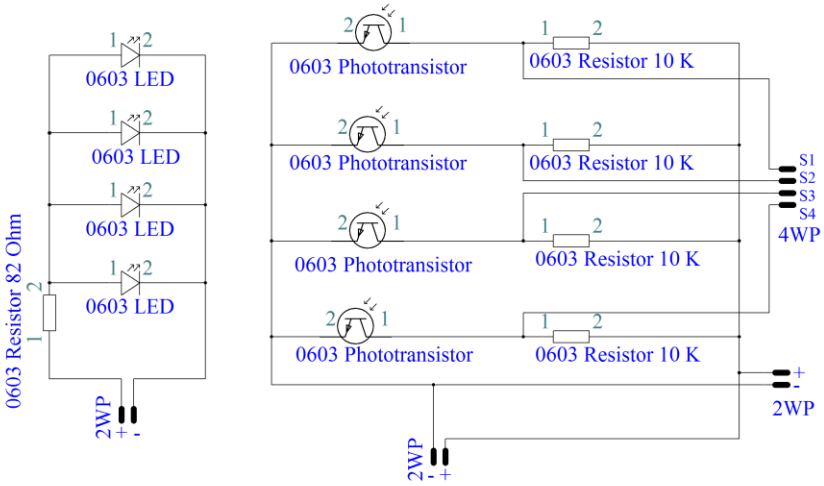


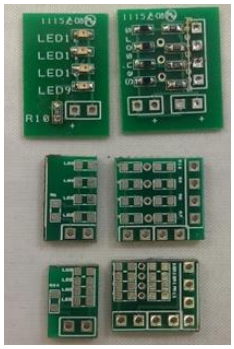
Figure 4.14 Gray Code Slider

The gray code slider uses a slightly different sensor than the buttons, but uses the same idea behind the IR photo interrupter. Figure 4.14 shows two different views of the slider. The side view shows the IR phototransistor sensor mount and that the LEDs and phototransistors line up across the gray code. The perspective views show the IR LED sensor mount.

A custom PCB was made (Figure 4.15) that contains 4 IR LEDs and another that contains 4 IR phototransistors. Figure 4.15a shows the schematic for the PCBs. The left schematic shows the logic for the IR LED board, which is used to illuminate the sensor board. The right-hand schematic details the logic for the IR phototransistor board, which is used to read the light output from the IR LED board. The sensors that detect IR light from the LED board is determined by a 3D-printed component (continue reading this section and 4.2.3 for more detail). Figure 4.15b shows three different sizes of the board. The LEDs, phototransistors and resistors are all 0603 (0.6 mm x 0.3 mm) surface mounts. The two PCBs are placed across a gap facing each other.



(a) Schematic



(b) Custom gray code sensors (LEDs on left, phototransistors on right), top was first design, bottom two, second smaller designs

Figure 4.15 Custom Gray Code PCBs

Gray codes are used to create unique patterns that differ consecutively by one bit. Since there are 4 sensors and each has a high and low state, a total of 2^4 or 16 unique patterns can be generated. These patterns can be laid out linearly, creating a code that can be used as a slider.

Before printing the gray code slider, a 3D modelling program called OpenSCAD[4] was used. OpenSCAD is a functional language that specializes in generating 3D models that can be printed. The phototransistors on the custom PCBs register as high or 1 when no light is shining on the sensor, and low or 0, when light is shining on the sensor. This means that there has to be geometry to block light from the sensor where the code is 1 and holes where the code is 0.

The gray codes were generated on OpenSCAD recursively. Then, those codes were used to build the geometric model for the slider by placing a rectangle wherever there was a 1. The combination of rectangles was extruded to create a solid. Figure 4.16 illustrates how the

[4] http://www.openscad.org

geometry for the code was built. In this case, each bit was represented by a 3mm x 3mm square. Notice that the geometry is placed where there are 1's.

0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0

Figure 4.16 4-Bit Gray Code Sequence

The material used to build a gray code slider must not be very reflective of IR light. Otherwise, IR light can be reflected around the slider and give false values. IR light being emitted from the LEDs will not be blocked from the printed representation of the gray code. Instead, it will reflect off of the code and other parts of the slider and could potentially find its way to one of the sensors that should be blocked from receiving light, causing the sensor to give a false reading. Black worked the best out of the material that came out of the Dimension because it is the least reflective, though green and gray also worked. The brighter colors, such as white or yellow, shown in Table 4.1 did not work well because they do not block enough IR light, which causes false readings.

To further control the amount of light being sent to the phototransistors, dividers and sensor mounts were used. Figure 4.14 shows the dividers. They are placed in between each row of the gray code sequence. The dividers are there to help keep IR light from bleeding over into sensors that should be blocked. The sensor mounts were designed to block the edges around the LEDs and phototransistors. In the case of the LEDs, this blocks out light being emitted sideways. For the phototransistors, the mounts block out light coming into the sides of the sensor, restricting it to detecting IR light that is shined directly in front of it.

Assembly for the gray code slider is challenging. There are 4 signal wires and 2 wires for power and ground. The LED PCB is lined up and powered by headers coming across the gap from the phototransistor PCB.

The widget's placement is restricted by its size and the wires coming off of the PCBs. The height of the slider can be changed by the height of the sensor. The width of the slider is affected by the width of the sensor. Making the sensor length shorter works, but eventually the sensor starts to miss or skip over codes if the gray code width is too small. The wires come off of the IR phototransistor PCB, which means there has to be room for them to slide along as the sensor moves across the code. The slider then has to be mounted on the prototype using glue or some other method.

The slider runner is printed in place as part of the slider component and it works when printed standing or on its side for the larger PCBs. The smallest PCB (last PCB in Figure 4.15) requires that the slider be printed standing up. Otherwise, the holes for the mounts do not print out properly when using the Dimension Elite. The smaller the PCB, the more the sloppiness of the print affects the readings from the sensor. The reason is that any small movements up and down in the runner have the potential to offset the gray code PCBs just enough so that they are no longer lined up with the gray code. The larger PCBs have more space between the phototransistors, and therefore a little movement up and down does not affect the sensor readings.

4.3.3 Gray Code Knob

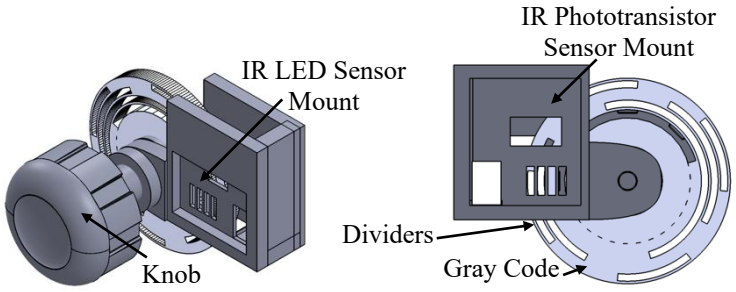


Figure 4.17 Gray Code Knob

The gray code knob uses the same sensor and pattern as the gray code slider, but the pattern is laid out in a circle. Figure 4.17 shows the knob from different perspectives. The knob turns the gray code wheel and the sensors are static. The dividers are there to divide each ring of the gray code (4 in this case since there are 4 channels or phototransistors). They help keep light from bleeding over to neighboring phototransistors when two or more holes are next to each other. The sensor mounts encase the LEDs and phototransistors. Similar to the gray code slider, this reduces the amount of emitted IR light from the LEDs. They also reduce the amount of IR light detected by the phototransistors.

OpenSCAD was used as well, except arcs were used to build the gray code geometry instead of rectangles. Figure 4.18 illustrates how the geometry was built using the codes. The middle black circle is not part of the gray code, it is a mounting bracket for the knob. The height of the arc is defined by the size of the LEDs and phototransistors as well as the space between each of them. The arc angle is defined by $360^{o}/4^{2}$ (4 being the number of channels). This will give 16 arcs around each ring with the correct height to block the phototransistors. Each ring refers to the series of arcs that make up the code for each channel or IR phototransistor. There are 4 rings in this case. Arcs are only added where the code is 1.

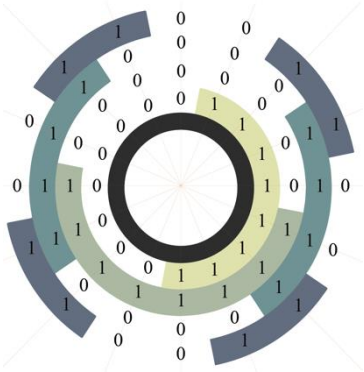


Figure 4.18 4-Bit Circular Gray Code

The same limitations as the gray code slider apply for the colors that can be used. Black, green and gray were the least reflective, and should be used so that IR light is blocked. Reflective material can cause erroneous results as the dividers and sensor mounts no longer block IR light.

Since the gray code knob uses the same sensor as the gray code slider, the same assembly is required to get the IR LED and IR phototransistors up and running. The knob is printed separately, so it has to be glued or mounted in some other way on a prototype.

The feasible widget placement for the knob is dependent on its size. The largest part of the knob is the gray code wheel. The size of this is dependent on the height of the sensor as well as the space between each of the LEDs and phototransistors. The height of each ring is the same height as the LEDs and phototransistors plus the space between them. The rings cannot change sizes, otherwise the code would not be lined up with the sensors. The mounting ring can change diameters. The widget can be mounted anywhere on the surface of a prototype as long as the gray code wheel fits.

4.4 Modular Widgets

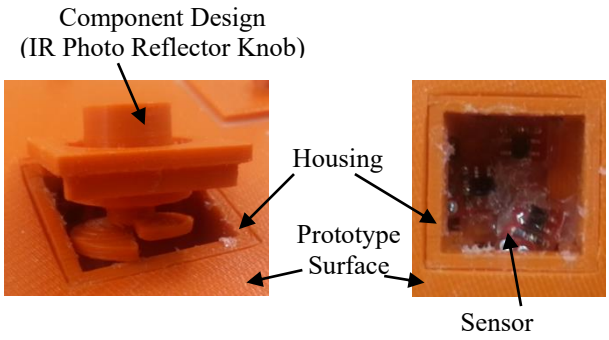


Figure 4.19 Modular Widget

Modular widgets (Figure 4.19) were created to make it easier to change out the component part of a widget. There are two parts to the component of a modular widget. It has one of the designs described in Chapter 3 and a housing. The housing is attached to the surface of a prototype and the sensor or sensors are mounted onto it. The design clips into the housing and is positioned so that it can interact with the sensor.

The cantilever button and magnetic button with the push button switch and the IR photo reflector knob were converted into modular widgets. The buttons are interchangeable, meaning that they both use the same housing and interact with their sensor the same way. The IR photo reflector knob has a different housing from the buttons.

The modular widgets are useful because a designer can quickly change out components. For example, if one of the cantilever buttons break, it can be replaced by popping the broken one out of its housing and snapping a newly printed one in place. No re-gluing or re-mounting of sensors is required. The trade-off is that the widgets are boxier and take up more room on the prototype.

4.5 Grid Array

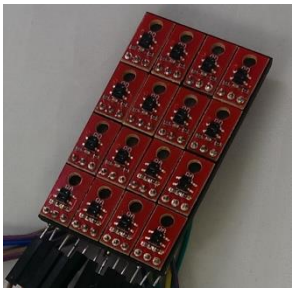


Figure 4.20 Grid Array of Proximity Sensors

The grid array consists of 16 IR photo reflectors. Each sensor is placed side by side in a 4x4 grid. In this case, sensors are no longer attached directly to a component. Instead, all of the components interact with the grid. A distinction for the purposes of this paper is that the definition of widget is somewhat different when using the grid. The grid is decoupled from the components rather than integrated into them, but functionally we still consider it a widget. Each component has to be registered, so that the computer listening to the sensors can differentiate between them. The interaction, sensitivity, and noise follow the same evaluation as was described in section 4.3. The only difference is that interaction is a little more complicated as there could be interference from other sensors. In this case, we did not worry about interference and are relying on the sensors being spaced far enough apart as well as their short 5-7 mm sense range to eliminate any interference that they may cause between each other.

Assembly is about the same for each of the widgets in this case. They can be printed in place if they are oriented right or if print orientation does not matter as in the case of the spring button. They can also be printed separately and attached to the prototype. The assembly for the hardware is just a matter of mounting the grid inside the prototype. The components have to be modified slightly to ensure that they move within the sensing range (inside 5-7 mm) of the IR

photo reflectors. For example, the button plunger has to be long enough to reach the grid when the button is pressed.

4.5.1 Button

The button is registered by collecting a button up and a button down state. A difference is found between these two states to find the sensor or sensors that the button interacts with. The value halfway between the button up and button down state is used to trigger a button down event, when that value is crossed. The button can be placed anywhere above or around the sides of the grid. As long as its plunger is moving over one or more sensors, the grid can sense its motion.

4.5.2 Slider

The Slider is registered by collecting three states. The slider runner is moved to the left most position, the center position, and the right most position. The difference between the left most and center state, and the difference between the center state and right most state is taken to find the sensors that the slider interacts with. If any of the differences are above an arbitrary threshold, then it is assumed that the slider interacts with the sensors corresponding to those differences.

A registered slider includes a low, middle, and high value. These values are used to interpolate the value of the slider widget. The value of the slider widget is found by taking the current state of the sensor values corresponding to the slider and calculating the Euclidean distance between that state and the left most ($distLeft$), center ($distCenter$), and right most ($distRight$) states. If $distLeft$ is less than $distRight$, a weighted average is found by

$$\frac{distCenter \times lowValue + distLeft \times middleValue}{distLeft + distCenter}$$

The sensor value is then set by rounding the result of the weighted average. If $distRight$ is less than $distLeft$ then the weighted average is found by

$$\frac{distRight \times middleValue + distCenter \times highValue}{distRight + distCenter}$$

The sensor value is then set to the rounded value of this weighted average.

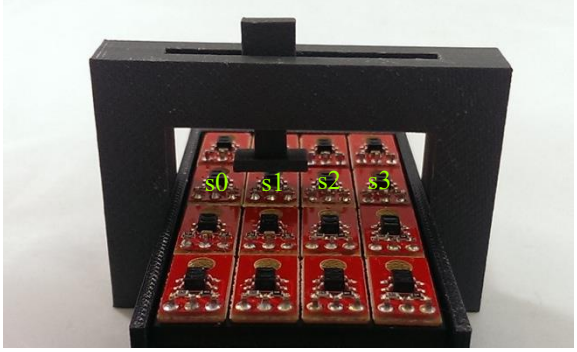


Figure 4.21 Grid Slider Example

For example (refer to Figure 4.21), suppose the low value is 0, middle value is 5, and high value is 11, and the sensor interacts with four sensors on the grid. The left most state is {s0: 500, s1: 800, s2: 900, s3: 1000} (s stands for sensor). The center state is {s0: 1000, s1: 600, s2: 600, s3: 1000}. The right most state is {s0: 1000, s1: 900, s2: 800, s3: 500}. Say the current state of the slider is {s0: 800, s1: 500, s2: 800, s3: 1000}. The Euclidean distances for each state are $distLeft$ = 435.89, $distCenter$ = 300, $distRight$ = 670.82. The $distLeft$ is less than the $distRight$ so the weighted average is found by

$$\frac{distCenter \times lowValue + distLeft \times middleValue}{distLeft + distCenter}$$

$$\frac{300 \times 0 + 435.89 \times 5}{435.89 + 300} = 2.96$$

The weighted average is rounded and so the sensor value is set to 3.

The low, middle, and high values can be changed, but the wider the range the more jumpy the values tend to be. Using a smaller range is stable, but the wider ranges are more prone to the noise coming from the sensor because slight changes in the reflectance values can change the interpolated value of the slider. The above method is not very stable when the slider is close to the middle state regardless of the size of the range. Using more sensors may help because there would be more sensor values that could be used for interpolation.

The feasible widget placement of the slider is either above or to the side of the grid. It needs to interact with (slide across) more than one sensor. The part that interacts with the sensors can be a piece of geometry that moves with the slider that is close enough to change the light levels of the sensors as it moves across them.

4.5.3 Knob

Similar to the IR photo reflector knob, a corkscrew wedge was used to detect increasing and decreasing light levels. The same problem appeared when reaching the end of the corkscrew where the sensors read intermediate values instead of jumping from the lowest reflectance value to the highest reflectance value or vice versa.

In an attempt to clean up the data, we collected reflectance states for all of the sensors on the grid. Specifically, we collected four different positions on the knob that represented each quarter turn. Those value were fed into a neural network. About 50 data samples were collected for every sensor at each quarter turn. The neural net seemed to do fine classifying the positions until the end of the corkscrew wedge was reached. Intermediate values were detected. For example, assuming the knob is being turned clockwise, 3,0 would be the desired sequence, However, going from state 3 to 0 might cause the sequence 3, 2, 1, 0 or 3, 1, 0, both indicating a counter-clockwise rotation, depending on how fast the knob is turned.

The only feasible knob widget placement is above the sensors. The knob can be slightly angled as long as the corkscrew wedge is close enough to the sensors to change the amount of IR light reflected back to them.

Chapter 5: Prototypes

Three prototypes were made to demonstrate the use of the widgets. A ski pole handle MP3 player controller, etch-a-sketch controller, and 3D mouse were made to demonstrate how the widgets can be used as building blocks to make an interactive prototype. We also built an array of sensors to prototype a generic "field of interaction" that could be placed below an object surface and used to explore different widget arrangements.

Each of these prototypes is evaluated by their functionality, assembly, and widget placement. Functionality is defined as whether or not the prototype functioned as a whole. It helps in understanding if the widgets work and can be used together. Assembly is an overview of the steps needed to make the entire prototype, which demonstrates how difficult it is to use the widgets. The placement of their widgets is a discussion on the size and limitations of each widget in regards to the prototype.

A brief discussion about the software architecture is presented in this section. The software was used to test the widgets and as a platform on which to build applications to test prototypes using the widgets.

5.1 Software Architecture

A Blend micro Arduino microcontroller was used to read sensor values and send them to a computer. The microcontroller wirelessly sends data via Bluetooth. A computer was used to receive the sensor values from the microcontroller via Bluetooth. The computer software was implemented in Java. Figure 5.1 shows a basic overview of software architecture. The architecture was designed to be an event handling system for the widgets and to be easily extendable to support new widget types with different designs and/or sensors.

There was a Bluetooth data collector that parses the data coming from the microcontroller. The data collector handles either digital or analog values. Each widget type had a base abstract class that raised common events for that widget. For example, the button had a button down and button up event. The abstract classes were extended to create differing functionalities depending on which sensor was used. For instance, the slider had an implementation for the potentiometer slider, the gray code slider, and the IR photo reflector slider.

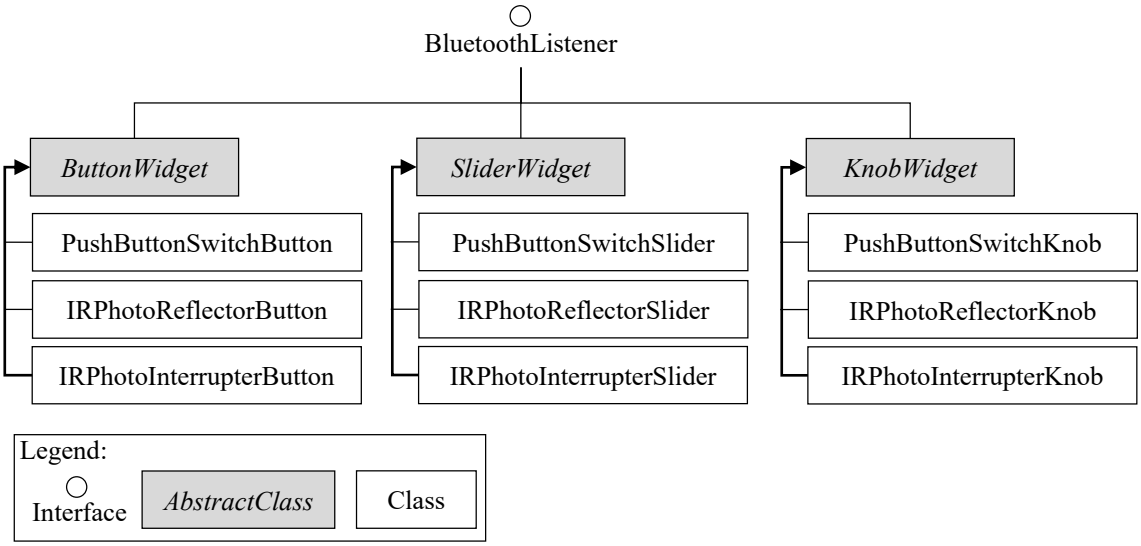


Figure 5.1 Overview of Software Architecture

Any data filtering, like low pass filters, happened in the individual widgets. The widgets were responsible for storing any data that they needed. For example, the slider with the IR photo reflector stored its previous filtered value. When it received a new value, it used its previous filtered value to filter the new value using a low pass filter.

The observer pattern was used to create events. The basic idea behind the observer pattern is that there is an object that contains a list of dependent objects or listeners. When that objects state changes, it notifies the listeners often by calling one of their methods. Every time a

widget is instantiated, it is registered with the Bluetooth data collector as a listener. The data collector triggers an event with a digital widget only if the signal has changed from low to high or high to low. The analog widgets receive events every time the data collector receives new data because analog data tends to be fairly variable. To raise an event on a listener, the data collector calls the handleBluetoothEvent method for either digital or analog signals.

Widgets contained a list of listeners called controllers. A controller implemented the events that are common to a widget. For example, a knob had three events, initialKnobPosition, valueDecreased, and valueIncreased. When creating a new prototype application, a programmer can implement the controller and tell it what to do when one of those events is raised.

5.2 Ski Pole Handle

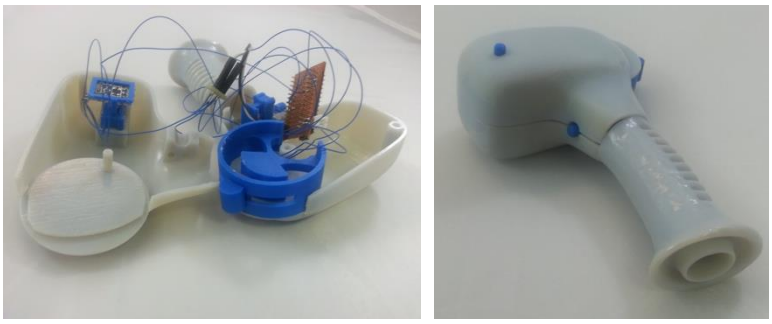


Figure 5.2 Ski Pole Handle

The ski pole handle MP3 player controller controls an MP3 player over Bluetooth. It is wireless and powered by a lithium ion battery. It has three button functions: play/pause, skip forward, skip backward, and one slider function: change volume.

Figure 5.2 shows a prototype of the ski pole handle that contains two compliant mechanism button widgets and a slider widget. One compliant mechanism button widget uses an IR photo interrupter and the other uses an IR photo reflector. Since there are three button functions, one of the button widgets has the skip forward function tied to a single click and the

skip backward function tied to a double click. The other button widget's function is play/pause. The slider widget uses an IR photo reflector and is used to change volume.

In this case, all of our controls functioned as designed. The assembly required gluing the sensors to their corresponding components. The IR photo reflector button widget required a registration step to detect the correct button down state. The IR photo reflector slider widget required a registration step so that the software could detect the left-most and right-most positions of the slider runner.

The compliant mechanism button widgets have a small amount of space to move around along the surface of the prototype as long as they do not interfere with each other or the slider widget. The slider widget is curved to follow the contour of the back of the ski pole handle. It works well in its position, but the size and shape of the widget would likely have to change for it to fit anywhere else on the surface of the prototype.

The ski pole handle is a perfect example of a PhysiComp. We designed three other prototypes of the ski pole handle with widgets before this example. The first one was designed just for functionality. The second was designed more for comfort. The third design utilized our gray code slider. We learned that although the gray code slider provided adequate volume control, it only provided 16 discrete positions. Our last prototype replaced the gray code slider with an IR photo reflector slider widget, which gave a smooth continuum of volume control. Moving the placement of widgets or using different widgets to fit a new form turned out to be straightforward, illustrating the effectiveness of 3D printing when building an interactive PhysiComp.

5.3 Etch-a-Sketch Controller

(b) Etch-a-Sketch Prototype (b) Etch-a-Sketch

Figure 5.3 Etch-a-Sketch Controller

The etch-a-sketch controller (Figure 5.3) controls an etch-a-sketch program on the computer wirelessly. An etch-a-sketch is used to control a pointer that draws lines as it moves. The cable is used for power, but a battery can also be used. This controller uses sliders instead of knobs to draw as an interesting departure from the original etch-a-sketch design. It contains a gray code knob to change line thickness. The button in the middle allows the user to change line color. It also changes the LED color corresponding to the color of the line.

The prototype functions as it is supposed to. The gray code knob was a little difficult to get working because the wheel is a little loose allowing it to move side to side. The smallest gray code PCBs were used for this prototype and the small amount of play was enough to move the sensor off of the coded part of the wheel. Mounting it to the prototype made it more stable, but every once in a while the wheel became offset enough to the point where the gray code changed incorrectly.

Assembly involved adding a push button switch to a cantilever button, one IR photo reflector to each slider, and an IR LED/Sensor PCB to the gray code knob. The sliders and cantilever button were printed directly into the housing. The gray code knob was attached using hot glue. An RGB LED was also attached using hot glue.

The microcontroller, wires, battery, and widgets use up most of the room in this prototype. The wires for the slider kept getting snagged on other wires and widgets making it difficult to move. We had to change to solid wires because the braided wires we originally used kept getting ripped off of the slider when they would get snagged.

The problem with the gray code knob could be resolved by using a larger wheel and a PCB where there are larger gaps in between each sensor. The larger gaps would allow the arcs that make up the gray code wheel to be larger. This would guard against the side-to-side motion of the wheel caused by the limited resolution of the printer because more motion would be required to offset the wheel enough to incorrectly change the code. Another fix would be to print on a higher resolution printer or print the component in parts for a tighter fitting wheel.

The problem with the slider widgets containing the IR photo reflector might be remedied by placing conductive strips above or below the runner, which contains the sensor. The sensor could have brushes attached to it that made contact with the runners, allowing it to be connected to power, ground, and signal. This way the wires are no longer moving with the sensor. Otherwise, a different widget may have been a better choice, like the slide potentiometer, or the prototype could have been enlarged to make more room for the moving wires.

The etch-a-sketch is an example of a PhysiComp that was designed to be comfortably used as a handheld device. The prototype showed that the 3D printer can be used effectively to print widgets directly into the housing of a prototype. Of course, the button cantilever was oriented the right way in the housing for the print, otherwise it would have needed to be printed separately and attached. The slider length and width could be changed as needed (as long as the width does not go below the width of the IR photo reflector) so that the device feels comfortable to the user.

5.4 3D Mouse

(a) 3D Mouse Prototype (b) .obj Viewer Application

Figure 5.4 3D Mouse

The 3D mouse (Figure 5.4) controls a custom .obj viewer application. An .obj file is a 3D geometry file format. The program loads the 3D object defined by the file. It allows the user to move the object/camera using the 3D mouse prototype to see the object in different views. The mouse has six buttons. The white buttons are all magnetic buttons. The other three are cantilever buttons. The buttons control the functions top view, bottom view, left view, right view, change axis of manipulation, and toggle pan/rotate. The middle of the mouse contains an IR photo reflector knob that is used for panning or rotating. The top contains a slide potentiometer that controls zoom.

All of the widgets on the prototype function well. This prototype helped us understand that the IR photo reflector knob needed three sensors to function. It functions as desired, but moving and rotating the 3D model in the mouse application is a little jumpy from noise caused from the component interacting with the sensor. This may affect user experience with the prototype negatively because the knob does not provide motion as smoothly as, for example, the knob potentiometer. The advantage is that the diameter of the IR photo reflector knob can be changed as long as it is large enough for the component to interact with the sensors.

The assembly used modular widgets. The sensors still have to be attached to their corresponding components, but they are attached to the housings that the components fit into. Six

push button switches were attached to housing for six different buttons. The cantilever and magnetic buttons use the same housing and can be interchanged. Three IR photo reflectors were attached in the housing for the knob. The slide potentiometer was not a modular widget. It was just hot glued to the surface of the prototype so that the handle was sticking through the surface.

This prototype has a lot of room to move the widgets around. The modular widgets are a little bigger because they have blocky housings that fill more space than non-modular widgets. The 3D mouse is also a good example of a PhysiComp because the design can be refined around the idea that it should be comfortable to use with one hand. The widgets can be moved, tweaked and reprinted until that comfort is achieved. The modular widgets take it a step further by allowing cantilever button and magnetic button components to be interchanged, thereby changing the feel of the widget without requiring the whole prototype to be reprinted.

5.5 Grid Array Prototype

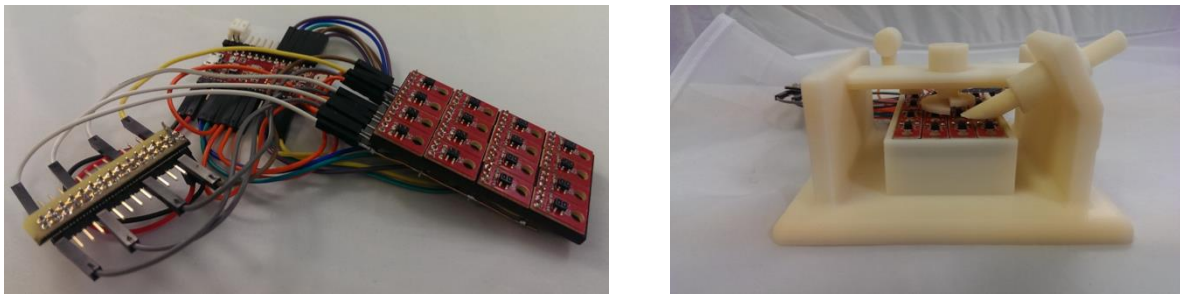


Figure 5.5 Grid Array Prototype

A simple prototype (Figure 5.5) was made using the grid array of sensors. It was made as proof of concept to show that it can work, and to reduce the amount of work to integrate sensors and components. One of each widget was printed, a button, slider, and knob, to interact with the grid array. Each widget was registered and their events were tested. The button worked well, the up and down states can be easily detected.

The slider was programmed to linearly interpolate between values 0 and 11 inclusive. It did well with every value except when it was near the middle. It seemed to jump between values 6, 7, and 8. The knob did not worked properly (refer back to section 4.4.3 for a discussion of why it did not work).

The only assembly required for this prototype was adding a spring to the button and placing the grid array into its mount. The widgets can be moved around to interact anywhere with the grid. The slider requires more than one sensor to function properly.

The grid has great potential to work as a sensor array for one or more components. It could greatly simplify assembly and prototyping, because a different set of widgets could be used with the grid without remounting sensors. In the work completed thus far, the array only worked effectively with one component, the button. It is possible that more sensors on the array would give more accurate and stable readings when using the slider and knob.

Chapter 6: Conclusion and Future Work

3D printing technology is an effective way to prototype PhysiComps. 3D printing enables the creation of custom widgets that are durable, printable in any orientation and create interaction that is pleasing to humans. The widgets support rapid prototyping of PhysiComps because they can be moved around and reprinted on the surface of a prototype to test different configurations.

3D printing enables the creation of custom widgets that are durable. We found that all of our widgets were surprisingly durable. Every one of the widgets met our minimum definition of durable (100 clicks), and all but the cantilever button exceeded that metric. This demonstrates the durability of the designs and material we used in this research.

3D printing enables the creation of custom widgets that are printable in any orientation, whether printed in place as part of a prototype or printed separately and attached. We found that the cantilever button, compliant mechanism button, pop cap button, gray code slider, gray code knob, and IR photo reflector slider widgets are affected by print orientation. As a result, we have to pay attention to their orientation to make the widgets useable in a prototype where they have been printed in place. However, they can still be used in any orientation by printing them separately and attaching them to the prototype post print.

3D printing enables the creation of custom widgets that create interaction that is pleasing to humans. Most of the button widgets provide good haptic feedback and function as expected. The cantilever button widget is limited to good haptic feedback only when used with the push button switch. The slider and knob potentiometers widgets provide the desired feel and behavior. The IR photo reflector and gray code slider and knob widgets provide reasonable haptic feedback when using synthetic grease. However, the IR photo reflector knob widget is an area for further development to refine its feel.

Three of the prototypes we created, the ski pole handle controller, the etch-a-sketch controller, and the 3D mouse illustrate the potential for how effectively 3D printing technology can be used to quickly and efficiently design and test prototypes. With further work and refinement, this could become a powerful tool to build commercially viable products.

The widgets created in this project are a stepping stone to building a more fully integrated toolkit for designers. The widgets are simple to integrate into prototypes and can be efficiently moved around or changed until the preferred design is found. Further development in automation tools will allow designers to more quickly integrate these widgets into their prototypes.

6.1 Future Work

There are many things yet to be done. We did not explore methods for making sure that a widget can be feasibly implemented in a housing. For example, a designer might create a PhysiComp shape only to find out that the shape cannot include the right widgets in the right places.

The knobs that interact with the IR photo reflector should be explored more to find a more consistent, stable solution. Possibly a different component design would interact better with the sensors, making it easier to detect the direction that the knob is moving. Using three IR photo reflectors works with the corkscrew design. Making a smaller sensor that incorporates the three sensors would enable the knob to be smaller.

The grid array of IR photo reflector sensors needs to be explored and understood better. There is potential to make the array smaller, or about the same size with more sensors, using a custom PCB. More stable techniques both in hardware configuration and software interface could be explored to make the grid more reliable.

The next steps in the research are to extend the widget library and to automate the addition of widgets to a prototype. Extending the widget library would include creating new component designs, making the designs more customizable through parameterization, and using different sensors to create new methods of interaction.

Automation would allow designers to iterate more quickly because they would not have to move widgets around by hand. One piece of automation would be to make the process of 3D scanning a model more streamlined. Currently, scanned 3D models require some clean up to fix the geometry so that they can be printed. This usually requires multiple steps and sometimes different programs. Creating a set of tools to automate this step would speed up the process.

The second piece of automation would be automatic addition of widgets to a prototype. A designer should be able to mark either on their physical or digital model where they want to put widgets and the type of widget they want. A tool would understand those markings and replace the location of the prototype on the surface of the prototype with the corresponding widget.

References

Akaoka, E., Ginn, T., Vertegaal, R., 2010. Displayobjects: Prototyping functional physical interfaces on 3d styrofoam, paper or cardboard models. In: Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction. ACM, pp. 49–56.

Avrahami, D., Hudson, S. E., 2002. Forming interactivity: A tool for rapid prototyping of physical interactive products. In: Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques. ACM, pp. 141–146.

Bdeir, A., 2009. Electronics as material: Littlebits. In: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction. ACM, pp. 397–400.

Gellersen, H., Kortuem, G., Schmidt, A., Beigl, M., 2004. Physical prototyping with smart-its. Pervasive Computing, IEEE 3 (3), 74–82.

Greenberg, S., Fitchett, C., 2001. Phidgets: Easy development of physical interfaces through physical widgets. In: Proceedings of the 14th annual ACM symposium on User interface software and technology. ACM, pp. 209–218.

Hartmann, B., Klemmer, S. R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A., Gee, J., 2006. Reflective physical prototyping through integrated design, test, and analysis. In: Proceedings of the 19th annual ACM symposium on User interface software and technology. ACM, pp. 299–308.

Hudson, S. E., Mankoff, J., 2006. Rapid construction of functioning physical interfaces from cardboard, thumbtacks, tin foil and masking tape. In: Proceedings of the 19th annual ACM symposium on User interface software and technology. ACM, pp. 289–298.

Lee, J. C., Avrahami, D., Hudson, S. E., Forlizzi, J., Dietz, P. H., Leigh, D., 2004. The calder toolkit: Wired and wireless components for rapidly prototyping interactive devices. In: Proceedings of the 5th conference on Designing interactive systems: Processes, practices, methods, and techniques. ACM, pp. 167–175.

Martin, F., Mikhak, B., Silverman, B., 2000. Metacricket: A designer's kit for making computational devices. IBM Systems Journal 39 (3.4), 795–815.

Savage, V., Chang, C., Hartmann, B., 2013. Sauron: Embedded single-camera sensing of printed physical user interfaces. In: Proceedings of the 26th annual ACM symposium on User interface software and technology. ACM, pp. 447–456.

Savage, V., Head, A., Hartmann, B., Goldman, D. B., Mysore, G., Li, W., 2015. Lamello: Passive acoustic sensing for tangible input components. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, pp. 1277–1280.

Savage, V., Zhang, X., Hartmann, B., 2012. Midas: Fabricating custom capacitive touch sensors to prototype interactive objects. In: Proceedings of the 25th annual ACM symposium on User interface software and technology. ACM, pp. 579–588.

Tokuhisa, S., Ishizawa, T., Niwa, Y., Kasuya, K., Ueki, A., Hashimoto, S., Koriyama, K., Inakage, M., 2009. xtel: A development environment to support rapid prototyping of ubiquitous content. In: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction. ACM, pp. 323–330.

Vázquez, M., Brockmeyer, E., Desai, R., Harrison, C., Hudson, S. E., 2015. 3d printing pneumatic device controls with variable activation force capabilities. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, pp. 1295–1304.

Villar, N., Scott, J., Hodges, S., Hammil, K., Miller, C., 2012. .net gadgeteer: A platform for custom devices. In: Pervasive Computing. Springer, pp. 216–233.

Yang, X.-D., Grossman, T., Wigdor, D., Fitzmaurice, G., 2012. Magic finger: Always available input through finger instrumentation. In: Proceedings of the 25th annual ACM symposium on User interface software and technology. ACM, pp. 147–156.