Interactive Prototypes

Use of physical, interactive prototypes in the design process

Mats Hage Eikemo
Department of Product Design
Norwegian University of Science and Technology

ABSTRACT

With the advance in micro controller technology, many solutions have emerged in interactive prototypes. This paper reviews different tools from the last decade with emphasis on how the tools fit into the design process. Special attention is given to which parts of the process the tools target and the fidelity of the prototypes. The findings show that most tools aim themselves at the early stages of prototyping and render a medium fidelity in prototypes. This is due to shortcomings in the prototyping tools as platforms to expand upon, using self-created actuators and sensors. The Arduino diverges from this trend and has prevailed as the go-to tool for interactive prototyping. However, its steep learning curve regarding electrical and computer engineering may be a hinder for some designers.

KEYWORDS: Arduino, Prototyping, Interactive, Physical, Design Process

1. INTRODUCTION

Prototyping has always been an essential part of the design process. From simple paper sketches to foam parts to CAD-modeling and to the recent democratization of 3d-printing, prototypes exist in all stages of the design process, serving different needs and giving designers valuable input. Prototypes give shape to ideas and allow ideas to exist among peers, clients and users.

For high-fidelity interactive prototyping, there are few known tools. Traditional tools are expensive, cumbersome, time consuming, and need a special skill set to master. With a recent expansiveness in micro controllers as well as the rising of mobile computing, there have been made several attempts at giving prototypes a life beyond passive models. By implementing electrical inputs and outputs into the prototype designers can breathe new life into their ideas. Models react to users handling and interaction with them, without the designer intervening or explaining.

The latest pinnacle of these interactive hardware prototypes is the Arduino, developed in Italy especially for design students so that designers and artist can harness the powers of computer programming coupled with electronic tinkering. The ease-of-use combined with vast power has made the Arduino especially attractive and several design studies have implemented them as part of the curriculum.

Arduino has presented itself as the go-to-tool for interactive physical prototyping and it would be interesting to review the literature surrounding the phenomenon. However, there were very few articles about it and none had assessed its strengths and weaknesses as a prototyping tool. To address this other prototyping tools were reviewed alongside Arduino, allowing for comparisons between them.

Specifically I want to review how interactive prototyping tools fit into the design process.
How do they integrate with the process and which part of the design process do they fit best with? When compared, do the tools compete for the same parts of the design process or are they targeted at different stages? Also, are some tools better at certain stages?

This article will first briefly cover the design process, with particular focus on physical prototyping. It will then present different prototyping tools from the last decade. The different tools will be compared and discussed viewed from the design process, giving particular attention to where they fall short and where they excel.

2. PROTOTYPING

Prototyping occurs throughout the entire design process, but different stages of the process need different types of prototypes. To establish a common ground for prototyping as an activity and prototype as a product we will look at how prototypes are typically described.

It is borderline self-explanatory that prototyping is a necessity in product design and used as a tool in the design process. There are several different techniques for prototyping, ranging from simple mockups of foam and presentation models to almost-complete working products. [6] The gap is huge and different prototyping techniques benefit different parts of the design process.

Most prototyping tools can be fit into two main categories, and they reflect the typical design process, swerving between functionality and form. Many prototyping efforts are either specifically directed towards how a product should look or how it should work. [13] Shape, geometries and ergonomics are typically explored though the use of foam and clay models. Functionality and interactivity are explored through simplified structural models using struts.

Designers typically use prototyping to isolate individual problems and use them to answer a simple question. [4] How will this part look? How will this part work?

2.1 What is a prototype

On a fundamental level, Lim et al. present two different aspects of prototypes:

1. Prototypes are for traversing a design space, leading to the creation of meaningful knowledge about the final design as envisioned in the process of design, and
2. Prototypes are purposefully formed manifestations of design ideas."

Even more simplified, Rodgriguez describes the prototype as “nothing other than a single question, embodied.” “In a way quite similar to the scientific method, productive prototyping is about asking a single question at a time, and then constructing a model in the world which brings back evidence to answer your question.” [4]

Prototyping is about asking different questions, one at a time. However, the questions asked need to be directed towards the answers one wishes to get back.

2.2 Categories of prototypes

Solberg and Sæter identified five main categories of physical models within the design process. Their findings were based on a set of literature on the subject coupled with their own experiences and ideas. The five categories are:

1. Mockups, simple build to test feel or size or shape.
2. Shape- and form-models, detailed views of special geometries and ergonomics
3. Functional models, for testing functionality
4. Show models, strictly visual to portray the feel of a model.
5. Prototypes, a functioning product communicating the essence of the final product.

For this paper we will be looking at category 3, the functional models.
Rande proposes dividing prototypes into a fidelity range going from low to high, describing low-fidelity prototypes as prototypes made from simple materials such as cardboard, foam, wooden blocks, office accessories, etc. [2] Implicit, high fidelity prototypes are more complex and complicated, created from electrical and mechanical components as well.

In that light, we are looking at interactive physical prototyping as a high fidelity, functional model.

Yang points out that any prototype is a decision between desired fidelity (appearance, functionality) and available resources (time, effort, cost). Accordingly, the ideal prototypes can be built fast, with cheap components and still give the designer the necessary feedback. [5]

Consequently, an important aspect of interactive physical prototyping is to maintain low resources while still prototyping for high fidelity.

2.3 When to prototype

Prototyping is encouraged throughout every stage of the design process, serving different purposes for each stage. Especially physical prototyping is considered a part of every stage and should be used to support the work done for that stage. [1]

Prototyping should be increased towards the end of the design process, with later stages having the most use of physical prototyping and an increase in fidelity. Low fidelity prototypes are best aimed at the earlier stages of development. [2]

Towards every milestone, prototyping should be used as a productivity tool. By ensuring that prototypes are present at every milestone, more time is spent “doing productive explorations”. [4]

2.4 Why prototype

There are many reasons for why designers should prototype, ranging from abstract reasons such as improved team communication and helping to share knowledge to operational benefits such as improved product quality and reduced development costs. [3]

Lim et. al. [4] have a fundamental principle for prototyping, and that its purpose "[...] is to look at certain qualities without compromising the larger picture."

Furthermore, they "[...] believe that the notion of a “good” prototype can only be understood in relation to the specific purpose of the design process and to the specific issue that a designer is trying to explore, evaluate, or understand." They separate the purposes of prototyping into four non-mutual categories, meaning that prototypes can have several purposes.

1. Evaluation and testing
2. The understanding of user experience, needs, and values
3. Idea generation

These are important overall reasons, but what are the reasons for specifically using physical prototypes? Broek et al [7] interviewed several Dutch design studios and found that "handmade models are still very often used" in the design process.

Because of their tangibility, physical prototypes appeal to human nature by allowing the use of more senses. “People like to see and feel the final product to as great an extent as possible.” [1]

Verlinden and Horváth found that the role of the prototype modeling "[...] enrich the product with information that is difficult to achieve in other ways." Furthermore, "In communication, the hints include design review, create better insight of product and process to other stakeholders." [5]
They explain how design spaces are extremely large and therefore difficult to explore all at once. Prototypes explore smaller parts of the design space, to "predict and evaluate certain effects" [5]

Yang [6] reviews time spent on prototyping, number of components on the prototype, compared with results from a race to see if there is correlation. It is an interesting quantifiable approach to how physical prototyping is used, and how it effects the design outcome.

Her main findings echo the classic mantra of less is more. Specifically fewer parts on the prototype and less time spent on building (as opposed to designing) correlates with a higher rank as well as a higher grade. Also, she finds that meeting a threshold of time and maintaining consistent working hours is important and has higher benefits for the outcome than simply "putting in the required hours". [5]

3. PROTOTYPING TOOLS

Research tells us that physical prototyping is prevalent throughout the entire design process, but skewed towards the end. This especially accounts for functional prototypes that investigate interactivity in the model. Furthermore, higher fidelity models are not usually made until the very end of the design process, when earlier low fidelity models have assessed individual problems [1].

One of the key problems with prototype models is the separation of form and function. Most research shows that models investigate one or the other, but seldom both. Prototypes incorporating both form and function in a single model are mostly found as 0-series prototypes towards the very end of the design process [6].

However, several attempts have been made at solving this with new prototyping tools, and I will present and discuss some of the options:

- Phidgets
- Switcharoo
- Calder Toolkit
- Boxes
- dTools
- Display Objects
- Arduino

This list covers some of the most important tools from the last decade. Many of the tools revolve around a microcontroller, but the implementation differs between them. Also, some non-microcontroller tools were included as a showcase of other possible directions.

As Arduino is the most recent and most used of the tools, it will be given more attention. The others will be discussed in comparison with it.

3.2 Phidgets

Phidgets is an abbreviation for Physical Widgets, created as a physical version of the software widget in computing. Saul Greenberg at the Department of Computer Science, University of Calgary, developed Phidgets in 2001 with the help of Chester Fitchett. [7] It was a response to the difficulties of creating device-dependent applications.

Some of the problems they encountered with existing solutions were:

- The need of knowledge in hobby electronics to be able to use components such as switches sensors and motors.
- Devices may be available, but without a developers API they may be exceedingly difficult to program. Hacking is required.
- On the other hand, devices with API are at the wrong abstraction level.
- The devices are not available to the programmer at the current stage of the project due to limitations such as expense, shipping, etc.

They wanted a standard system for physical devices to be used by designers as low-level tool for creating prototypes. By achieving this many of the custom-built solutions at different practitioners could be avoided and simplified, reducing costs and time constraints. [8]
These experiences were echoed by team they spoke with, who found themselves spending time on selection of electrical components, microprocessor programming, etc. in stead of actually creating a media space, which was the initial design brief. In fact, they ended up relying on the expertise of a visiting electrical engineer to get a working project. The team therefore made an effort to develop a brand new prototyping tools based on widgets from computers. They distilled the requirements down to the following list:

- A connection manager was essential to sense if a device is connected or disconnected and "give the programmer a handle to devices"
- The software and hardware counterparts had to be coupled through identification of some sort. This is especially important if there are many instances of a single type of component (for instance 5 buttons).
- A simulation mode was required so that the programmer could work on the project even if the physical components were not available.

Based on these specs a working model was created. It was based on an USB-controlled microprocessor, rendering the electric components as simple USB devices to the computer. A Phidget is attached to the microprocessor; transmitting what type of phidget it was as well as a unique ID to the computer. Each Phidget was also created as a software-instance, allowing the programmer to access it visually onscreen. The software also allowed simple programming though an API, requiring minimal coding-efforts. As a proof of concept the team created several phidgets including a servo controller, a controllable power bar, a proximity sensor and a 8-input / 8-output device. Using their newly developed Phidgets they re-created art installations. [8]

3.3 Switcharoo

Avrahami and Hudson at Carnegie Mellon University proposed Switcharoo in 2002. [9] It is a low cost prototyping toolset presented as an alternative to expensive, time consuming tools and consists only of a "set of input components and a Macromedia Director interface". The toolkit is a result of observing the shift from product-centered design towards user-centered design at the end of the last century. This shift required a shift in the design process and they identified several key principles that a tool had to work with.

- Rapid prototyping, here meaning prototypes created at a quick rate to explore different aspects of form, ergonomics, etc.
- User testing and iteration should begin as early as possible and repeated throughout
- User data needs to be gathered alongside testing of users
- Clients should be integrated as design-team member

Switcharoo was created to work with foam models and be able to co-exist alongside the exploration of form and space that accompanies these tools. The input components needed to be wireless, so they could accompany the form and not overpower it. This was achieved by using passive RFID-technology, which allows the components to be very small (as they do not require a dedicated power source), and only need a nearby antenna to connect to. The input components were fastened to the foam using three pins protruding from the underside; these three pins secure the component and prevent rotation. The components had to work with Macromedia Director on the computer. It does this by communicating with a local Java server that parses the radio signals to a syntax Director understands and the user can work with.

3.4 Calder Toolkit

Lee et al., at Carnegie Mellon University, developed the Calder Toolkit in 2004. [10] It bases itself on previous efforts, such as Switcharoo (also by Avrahami and Hudson), Phidgets and iStuff, but builds upon them and fuses different elements from each together.
The Calder Toolkit was developed to support designers in the early stages of design process, after the designers have done a series of paper sketching and have moved on to physical prototyping with foam models for prototyping. They identified several key aspects related to these activities that a prototype toolkit would have to fit with:

- Design exploration, driving idea generation
- Communication to others outside the design process such as clients, managers, users
- Design testing and data gathering from users

Specifically the driving factors for the toolkit were "introducing and extending executability and preserving fluidity and flexibility". [10] More practically this meant that designers should not have to learn (another) programming language, but should be able to use the software they already were using to create interactive prototype. Furthermore, the components of the toolkit would have to work together with existing prototyping techniques, such as foam modeling. This required special attention to fastening, size and wiring of the components.

As many other hardware prototyping tools, the Calder Toolkit revolves around a microcontroller and USB host card that connects to and communicates with a computer. I/O-devices are then attached to this hardware board. The Calder Toolkit consisted of a set of wired components that connected to a wired I/O-board, and a set of wireless components that communicate with a wired uplink transceiver. The wired components are very similar to Phidgets and the wireless components naturally expand upon this concept. Each component presents itself as a USB device with a unique ID. Furthermore these components present themselves through "surrogate objects" on the computer, which the designer then can work with as a standard I/O device in her software of choice.

3.5 Boxes

Hudson and Mankoff at Carnegie Melon University, Pittsburgh, USA developed Boxes in 2006. It was developed to reduce the polarized typical prototyping development. Most prototypes either discuss/test form or interaction, and both aspects are only simultaneously tested at a very late stage in the process. Boxes want to combine these parallel developments at a very early stage by using readily available parts, such as cardboard, thumbtacks, tin foil and masking tape. [11]

As the developers describe, Boxes has a few important features:

- Implementation independency
- Form independent, works as long as tin foil and thumb tacks can be attached
- Supports the design process, iterative.
- Immediately executable
- Does not have a display

Similar to many other solutions Boxes relies on a hardware board with a micro-controller and basic I/O coupled with a software solution. However, it separates itself from other micro controllers on two key areas. By implementing hardware touch sensing, there is no need for actual actuators; the user can initiate a button-press by simply touching a thumbnail tack. The thumbnail tack is connected to the hardware board and even slight touches are interpreted, thanks to clever implementation of capacitive touch.

Secondly, the software is a variant of USB-controller mixed with a screen grabbing utility. Each thumbnail tack is associated with a desired action, such as clicking the mouse button at a certain coordinate on the screen. This is combined with grabbing a portion of the screen (such as the display of a already-made interface). Since the user captures an already existing portion of the screen, the designer can create her interface with whatever software tool she wants, thereby bypassing the need for direct software programming.

The creators tested their prototype with a few case scenarios and the response was largely positive, given it being a tool for early-stage
implementation. All of the scenarios showed the users quickly finding their ways to work with Boxes and adapting it to their workflow. However, some problems manifested during test; one of the main problems was not being able to use blue foam, so creating more complex three dimensional shapes was not possible.

3.6 dTools

dTools was developed in 2006 by Hartmann et al. at Stanford University HCI Group. It was developed to specifically target the design-analyze-test iterative cycle, omnipresent in design process. At the time, few tools directly targeted the iterative style of design, and most focused solely on the design part of the cycle. By implementing analyzing and testing tools from the get go, they are an integral part of the toolkit. [12]

dTools is based on many of the same principles as Phidgets or Calder and revolves around a USB-connected microcontroller with attachable modules. However, the microcontroller needs its own software to be programmed and is not directly intended for use with other software. The software focuses on a visual programming of the product, employing a graphical authoring environment.

The key features of the software are:
• A statechart editor, allowing a visual representation of the product and its I/O.
• A source code editor, implemented as a flow chart with visual representations of different states and arguments as arrows between them.

The benefit of a visual editor is an easier to use interface for designers not used to textual programming. It is reminiscent of a non-linear slide presentation, with different triggers between slides. Because of this, it has its clear limitations regarding the number of states and the possibilities; too many visual states simply become overwhelming.

To counter these limitations it is possible to use textual programming through Java. This gives the designer the ability to expand when the time is needed. If not, the simplicity of the initial visual authoring allows the designer to disregard textual programming if it is not necessary.

Beyond designing, the toolkit allows for user testing and analysis. This is done by logging all user-activity within the program as well as video-recording the user actions. By doing so the two are naturally coupled, allowing the designer to observe a user-reaction based on input. Events in the video and in the statechart overview are visually linked by color-coding, simplifying the identification of individual events. The designer can filter events or actions allowing her to observe specific sequences.

3.7 Display Objects

Akoak, Ginna and Vertgall at Queens University, Kingston, Canada, presented Display Objects in 2010. Display Objects uses motion tracking and video-projection to make inanimate objects come to life by presenting an image on its surface. The interaction of the object is tracked in real-time allowing direct manipulation of the object as well as the projected image. [13]

Display Objects was the result of observing designers and architects creating prototypes of their work. Based on this, some of their requirements for Display Objects were:

• Prototyping should be rapid and iterative.
• There should not be a need for separate software to program.
• Physical prototypes should be able to be made out of cheap and readily available components.
• Interactive displays should be provided seamlessly on the physical prototype.

Their system required several different components as well as a necessary setup of these. Because the system needs to work in a three-dimensional space the necessity of multiple cameras and projectors is absolute, to avoid the user shading his interactions from one of the cameras or projectors. The physical
models need several different markers, so the software can track it in 3d-space. The physical components needed:

- Multiple Vicon motion-tracking cameras
- At least two video projectors
- Physical models with attached IR reflective markers
- An Interaction Palette, containing interaction styles and skins

The system needs to be calibrated before use, and is a one-time process using a standard T-shaped model with markers.

Additionally, the setup involves dedicated software for recognizing basic shapes and geometries. Getting the software to identify 3d-models, virtual models are fed into the software by the user. These virtual models are simple geometries such as spheres, cubes or cylinders. By using simple models a balance is achieved between fidelity and speed of processing.

Using the system requires the user to augment his hands or fingers with IR-reflective markers so the motion cameras can track actions. Once this is done, the physical models can be interacted with either one-handed or two-handed and the system recognizes several different actions. Tapping, clicking, dragging, pinching, wiping, drag and drop are all recognized.

After developing an initial system the team tested it on two case studies and the feedback from users were mixed. Most users applauded the prototyping possibilities and the ease of use the system had. However, the actual usability of the system was limited due to the tedious setup with software as well as the many hardware components. Due to objects sometime being occluded, the usage felt unnatural and constrained. Also, the extensive hardware requirements made the setup and portability suffer; testing had to be performed in a designated locale instead of real-world testing.

### 3.9 Arduino

Arduino is an open-source microcontroller designed to make the use of electronics in projects more accessible. Massimo Banzi and David Cuartielles developed it at Ivrea in Italy. [14] It bases itself on several previous works, most notably Wiring and Processing. The Arduino Project is a fork of Wiring, a master thesis by Hernando Barragán. [15] Wiring was built as an electronic counterpart to Processing. Processing was a result of the Design by Numbers methodology, initiated at MIT Media Labs by John Maeda. [16]

Massimo Banzi worked as a professor at Ivrea and created Arduino to help his students fulfill their projects. Arduino was specifically designed to lower the entryway for designers and artists into the world of electronics and programming. [17]

Arduino consist of two parts, a single hardware board with an attached micro-controller as well as inputs, outputs and other circuitry; and a integrated development environment, which is a application allowing users to write programs as well as compile and upload them to the Arduino board through a standard USB cable. These two components greatly simplify the connection between the tangible hardware of electronics, with the virtual coding of computer programming. In fact, most users have written and tested their first program within few minutes after receiving the Arduino. [18]

The Arduino project consists of a large family of different hardware boards. The basic board is known as the Arduino Uno. They all serve different purposes, such as the Arduino Mega with more inputs and outputs, the Arduino Lilypad which can be used with textiles, the Arduino Nano for use in small designs. However, they all share the same basics as described above. [18]

An important aspect of the Arduino board is the standardized layout of the input and output pins. This allows the pins to connect to other boards, known as shields, which can simply be stacked on top of the main Arduino board. The logical layout and components of the hardware
Interactive Prototypes

Phidgets, Calder Toolkit and dTools use standard microcontroller and additional components. Most of the tools are based on a USB microcontroller and additional components attached to the microcontroller. The Arduino uses standard electric components, while Phidgets, Calder Toolkit and dTools use their own actuators and sensors. Boxes allows the usage of anything able to carry an electric current. Switcharoo uses RFID inputs coupled with an RFID receiver antenna.

Display Objects uses an extensive setup for motion tracking and real-time video projection. Several of the tools require some sort of computer programming, with Arduino being the most extensive. Phidgets, Calder toolkit and dTools also require coding, but through use of an API it simplifies the connection of code with actual components.

Switcharoo relies on Macromedia Director, and Boxes features a screen grabbing feature, allowing designers to work with existing software.

dTools is the only tool with a visual authoring tool.

4. COMPARISON

The tools presented have many similarities as they are all directed towards physical, interactive prototypes. However, they approach this area with very different ambitions and using very different components to accomplish the task. To simplify comparing the tools, we will focus on four main areas.

- Main components
- Which stage of the design process the tool is aimed at
- Desired fidelity of the prototypes created with the tool
- Portability

The tools vary greatly within these categories, making them suitable for comparison. A table is supplied on the final page.

4.1 Main components

Most of the tools are based on a USB microcontroller and additional components attached to the microcontroller. The Arduino uses standard electric components, while Phidgets, Calder Toolkit and dTools use their

4.2 Design stage

Many of the tools aim themselves squarely at the early stage of prototyping, and amongst the tools based on micro controllers; only the Arduino targets later stages. This is mostly due to the higher learning curve to get started. It is the only tool that requires computer programming.

With its extensive hardware setup Display Objects renders itself difficult to use in early stages, naturally skewing it to later in the design process.

dTools differs from the other by allowing a complete toolkit for not only design, but also analysis and testing. This gives it possibilities to be used in every stage, encouraging early testing and user analysis.

4.3 Desired Fidelity

Switcharoo and Boxes are the only two tools targeting a low fidelity; Switcharoo with its basic RFID tacks and Boxes with thumbtacks and tin foil. While this does not make them unsuitable for higher fidelity it definitely
hinders it naturally. Especially Boxes that lacked support for use on foam models.

Display Objects skews toward low to medium level of fidelity. This is mostly due to the awkwardness of occlusion when motion tracking objects.

Phidgets, Calder Toolkit and dTools aim themselves at a medium to high level of fidelity. There are possibilities with both to achieve a high fidelity, but in comparison to the other tools aimed at high fidelity they fall short. This is mostly due to the usage of own proprietary components instead of standard available components, limiting the possibilities.

Arduino is the only microcontroller-based tool aimed at high fidelity. This comes from a combination of using standard hobby electronic components, allowing a virtually endless number of possibilities, and the use of shields, allowing the Arduino board to expand in capability as desired.

4.4 Portability

Portability relies on a number of different parts of the system being portable and the only true portable tool is Arduino. Once the program is loaded onto the Arduino board it does not require a computer connected to it anymore. This is the case for all the other tools. However, with laptops a reasonable portability is achievable, but obviously in a completely different way than Arduino.

dTools employs a video camera for user testing, rendering it less portable than the other tools. The only truly un-portable system is Display Objects, which requires multiple cameras as well as video projectors.

5. DISCUSSION

While catering to solve the same basic problem, getting passive models to become interactive prototypes, the tools presented in this article have very different ways of solving that problem.

Today, Arduino has prevailed as a "winner" of sorts, being used by a larger audience each day and being implemented into design studies around the globe. By reviewing this other tools, can we see what makes the Arduino so popular and why it excels over the others.

One of the most important aspects is the openness of Arduino. Many of its advocates talk about how Arduino being open source has helped it spread. As Banzi himself said, "if enough people work on it, it’s like hundreds and thousands of people polishing a sculpture. Everybody’s got a bit of a sand paper, and together they make a beautiful thing. It’s a group effort" [17] and while this is obviously true I believe there are other aspects of the openness that have not been discussed.

By using standard components, instead of proprietary motors, buttons, etc, Arduino can use any electronic part thinkable. If a user demolishes a RC-car, the Arduino can quite easily use the innards. Even though this may be true for the others, Arduino directly encourages the use of it by not relying on its own parts. Furthermore, the use of shields to expand the Arduino has opened many possibilities. The shields are simple to create, thanks to Arduino’s openness, and when shown to the community, accelerate further innovation.

Several of the other tools catered towards the early parts of the design process. This early stage already has several readily available prototyping tools, most of which are already well established. One might argue that these tools do not allow for interactivity, but perhaps this is not necessary. In the early stages, different aspects of the design are being investigated in parallel. By introducing more complex tools and investigations, innovation might be lackluster.

One can compare with a typical interaction design process, where sketches are made, then wireframes, then paper prototypes, and finally software prototypes. Physical models might correspond to wireframes and it is not until the next phase (paper prototyping) that
interactivity is introduced. Likewise, interaction in product designs needs to wait for a later stage.

This resonates with the findings of Yang, [5] where prototypes with less complexity performed better than those with a higher degree of complexity. A very obvious reason for this is by introducing more parts, more variables need to be accounted for, thereby delaying the design and making the designer focus on troubleshooting and debugging instead.

Arduino is obviously not without faults, and one of them is the steeper learning curve, when compared to other tools. Since designers have to learn electrical wiring and computer programming, the price to enter may seem high. The API-based solutions of, for example, Phidgets seems more in line with the capabilities of the average product designer.

However, this higher threshold for getting started also correlates to higher fidelity of the final prototypes. One can compare this to musical instruments. Skeie [20] found in a review article of musical interfaces, that while more complex interfaces were more difficult to master, those who took the time to master them were able to produce better results. Likewise, musical interfaces that were simple to master did not encourage the creation of complex musical arrangements.

5. CONCLUSION

Being able to create interactive prototypes is an important asset for product designers. There are several tools available, with the most prominent being the Arduino. This article looks at other tools as well as the Arduino to survey the landscape of interactive prototyping tools and how they could fit into a design process. By comparing different tools by certain categories, it is evident in which areas the Arduino excels and falls short.

The strengths of the Arduino lies in its possibilities as a platform. Thanks to openness and readily available components from a slew of producers, the basic Arduino board can easily be expanded however the designer wishes. This is in stark contrast to many of the other tools that lock the designer in to a proprietary system.

The weakness is its obvious steep learning curve for designers not used to electrical- or computer engineering. For many this might be a too big hurdle to overcome. However, its steep learning curve is a byproduct of its many abilities, making it most suited for high fidelity prototypes.

In my work with this paper, I found that research surrounding these tools and the Arduino in a design process is greatly lacking. The author proposes research on interactive prototyping as a tool for industrial designers is carried out to supplement this paper and future work.
REFERENCES


<table>
<thead>
<tr>
<th></th>
<th>Phidgets</th>
<th>Switcharoo</th>
<th>Calder Toolkit</th>
<th>Boxes</th>
<th>dTools</th>
<th>Display Objects</th>
<th>Arduino</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Main components</strong></td>
<td>USB</td>
<td>USB Microcontroller</td>
<td>USB Microcontroller</td>
<td>USB Microcontroller</td>
<td>USB Microcontroller</td>
<td>Motion tracking cameras,</td>
<td>USB Microcontroller</td>
</tr>
<tr>
<td></td>
<td>with additional</td>
<td>with touch capacitive buttons</td>
<td>with additional components</td>
<td>with additional components</td>
<td>with additional components</td>
<td>projectors, control-board</td>
<td>IDE</td>
</tr>
<tr>
<td></td>
<td>components</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Directed towards</strong></td>
<td>Early stage of prototyping</td>
<td>Early stage of prototyping</td>
<td>Early stage of prototyping</td>
<td>Early stage of prototyping</td>
<td>Complete design-analysis</td>
<td>Interaction testing with</td>
<td>Late stage of prototyping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>test prototyping</td>
<td>video projection</td>
<td></td>
</tr>
<tr>
<td><strong>Fidelity</strong></td>
<td>Mid-High</td>
<td>Low</td>
<td>Mid-High</td>
<td>Low</td>
<td>Mid</td>
<td>Low-Mid</td>
<td>High</td>
</tr>
<tr>
<td><strong>Portability</strong></td>
<td>Partially</td>
<td>Partially</td>
<td>Partially</td>
<td>Partially</td>
<td>Partially</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>