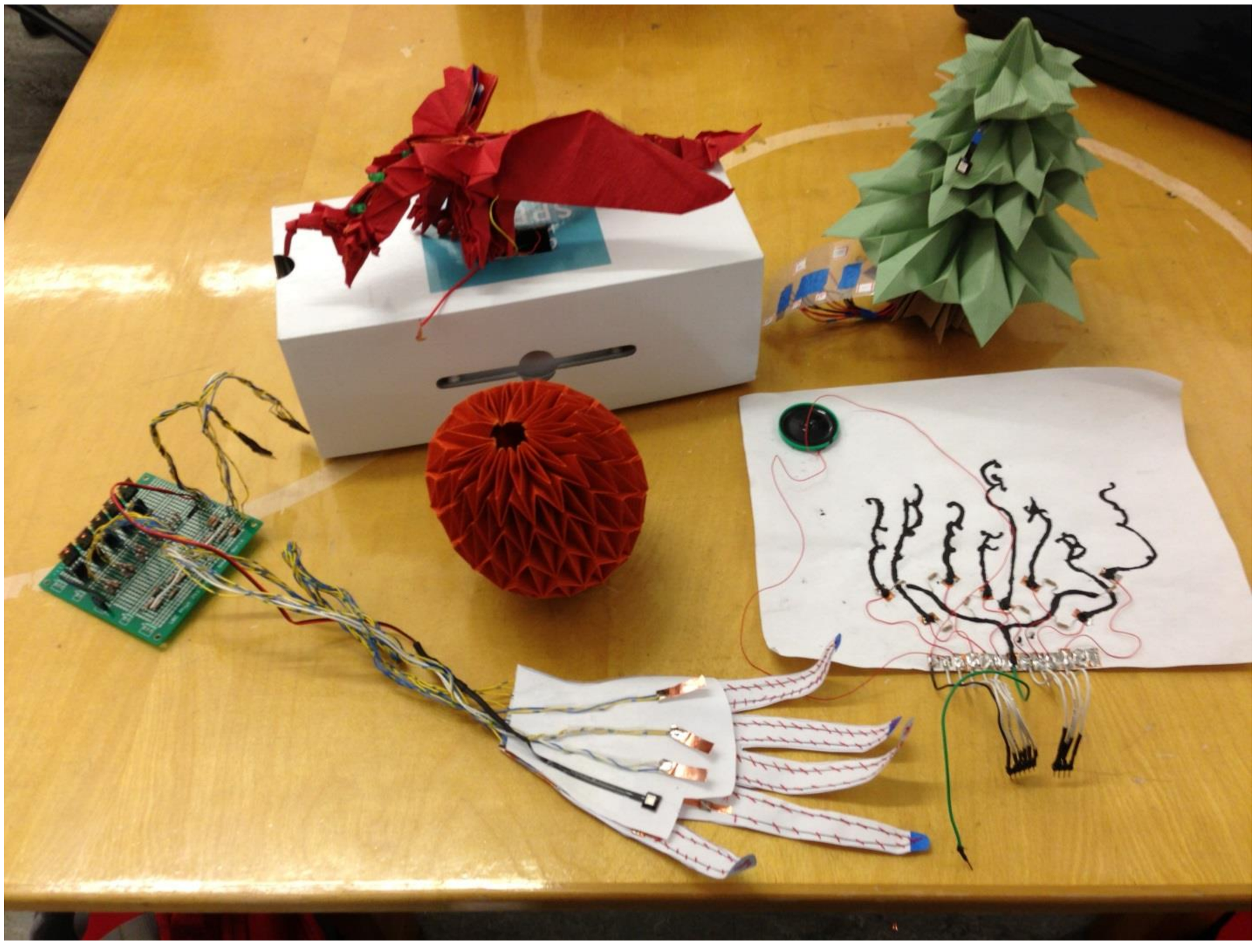
ROBOTIC ORIGAMI



Jianxing Niu
Tess Peng
Vicky Wang

Project Sponsor:
Dr. James Olson

Engineering Physics 479
Engineering Physics Project II
The University of British Columbia
January 7th 2013

EXECUTIVE SUMMARY

The Robotic Origami project is sponsored by Dr. James Olson at the University of British Columbia for ENPH 479 Winter 2012. The scope of the project is to incorporate electronic circuits onto paper and produce an environment sensitive origami showpiece that is composed of about 80% of paper based products. The design should be artistic in visual appearance, cheap and easily mass producible with the ideal of becoming a consumer product.

The group explored different methods for placing electric circuits onto paper such as printing with conductive ink, drawing circuits on paper with Bare Paint and using copper tape to attach electronic components onto paper. Each technique has its advantages and drawbacks. For example, the most promising technology, namely conductive ink, turns out to be printing on a plastic sheet instead of paper, rendering itself undesirable for origami. Nevertheless, this technology is demonstrated in a circuit for driving LEDs on a Christmas tree. The Bare Paint technology is used in a paper based piano.

A significant amount of time is spent on incorporating Flexinol, a type of Shape Memory Alloy (SMA), with origami designs. The methods for attaching Flexinol have been limited to taping and sewing the wires directly onto the paper surface, which severely limited the geometry in which the Flexinol could flex. Thus Flexinol, while being strong and powerful in terms of force per weight, may not provide enough pull force to a piece of origami for noticeable motion. For instance, six methods of incorporating Flexinol with magic ball origami are implemented and found to be ineffective. In the end, a piece of robotic dragon origami is made, which can sense ambient light, sound and touch and respond by actuating the Flexinol to move its wings or tail and play a short piece of music. Interactive Fingers that implement the Flexinol and capacitive touch sensors are also produced.

We believe that robotic origami is of great interest and potential and it is worthwhile for future group to further investigate relevant technologies and apply to other paper products. Specifically ideas such as: incorporating Flexinol in different geometries to paper, exploring the use of paper-based sensors and printable batteries and explore substrates that printers can print onto paper instead of plastic are noteworthy.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

This section outlines the significance of the project. It provides the motivation, objectives, the scope and the limitations of the project.

1.1 Background and Motivation

In the last two years, two groups from ENPH 459 and 479 courses worked on projects kindly sponsored by Dr. Olson from the Pulp and Paper Centre; both projects are related to paper products. In particular, one group of students purchased a machine that would cut and make crease lines on the paper, and when the paper is heated in the oven, it will automatically fold into the desired geometry due to the presence of a smart shape alloy. The result of that project inspired this year's project to design and build a paper based product that is responsive to external environment by actuating simple motion.

Paper is easily available, cheap, biodegradable and a renewable resource that is used for a variety of applications. Electronics are constantly being discarded with the influx of new technologies. We aimed to minimize the harmful environmental impact of such wasteful behaviors by replacing most of mechanical aspects of electronics with paper. As a start-up project in this field of interest, we built some origami that demonstrates the potential of fusing paper products with electric components. Such technology may lead to new innovative paper based products with real-life application in either functionality or entertainment.

1.2 Project Objectives

Sponsor Dr. James Olson stated the project objective is to design inexpensive movable autonomous origami shapes. A variety of "paper-based" and non-paper based technologies should be used. The origami serves as a demonstration/test bed for the new generation of paper based technologies. The robotic origami should satisfy the following requirements:

- Be artistic and visually appealing
- Senses its environment
- Move in response to the sensing
- Largely made from conventional printing paper
- Self-contained
- Inexpensive and easily mass producible

1.3 Scope and Limitations

The original scope, as detailed in the project objectives, is achieved. This project has produced: a dragon origami which can sense ambient light, sound and touch and respond by actuating a piece of Flexinol to move its wings and/or tail and play a piece of music; a hand that senses touch and curls its own fingers in response; a piece of Christmas tree origami with a LED driver circuit to demonstrate printable conductive ink technology; and finally a paper piano to demonstrate hand-drawn circuits made with Bare Paint.

Despite the success of fulfilling the objectives and producing creative origami, the project still has a few limitations. Firstly, the usage of Flexinol as the muscle of the origami limits the forms of the origami. Poor methods of attachments and the geometry between Flexinol and paper do not allow the Flexinol to produce significant pulling motions. Secondly, the three paper circuit production methods: printable conductive ink, Bare Paint and copper tape, all have limitations such as ridged structures being non-suitable for folding, low conductivity or troublesome to implement. As a result, while the team has origami that demonstrates different paper circuits, the dragon origami and finger origami utilize printed circuit boards.

1.4 ORGANIZATION

The report will first discuss usage of Flexinol, the constructions of the dragon origami, paper piano, interactive fingers, experiences with magic ball and then printable paper-based circuit. Finally the report will present the findings, results, conclusions and recommendations.

2. DISCUSSION

2.1 PAPER-BASED CIRCUIT

The following sections explore the usage of different paper-based circuits.

2.1.1 INTRODUCTION

In addition to exploring the design of showpiece of interactive art from paper with sensors and actuators embedded on the inside, we are also interested in paper-based technology. In particular, we would like to delve into paper-based circuit. As part of the goal in this project, three technologies are studied and explored: paper-based circuit with Copper tape, paper-based circuit with conductive pens and printable paper-based circuit with Silver ink.

Table 1 below provides a summary of the sub-projects that are delivered and the components involved.

Sub-projects	Sensing of the Origami	Actuation of the Origami
Dragon	One light sensor (purchased)	Flexinol (purchased)
	One capacitive touch sensor (designed)	

	One sound sensor (purchased)	
Interactive Fingers	Three capacitive touch sensors (designed)	Flexinol (purchased)
Paper Piano	Eight capacitive touch sensors (designed)	N/A
Origami Christmas Tree	Six Surface mount LEDs (purchased)	N/A

Table 1. List of Deliverables.

2.1.2 *Paper-based circuit with Copper tape*

Copper tape is one of the most commonly used technologies for paper-based circuit for its good conductivity and ease of manipulation. Hence it is the first technology that we look into by constructing a simple LED circuit with a 9V battery. The resistance that we measure is very small, and we use copper tape for all the origami design pieces for the same reason without worrying about loose connection and bad conductivity.

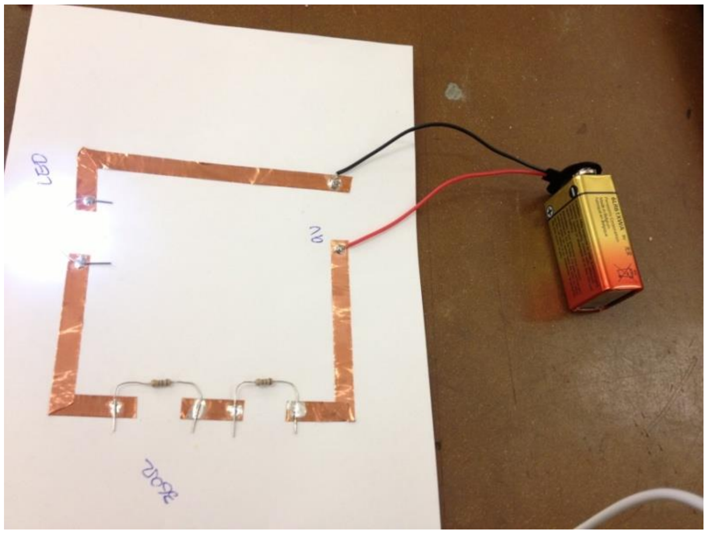


Figure 1. LED circuit with Copper tape.

2.1.3 Paper-based circuit with Conductive pens

The conductive pens we used are called Bare Paint, which has some internal resistance. The resistance of the circuit path varies throughout the circuit due to human errors such as non-uniform thickness and width of lines being drawn, but it is roughly on the order of 50ohms per centimeter. According to Bare Paint's website, it can be used to produce resistors (which are really hard to measure) and capacitors (which require rather large pieces of paper). We build a paper piano to demonstrate the usage of conductive pens.

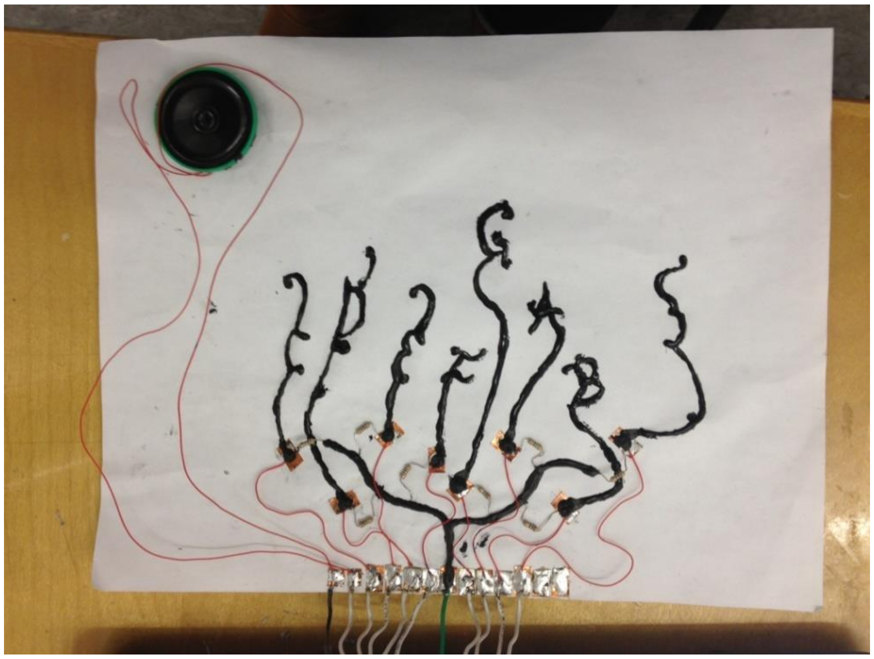


Figure 2. Paper piano made with conductive pens.

2.1.3.1 Electrical End of Paper Piano

All the circuitries for the paper piano are put onto a single piece of 9'' by 11'' piece of paper. Due to the limitation of the Arduino Uno board of having a maximum of thirteen digital pins, we can only have 8 notes for our piano.

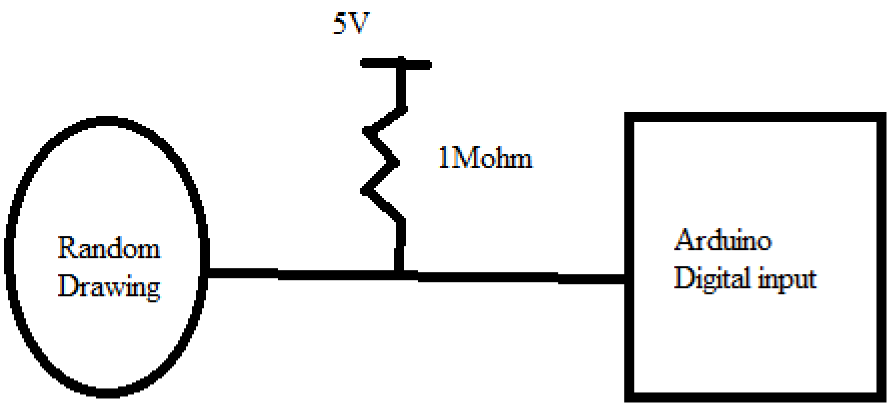


Figure 3. Circuit for the touch sensors for the paper piano, the random drawings are the keys to the piano.

The paper piano utilizes simple touch sensors for the keys. The Arduino has an internal pull up resistor (30kohm); however, it is too small for this purpose since the Bare Paint has such large resistances. By using a 1Mohm resistor, we have the freedom of drawing really long circuits. An Arduino function first sets the pin to ground and turns on the internal pull-up resistor. Then it measures the time for the pin to return to the high state. This value is 1 if the wire is not touched and it reads 17 when the pin is directly grounded. It is found that a skin touch will read to 2 or 3. The response time for the touch sensor is about one second. It's noteworthy that this function works on Arduino Uno R2 and R3 board but does not work on an older version of board. This design also works with graphite pens; the user can simply touch the drawing with a piece of paper in between without smudging the paper.

2.1.3.2 Algorithm of Paper Piano

Similar to the touch sensor used in the dragon, when no printed trace is touched, the function reads 1 and no note is played; when any of the drawn traces is touched, a function that measures the capacitance will read a value larger than 1, and then points to the speaker to play the note for a constant period of time. The optimized note duration is found to be 300ms by trial and error. The pseudo code is written below. The actual code is included in Appendix A.

```
Int touchedCutoff=2;
Int noteDuration=300;
Void loop()
{
        If (readCapactivePin(pinNoteA5)>touchedCutoff)
           Tone(pinSpeaker, NOTE_C5, noteDuration);
        //Then repeat the above code for different notes)
```

}

2.1.3.3 Test Result

The goal of the paper piano is to firstly serve as a demonstration of the conductive pen, and secondly mimic a real piano that can play different notes. As a result, the test protocol is for the user to place one finger at a time on the note, and the paper piano should play the note that is being touched. One thing to note is that the duration and order of each note being played does not matter. By carrying out our test protocol, the paper piano works as expected.

2.1.4 Paper-based circuit with Silver ink

2.1.4.1 Introduction

At the beginning of the term, as we were investigating various technologies of printed paper-based circuit, we came across to find a unique technology developed by NovaCentrix. The manufacturer develops metallic ink that can be refilled into regular ink cartridges and then installed in a commercially available printer. We can design and draw circuit in Word document or any other format, and print out the circuit straight from the printer. An inkjet kit that includes 50-mL of silver ink, 150-mL aqueous solution, an Epson printer (Model number: WorkForce 30 Series), a pack of 20 sheets of Novele substrate, five regular graphic inkjet cartridges, and five empty inkjet cartridges were purchased to study this technology.

2.1.4.2 Components

For the conductive ink, we decide to use water-based Silver ink (JS-B25P) for a number of reasons. First of all, water-based ink is easier to clean after we are done printing, and it is also less problematic when it comes to cleaning the nozzles inside the printer and the print heads of the ink cartridges. Moreover, JS-B25P is one of the most conductive inks among all the silver inks developed by this manufacturer. In addition, an aqueous solution is required to purge the nozzles after we are done printing.

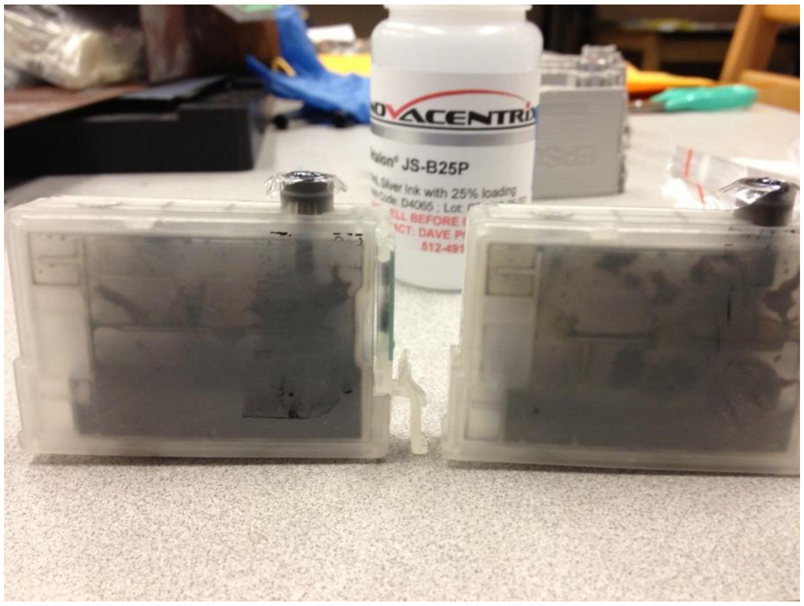


Figure 4. Picture of water-based Silver ink (JS-B25P).

In inkjet printing, drop-on-demand method is one of the most popular approaches that are used nowadays. Drop-on-demand (DOD) method can be divided into two subdivisions: thermal DOD and piezoelectric DOD. Printer Manufacturers such as Canon, Hewlett-Packard (HP) and Lexmark use thermal DOD process. On the other hand, Epson and Brother use piezoelectric DOD. In piezoelectric DOD method, a piezoelectric material is used behind each nozzle, and it changes shape when a voltage is applied across it. A pressure pulse is then generated due to the change in shape which forces the ink droplet to come out of the nozzle. Another advantage is that it allows a wider range of inks compared to thermal DOD method, which is why we use Epson printer in our application.

For paper-based circuit, the dominant factor that affects the conductivity is how well the Silver powders deposit onto the paper substrate. Therefore, according to the communication with the manufacturer, a chemically coated substrate named Novele substrate that looks like a plastic sheet is used for the printer to deposit silver powders onto it. First of all, the substrate is porous so it is more likely for the silver powders to stay on the paper. Second of all, in order to improve stability of the silver powders to attach to the substrate, it is also recommended by the manufacturer to thermally treat the printed circuits for a couple of hours. If we are to air dry the printed circuits, the printed samples have to be dried for a few hours.

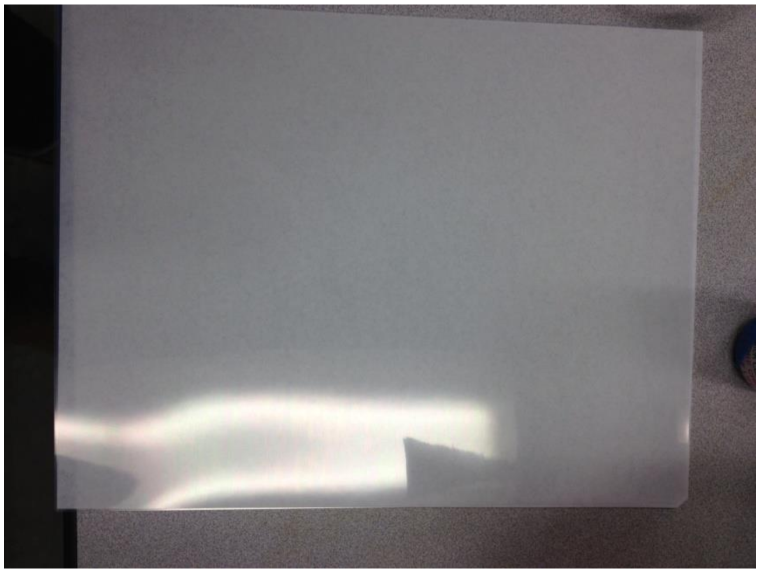


Figure 5. Picture of Novele substrate.

2.1.4.3 Procedure of printing circuits

In this section, we will describe the procedure of how to refill the ink cartridges and print the circuits.

The procedure can be divided into five main steps: nozzle check of the graphic cartridges described in the blue box in the figure bellow, cleanout on individual colors using aqueous vehicle provided from the manufacturer (yellow box), nozzle check of the ink cartridges containing silver ink (green box), print task of the designed circuits (red box), and purging of individual colors using aqueous vehicle after the print task (yellow box again).

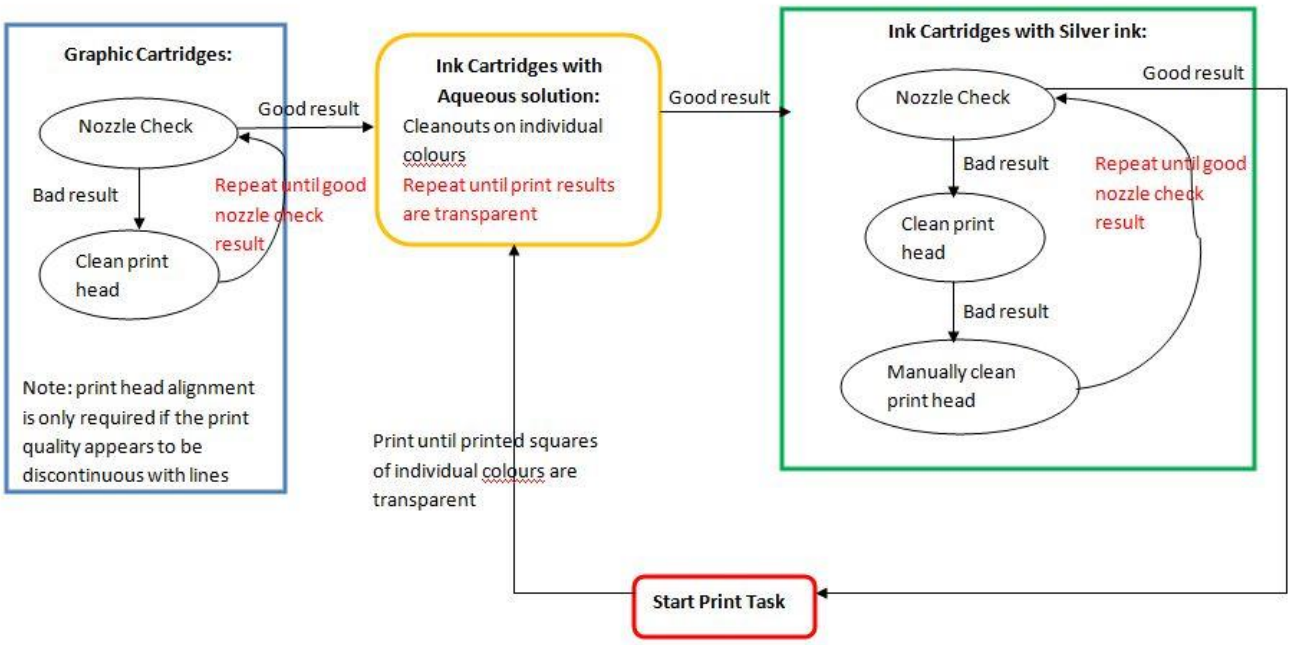
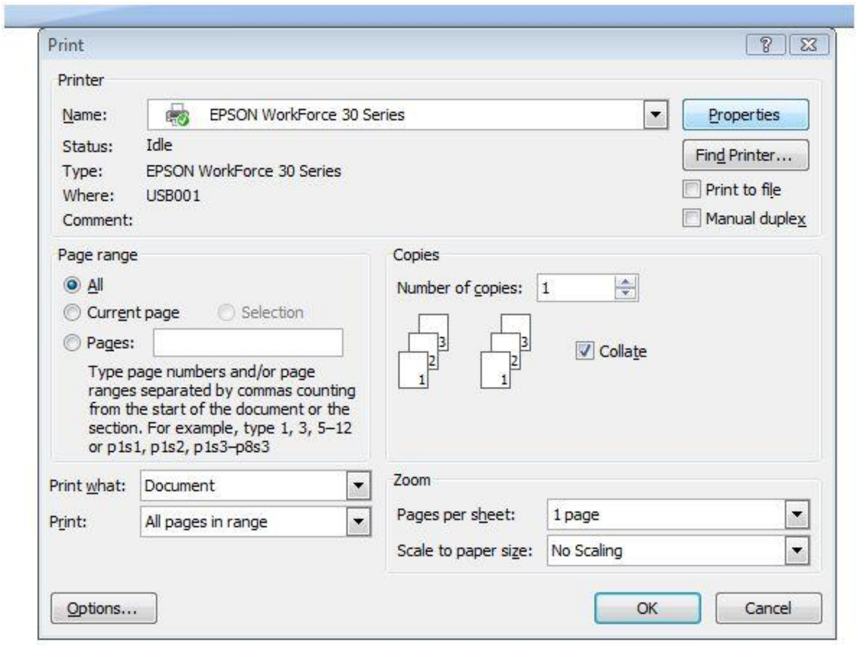

Graphic Cartridges:
Nozzle Check
Bad result
Clean print head
Good result
Repeat until good nozzle check result
Note: print head alignment is only required if the print quality appears to be discontinuous with lines
Ink Cartridges with Aqueous solution:
Cleanouts on individual colours
Repeat until print results are transparent
Good result
Ink Cartridges with Silver ink:
Nozzle Check
Bad result
Clean print head
Bad result
Manually clean print head
Repeat until good nozzle check result
Good result
Print until printed squares of individual colours are transparent
Start Print Task

Figure 6. Procedure of how to print paper-based circuit.

- Nozzle check of the graphic cartridges (blue box)

Before performing nozzle check of the graphic cartridges, the user should have the designed circuit ready and open in any file format. In our case, we use word document. First of all, we need to go to the print icon and select properties.



Print

Printer
Name: EPSON WorkForce 30 Series
Status: Idle
Type: EPSON WorkForce 30 Series
Where: USB001
Comment:

Page range
- [✓ All]
- Current page
- Selection
- Pages:

Type page numbers and/or page ranges separated by commas counting from the start of the document or the section. For example, type 1, 3, 5–12 or p1s1, p1s2, p1s3–p8s3

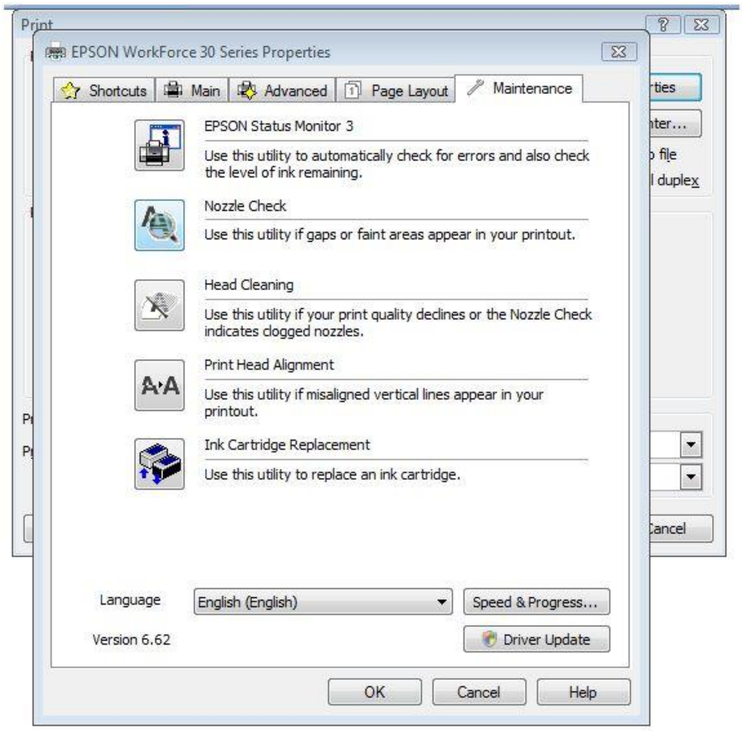
Copies
Number of copies: 1
Collate ✓

Print what: Document
Print: All pages in range

Zoom
Pages per sheet: 1 page
Scale to paper size: No Scaling

Figure 7. Print Properties.

Then, under Maintenance tab, first select Nozzle check and then click print so the printer can carry out the task.


EPSON WorkForce 30 Series Properties

EPSON Status Monitor 3

Use this utility to automatically check for errors and also check the level of ink remaining.

Nozzle Check

Use this utility if gaps or faint areas appear in your printout.

Head Cleaning

Use this utility if your print quality declines or the Nozzle Check indicates clogged nozzles.

Print Head Alignment

Use this utility if misaligned vertical lines appear in your printout.

Ink Cartridge Replacement

Use this utility to replace an ink cartridge.

Language: English (English)

Version 6.62

Figure 8. Nozzle check option under Maintenance tab.

Repeat the steps described above until the quality of the printed lines is similar what is shown in the figure bellow.

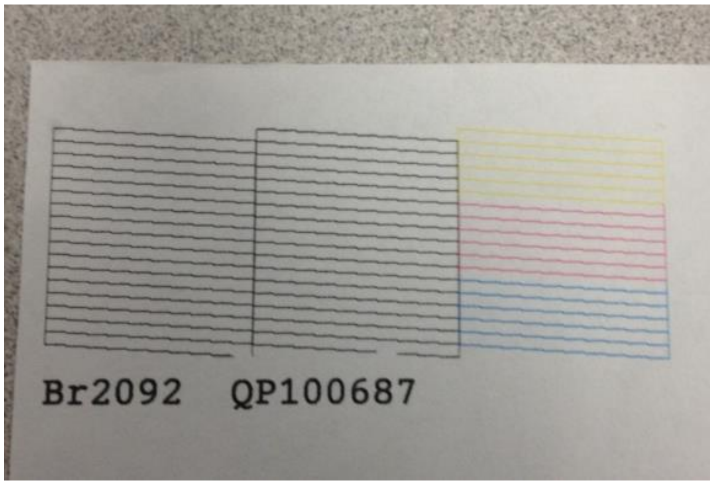


Figure 9. Printed lines from the Nozzle check.

- Cleanout on individual colors using aqueous vehicle (yellow box)

We need to perform cleanout on individual colors (black, yellow, red, and blue) using the aqueous vehicle provided by the manufacturer. We will use black color as our example for demonstration. There is a cleanout document for each color that needs to be opened first, and the one for black is shown in the figure bellow.

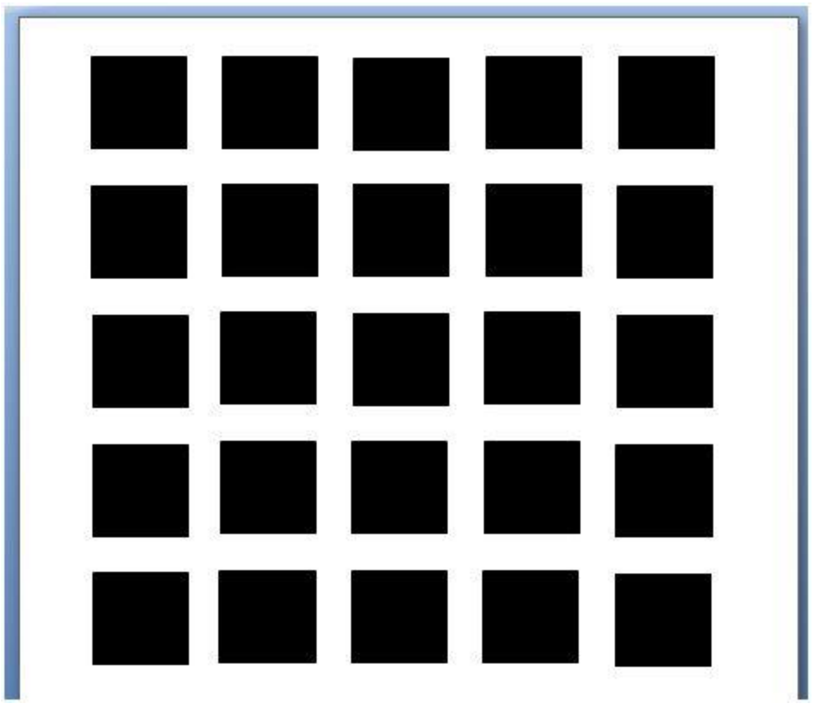
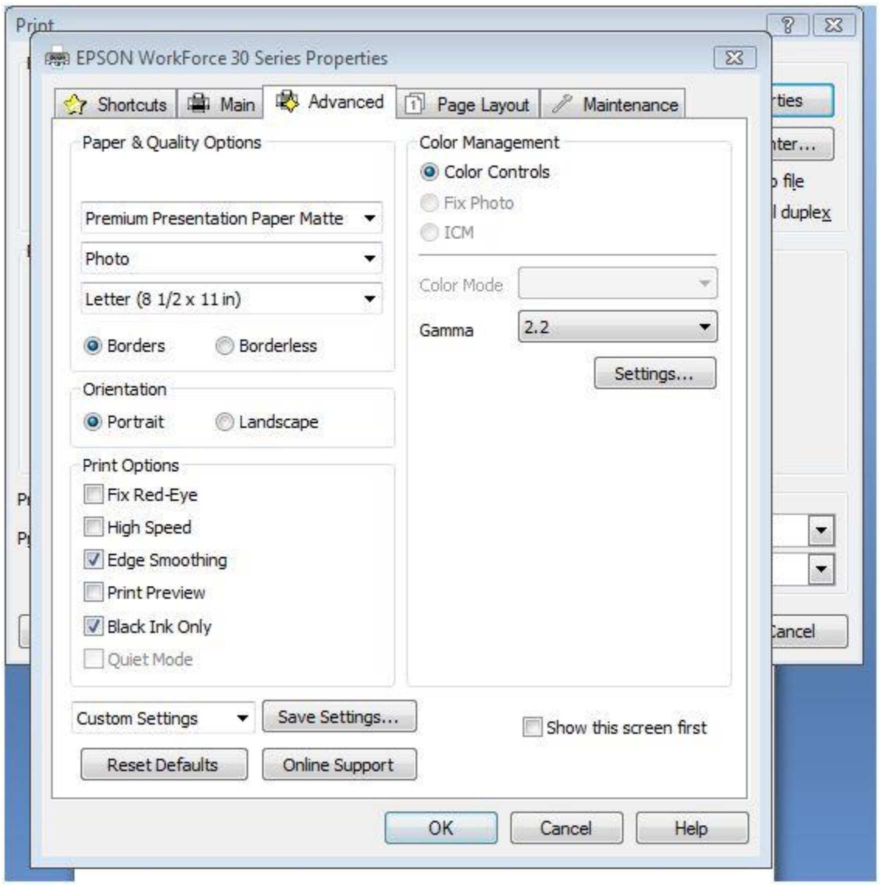


Figure 10. Cleanout document for black cartridge.

Then under the advanced print option, make sure "high speed" is not high-lighted and "black ink only" is select (Figure 11).


EPSON WorkForce 30 Series Properties

Shortcuts | Main | Advanced | Page Layout | Maintenance

Paper & Quality Options
- Premium Presentation Paper Matte
- Photo
- Letter (8 1/2 x 11 in)
- [● Borders] Borderless

Orientation
- [● Portrait] Landscape

Print Options
- [ ] Fix Red-Eye
- [ ] High Speed
- [✓] Edge Smoothing
- [ ] Print Preview
- [✓] Black Ink Only
- [ ] Quiet Mode

Color Management
- [● Color Controls] Fix Photo ICM
- Color Mode
- Gamma: 2.2

Custom Settings | Save Settings... | Reset Defaults | Online Support | [ ] Show this screen first

OK | Cancel | Help

Figure 11. Advanced print option for cleanout.

- Nozzle check of ink cartridges containing Silver ink (green box)

Repeat the same steps described in the previous section for the graphic cartridges. The only difference here is that we need to replace the graphic cartridges with the silver ink cartridges and the cartridges that contain the aqueous vehicle. The two silver ink cartridges are to be replaced with the black ink cartridges, and the rest of the graphic cartridges (red, yellow and blue) are replaced by cartridges containing aqueous vehicle. The only printed lines that will be shown are the ones with the silver ink because the aqueous vehicle is transparent. Repeat this section until good nozzle check result is obtained.

- Print task of designed circuits (red box)

Then we can start printing our design circuit. The Novele substrate for the Silver ink is transparent; we need to trick the optical sensors inside the printer into thinking that it is like regular ink-jet paper by stapling another piece of paper underneath the Novele substrate. We also need to remember that the "high speed" option under the advanced print option is not selected because we would like to ensure a sufficient amount of ink is laid out on the substrate. Then the printed circuits can be air dried for several hours. To increase the conductivity of the printed samples, we can also try to put them in the vacuum oven so heat can help ensure that good conductivity is obtained.

- Purging of individual colors using aqueous vehicle (yellow box)
  Repeat the same procedure described in the previous section, "Cleanout on individual colors using aqueous vehicle," until the printed squares appear to be clear without any residue of Silver ink. On average, it takes about ten to twenty pages for the colors to be completely purged.

2.1.4.4 Demonstration of printable paper-based circuit

In order to demonstrate the technology of printing circuits directly from a commercially available printer, we made an Origami Christmas tree with four layers in total as our example. First of all, we can print a simple LED circuit from the printer, let the printed circuit air dry for a day and tape two ends of each surface mount LEDs on the circuit. Then starting from the bottom layer, we hide the electrical wires of the LEDs inside the tree and only leave the LED part on outside, and repeat the same procedure for each layer.

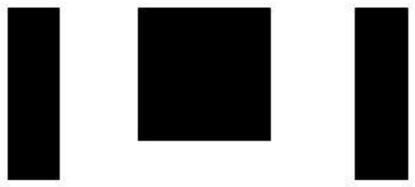


Figure 12. LED circuit for the conductive ink.

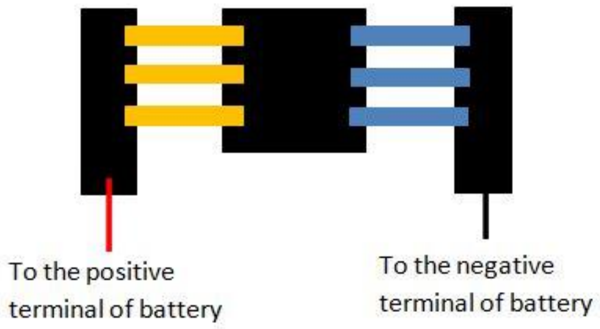

To the positive terminal of battery
To the negative terminal of battery

Figure 13. LED circuit with surface mount LEDs and battery drawn.

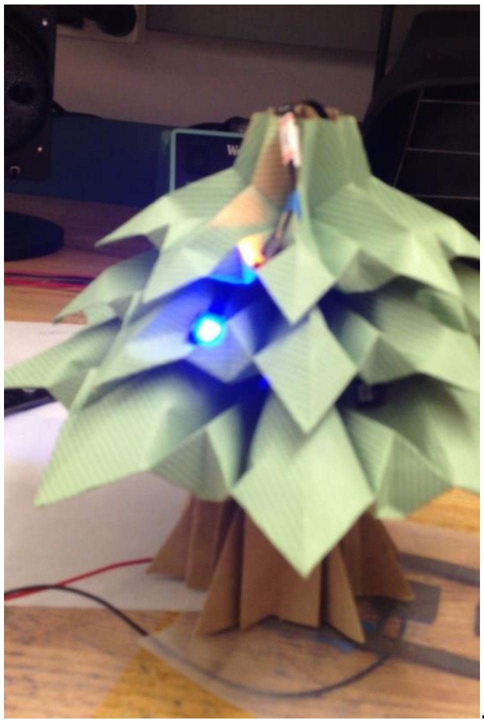

.

Figure 14. Christmas tree with LED.

2.2 Shape Memory Alloy: Flexinol

Shape Memory Alloy (SMA) is an active material that exhibits a mechanical response when subjected to thermal changes. SMA has relatively high actuation energy densities and is able to recover its shape even under high loads.

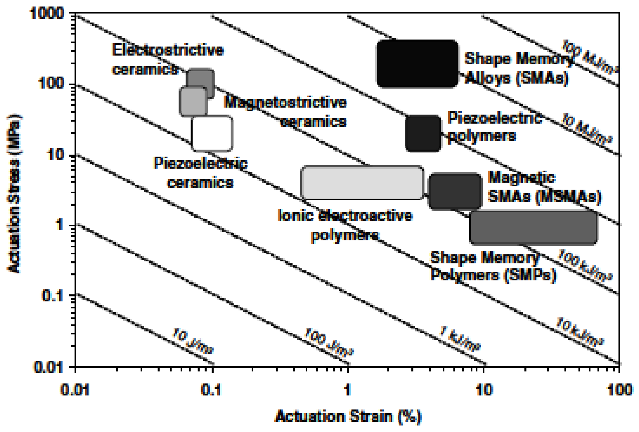


Figure 15. Actuation energy densities diagram for different active material that exhibit direct coupling.

However SMA have low actuation frequency response, but due to the scope of the project, this is more than satisfactory.

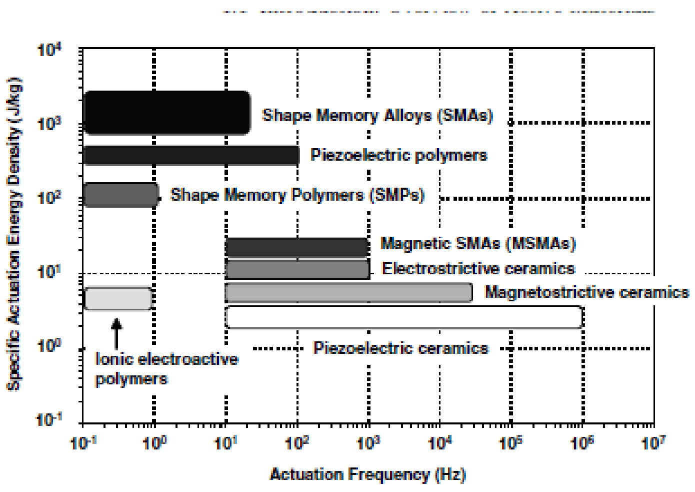


Figure 16. Actuation frequency diagram for different active materials that exhibit direct coupling.

SMA has different phase structures at different temperature; at high temperatures (austenite), the crystal structure is typically cubic; at low temperatures (martensite), the crystal structure can be tetragonal, orthorhombic or monoclinic. Twinned martensite is the “natural” crystal structure; detwinned is the reorientation of the crystal back to its natural crystal structure after stress is applied. SMA has reversible phase transformation, thus shape transformation.

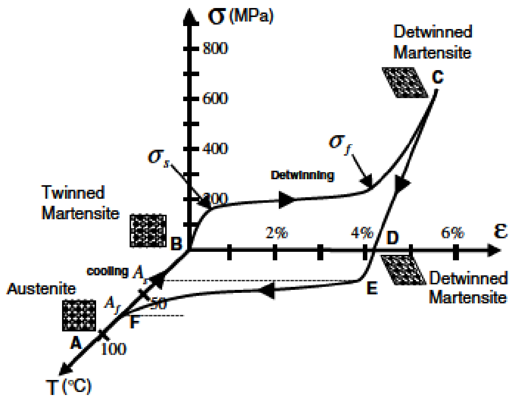


Figure 17. Stress-strain-temperature data for NiTi SMA.

SMA can be made from a variety of different alloys, such as nickel and titanium (NiTi) based, copper based or iron based. NiTi alloys are the most extensively studied and the relationship between the composition and the transformation temperature is well known. NiTi alloys are more expensive than other SMAs, but they have a wider hysteresis. NiTi is also corrosive resistant and biocompatible.

Flexinol is made from NiTi and will contract to 10% of its original length when heated to 70C. Reaction time and pulling force depends on the length of the SMA wire and its diameter. The method that will be used in this project is to simply run current through the SMA and it will heat up as more power is drawn from the power source. Thus repeatable actuation is achieved with only electricity, which removes the need for additional equipment such as heat guns or induction coils. Flexinol have a rather large pull force in respect to its weight and size. For example, a piece of wire 0.006 inches in diameter has a pulling force of 321 grams. It is fairly flexible, thus allowing us to put it into almost any shape and form inside the paper. Flexinol is readily commercially available and comes in a variety of different thickness and pull strength. For our application of moving paper pieces, this is a very good choice for a motion actuator.

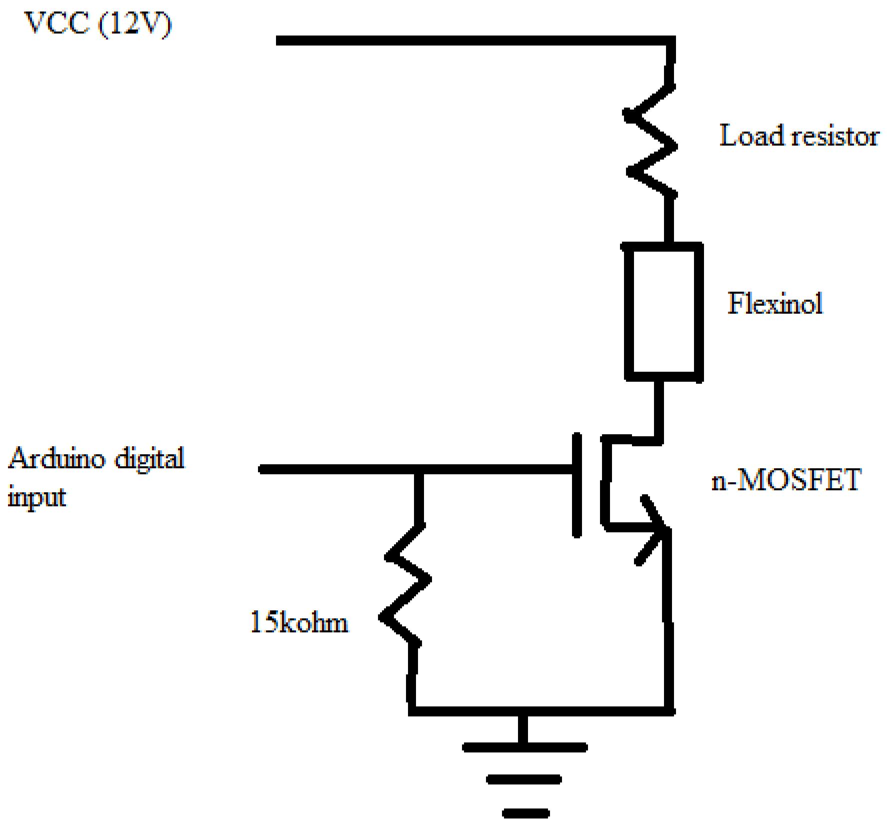


Figure 18. Electric circuit diagram for SMA.

The figure above is a circuit that turns on the MOSFET and activates the Flexinol when the input is high and off when the input is low. The calculation for the load resistor depends on the input voltage, the length and diameter of the Flexinol. The following is a sample calculation for the load resistor. Let the Flexinol be 0.006’’ in diameter and 10cm in length, which have an internal resistance of 1.3ohm per inch and an activation current of 0.4A. Let the circuit be powered by a 12V battery.

Voltage input = V_i = 12V
Length of wire = L_{wire} = 10cm
Activation current = I_{active} = 0.4A
Resistance per unit length of wire = ρ_{wire} = 1.3ohm/inch
Resistance of wire = R_{wire}
Load resistor = R_{load}

$$R_{wire} = L_{wire} * \frac{1inch}{2.5cm} * \rho_{wire} = 10cm * \frac{1inch}{2.5cm} * \frac{1.3ohm}{inch} = 5.2ohm$$

$$R_{load} = \frac{V_i - R_{wire}I_{active}}{I_{active}} = \frac{12V - (5.2ohm)(0.4A)}{0.4A} = 24.8ohm$$

If two pieces of Flexinol wires of the same length are put in parallel, then you need to put two load resistors in parallel, and so forth for three and more pieces of Flexinol in parallel. You do not want to use only one resistor for parallel Flexinol circuits (unless it's a power resistor), because the high current will burnout the resistor. In fact turning on the MOSFET for more than 2 seconds will also cause the load resistor to burnout. It is advised to always round up the calculated load resistor value. By putting Flexinol in parallel you can increase the pulling force, the activation time is shortened drastically, but the cool down time is increased.

Flexinol can also burnout if more than 0.4A of current runs through the wires and the paper that it is attached to will burst into flames. For the 0.006'' Flexinol, its activation time is 1 second and its cool down time is 2 seconds. Heating the Flexinol for more than the activation time will cause the Flexinol to burnout and insufficient cool down time will also result in burnout. Burnout Flexinol will contract less or seize contraction when reheated.

Flexinol cannot be soldered, in order to attach the electric wires the ends of the Flexinol are wrapped with copper tape to make little crimping beads which can be soldered. The Flexinol are attached to the paper via sewing to prevent tearing, tapes are put over the paper before sewing. Since Flexinol has very small contraction length, which is listed as 10% but observed to be around 5-8%, the Flexinol is attached in a "U" or "M" shape to maximize the length of Flexinol to be used. We observe that having too many sharp-angle bends will result in too much friction, so the contraction will be localized to the straight part of the wires, and there might not be an increase in contraction length, however the pull strength should increase.

Although different geometries have been tried, the only noticeable motion achieved is a curling motion when the Flexinol is sewn onto the flat surface of a piece of paper. The Flexinol must be attached under tension to generate the greatest contraction. Any crease in the paper will cause the

Flexinol to lose tension, thus it is not possible to produce hinges or any sharp movements. The Flexinol should also be sewn in a way that as much of the surface is in contact with the paper, so that the wire and paper will move uniformly as one and not the Flexinol trying to pull itself away from the paper. The paper used should be stiff enough to pull the Flexinol back under tension but flexible enough that there won't be any crease made when the paper is bent. An overwhelming quantity of moisture from the hand can be soaked up by the paper during sewing, which causes the paper to become limp and thus the Flexinol will lose tension. Gloves should be worn while sewing and the stitches should be small, close together and uniform in distance.

However even the most carefully planned and replicated Flexinol to paper sewing may not produce the same result. Thread can tear the paper when the Flexinol is activated. Flexinol might have overheated during soldering (since soldering iron is usually heated to 70C), and the paper itself could have different stiffness in various locations. All in all, paper is just not a very stiff material and the poor method of attachment doesn't allow the Flexinol to exhibit all its pulling force.

2.2.1 *Magic Ball, Flexinol to paper configuration tests*

The following describes the production of Magic Ball origami, most notably the different Flexinol and paper attachment configurations.

2.2.1.1 Origami Design

The origami pattern is from this website: http://www.youtube.com/watch?v=tGhcTwIJ4Es&feature=related.

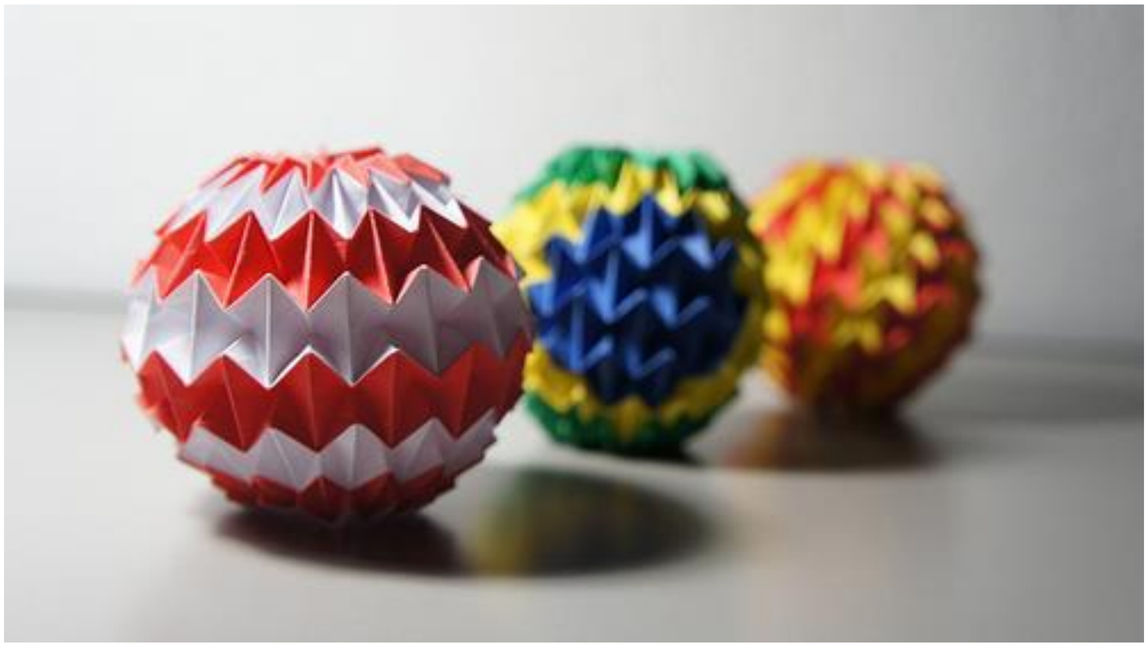


Figure 19. Magic ball origami.

2.2.1.2 Mounting Configuration of the Flexinol

2.2.1.2.1 Version 1

The first approach that we use to mount the Flexinol wire onto the inside of the magic ball is to directly sew two strips of wires on the edges of the ball. We then tape the two short ends together to make it into a ball configuration. We had hoped that when the Flexinol contracts it will minick the action of squeezing the ball at both ends, causing the ball to contract. However the Flexinols are only attached to the ball at the stiches, due to the ridges from the fold, there is not enough tension in the wires to produce notceable motion when put into the ball formation. When it is unfurled we manage to get a very slow breathing rate contraction, minicking a rocking fetal position. These wires are also shortened to increase tension, but again the folds in the origami absorb most of the tension.

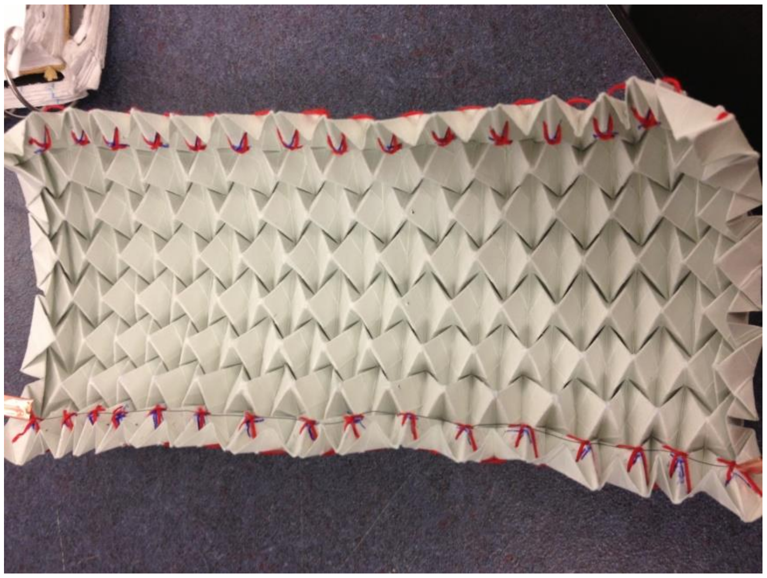


Figure 20. Location of the Flexinol wire sewn onto the edges of the magic ball.

2.2.1.2.2 Version 2

Next we try a zigzag pattern by sewing the Flexinol onto the surfaces of the ridges instead of just the peaks. Due to friction between the paper and the Flexinol wire and the large number of bends, there is no observable motion but creaking noises is heard.

2.2.1.2.3 Version 3

The third approach that we try is to suspend the Flexinol wires from one edge of the origami to the opposite edge. The origami has a tendency to unfurl itself, so we hope that would result in enough tension in the Flexinol. The result is that the paper itself does not budge; all that is observed is the Flexinol withering in midair. We’ve tried this method with both the long and short edges.

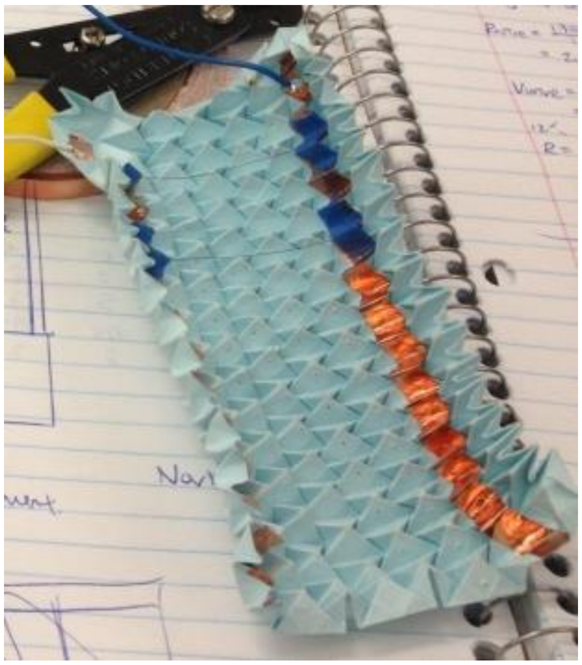


Figure 21. 3rd mounting configuration of the Flexinol wire.

2.2.1.2.4 Version 4

We decide to discard the idea of directly mounting the Flexinol wire onto the magic ball because the length of the Flexinol wire is restricted by the size of the origami paper. Hence we design an internal paper object that would act like a mechanical spring and pull at the magic ball from the inside. The object can be designed and bend into any shape, which would allow room to attach longer Flexinol wires.

The figure below displays our mechanical spring. The spring would be attached to the magic ball via the top and bottom cones and the Flexinol would be sewn to the 8 arms. The paper arms are not folded, so when compressed, the arms would push to resist the compression. However we only attach the Flexinol at the bends, so again we observe the same behavior from version 3 the wires are just withering in space.

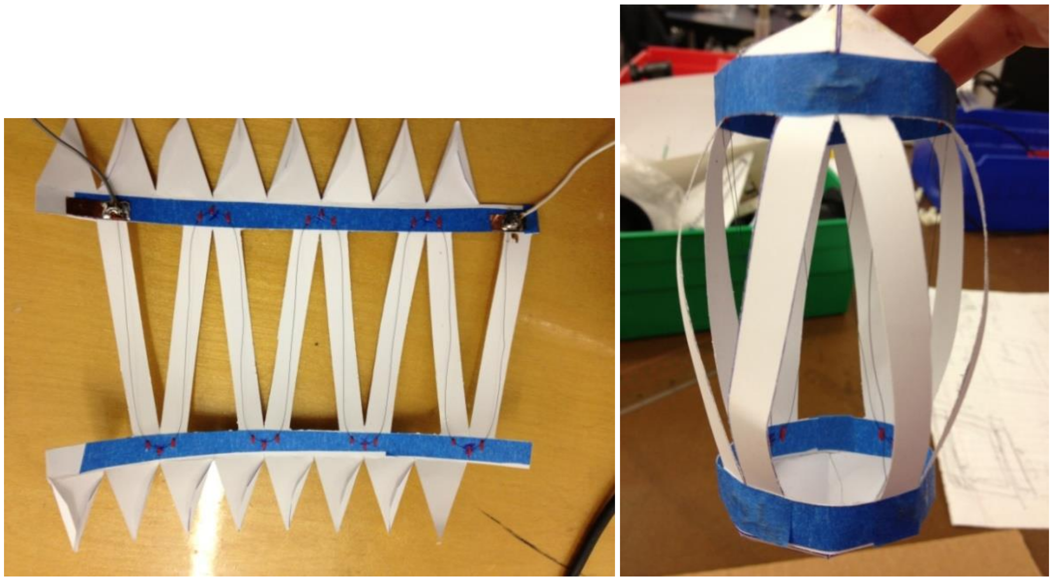


Figure 22. 4th mounting configuration of the Flexinol wire. The unfurled version the left and the furled version on the right.

2.2.1.2.5 Version 5

We decide to sew the Flexinol onto the arms of the spring as well as the bends, so instead of withering, the Flexinol would actually pull on and bend the paper arms. The compression of the spring is around 2cm in the vertical direction. So we put the spring into the magic ball. Due to the additional weight and the folds in the magic ball which resist compression, all we could observe is a tiny quiver.

We then produce a series of different versions of the spring in the hope of improving the design, such as having thinner arms to elevate tension, more arms for longer Flexinol, tighter stitches to attach the Flexinol more closes to the paper. All the versions work well by themselves (the greatest vertical displacement we got was around 8cm), but once they are put inside the magic ball, there just is not enough pulling force to produce very noticeable contractions.

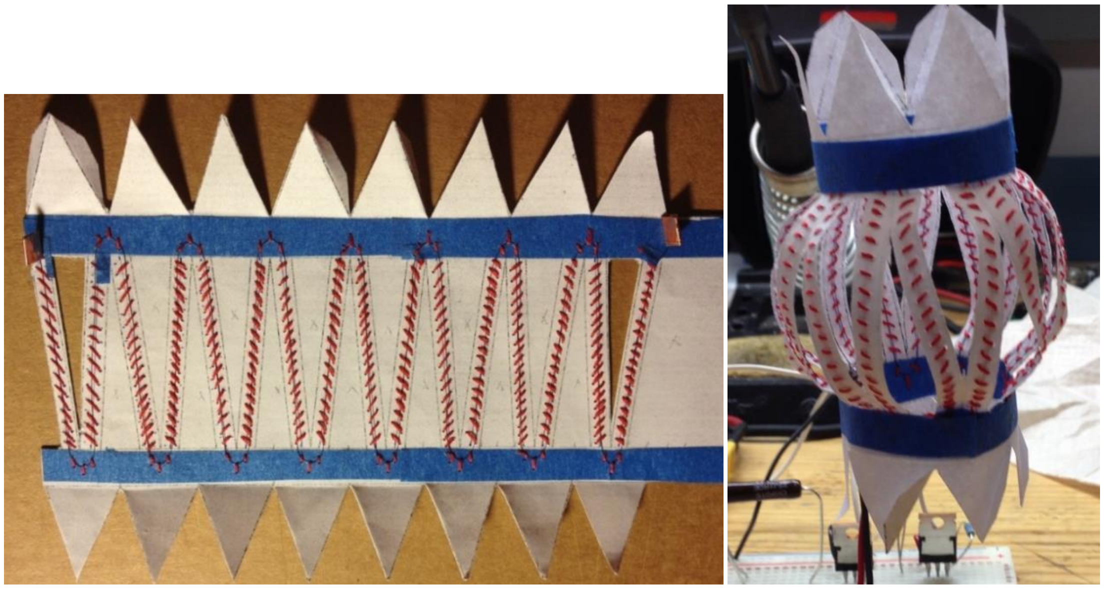


Figure 23. Spring with thinner arms, more arms and tighter stitches.

2.2.1.2.6 Version 6

The last version we tested is based on the first Flexinol test that we did for a flapping crane. The wings are just a flat piece of paper with a “U” shape Flexinol stitching pattern. This design can produce a huge displacement, where the paper can actually bend around 150 degrees. We replicate this design with an “M” shape stitching pattern and put three pieces of Flexinol in parallel to increase its pulling force. When placed into the magic ball, this design has produced the largest total displacement of 2cm. However due to the number of Flexinol used and the fact that it is inside the magic ball, the cooling time for the Flexinol was around 10 to 15 seconds. Considering the low repetition rate and minimal contraction, we decide to not use the magic ball in our project.

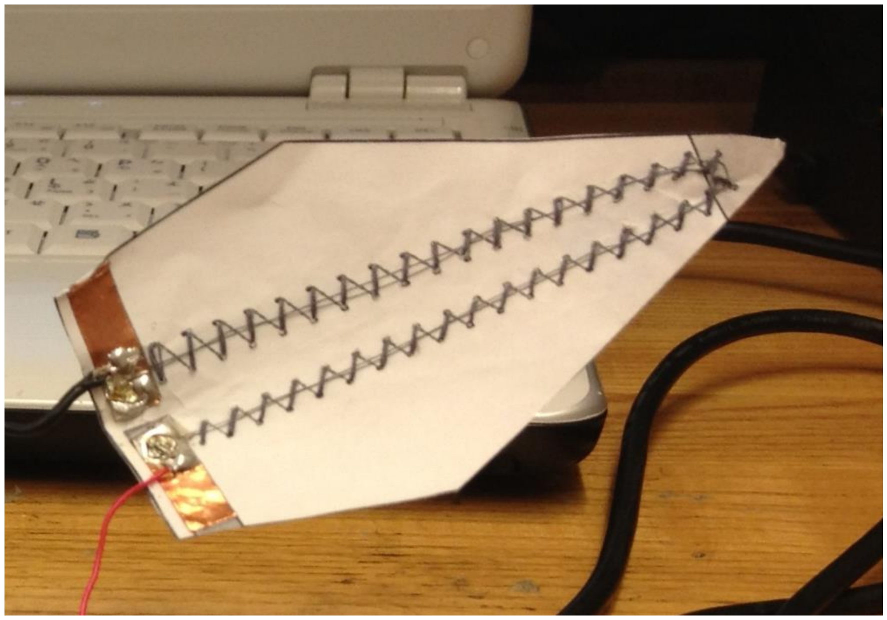


Figure 24. Crane wings with Flexinol wires.

2.3 Dragon

The designs of the origami dragon will be discussed here.

2.3.1 Origami Design

The origami pattern is from the following website http://www.youtube.com/watch?v=oR10rGLzXp4&feature=watch_response.

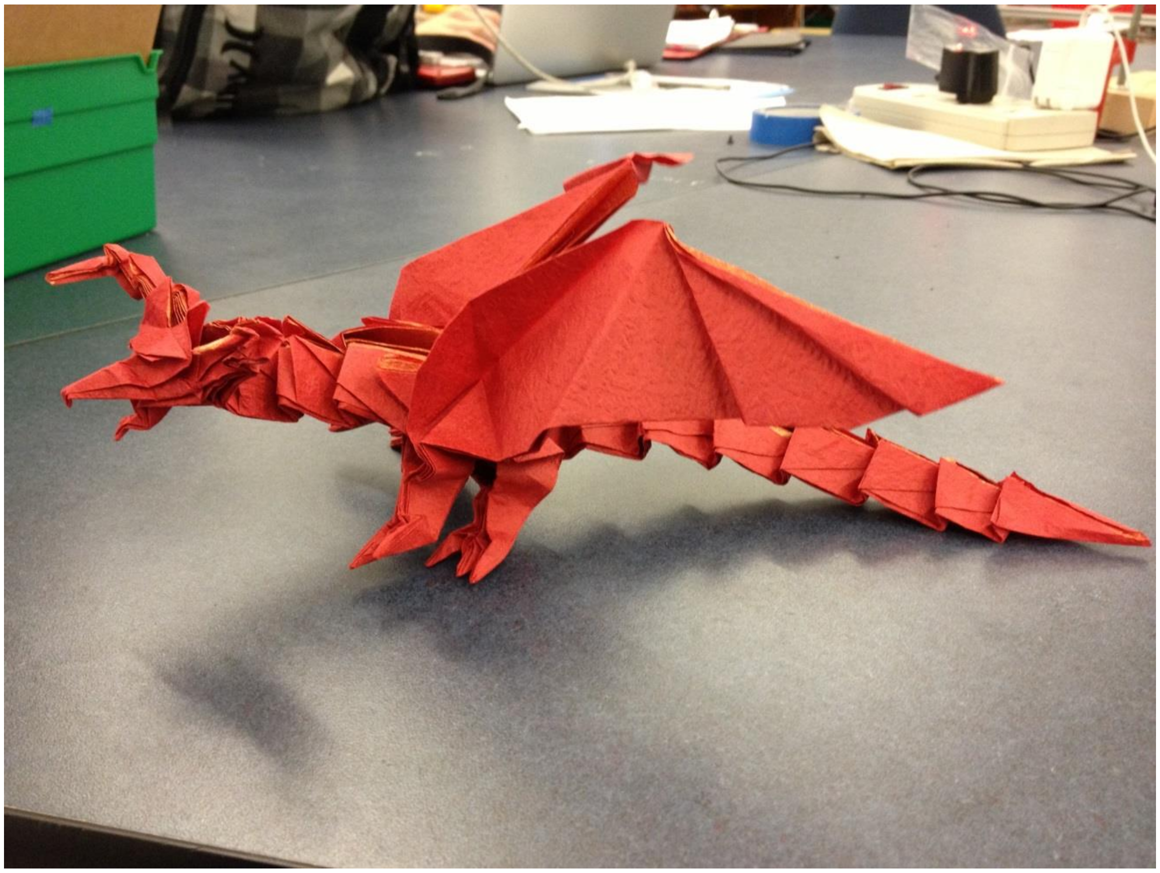


Figure 25. Side view of the dragon origami.

2.3.2 *ALGORITHM*

The dragon is mounted with LEDs and sensors, as shown in Figure 26. A Lilypad light sensor is mounted at the back of the dragon to measure the ambient luminosity. A self-designed capacitive touch sensor, same as the one used in the paper piano, is placed at the claw of the dragon to detect skin touch. A sound-sensor is also used. As decorations, three green LEDs are mounted on the back of the dragon and one red LED is hidden inside its mouth. Two pieces of Flexinol wires are sewed to the dragon to provide the motion for its tail and wings.

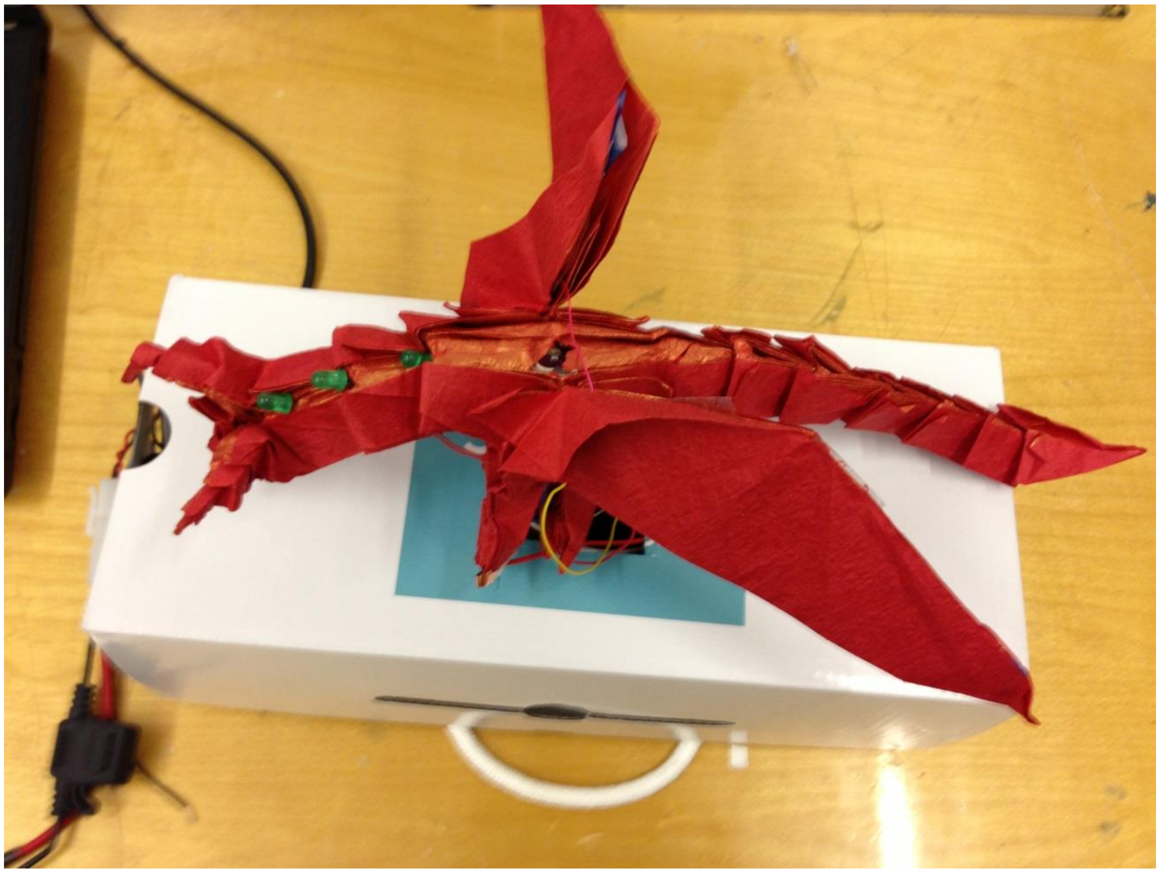


Figure 26. Top of the dragon with LEDs and sensors.

Once powered on, sensors will keep sensing the environment. If the sound sensor is triggered, the dragon is in sleepy state and all three LEDs will shine in turn and repeat for 5 times then the dragon is asleep. If the light sensor reads an increase in luminosity of more than 100%, the dragon is woken and will be in the "active mode". A square PWM current with a peak current of 0.4 amp will be sent through Flexinol in the tail which contracts and curls the tail. A speaker will play Amazing Grace and both red and green LEDs will shine to the beat of the music. If the touch sensor is activated, the dragon is in excited state. Its wings will flap and Jingle Bell will be played. LEDs will shine to the beat of the tune. A flow chart is provided in Figure 27. The board connection is shown in Figure 28.

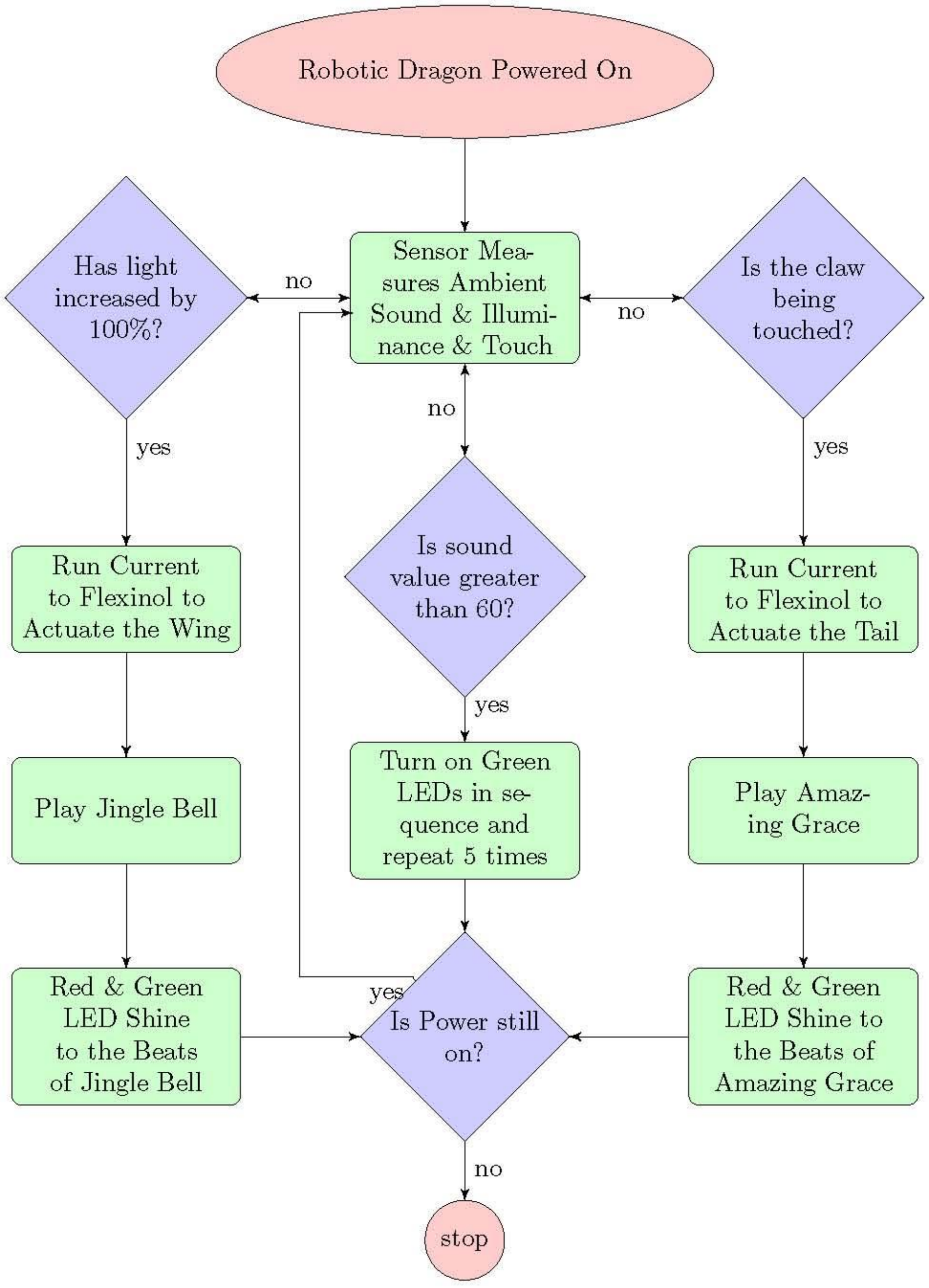

Robotic Dragon Powered On
Sensor Measures Ambient Sound & Illuminance & Touch
no
Has light increased by 100%?
no
Is the claw being touched?
no
yes
Is sound value greater than 60?
yes
Run Current to Flexinol to Actuate the Wing
Run Current to Flexinol to Actuate the Tail
yes
Turn on Green LEDs in sequence and repeat 5 times
Play Jingle Bell
Play Amazing Grace
Red & Green LED Shine to the Beats of Jingle Bell
yes
Is Power still on?
Red & Green LED Shine to the Beats of Amazing Grace
no
stop

Figure 27. A flow chart for the robotic dragon.

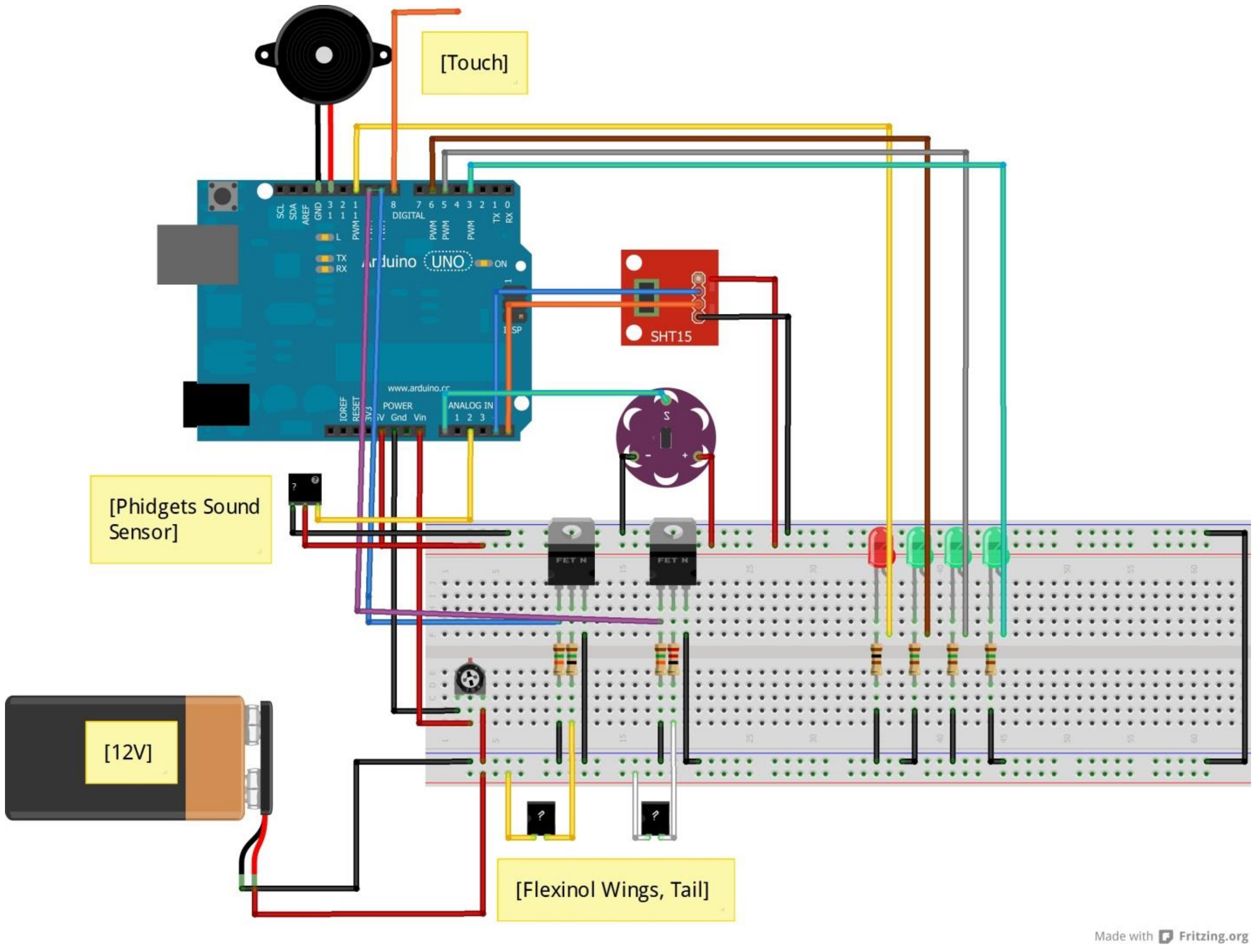


Figure 28. Electronic board and connection for the dragon.

2.3.3 *TEST RESULTS*

The dragon functions as expected. The current is chosen to be 0.4A because smaller current will not provide enough contraction for the dragon and a larger current may reduce the life time of the Flexinol and possibly burn the paper. By trial and error, the optimal on-time for the Pulse Width Modulation (PWM) current is 2 second and the off-time is 2 second.

2.4 INTERACTIVE FINGERS

The designs for the interactive fingers will discussed here

2.4.1 *ORIGAMI DESIGN*

Design is produced via tracing a hand's outline over two pieces of paper. The four fingers and the thumb all have a piece of Flexinol sew onto the surface. The palm has three touch sensors and an LED.

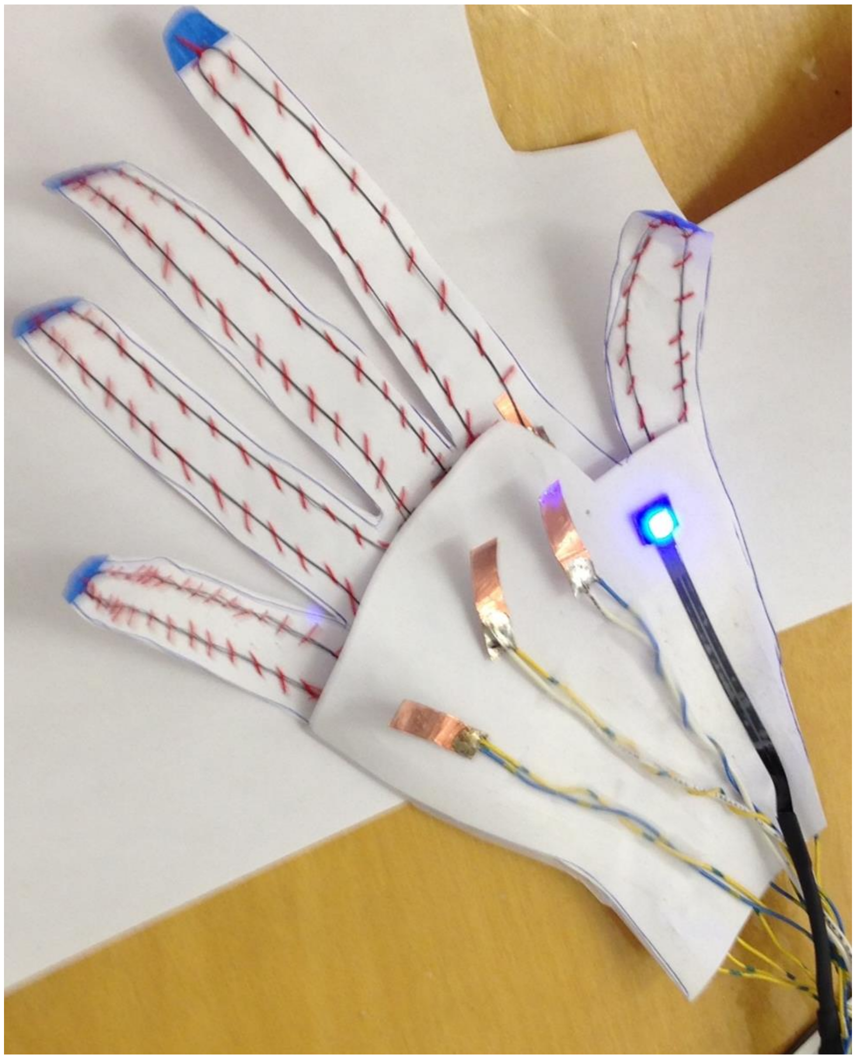


Figure 29. Interactive fingers.

2.4.2 *Software Algorithm*

The code implemented is in the appendix, to use this piece of code, ensure that the CapitiveSense Library has been installed. The library can be found at http://playground.arduino.cc/Main/CapacitiveSensor?from=Main.CapSense.

The LED on the palm turns on when the Arduino is ready to accept a command and it will turn off when one of the touch sensors is activated. When the left sensor is activated, the fingers will flex one by one from pinky to thumb. The activation time for each finger is 1sec and after all the fingers have flexed, there is a cool down time of 5sec. After 5 seconds has passed the LED will turn on again, indicating it's ready to take another command. The activation and cool down time for the fingers are different from the dragons, because there are three pieces of Flexinol in parallel for each finger instead of just one for the dragon's tail and wings. When the right sensor is activated, the fingers will flex from thumb to pinky with the same activation and cool down time. When the middle sensor is activated, all the fingers will flex at the same time with the same activation and cool down time.

2.4.3 Electrical End

There are three capacitive sensors on the palm. The capacitive sensor library measures the time it takes for the voltage to charge on the input pin. When the user touches the copper touchpad, it takes longer for the voltage to charge and the program returns a larger number. If the resistance between the input and the sensor pin is big enough (i.e. 30Mohm), there is a larger chance for the program to detect the signal just by being in close proximity to the copper touchpad. However the large resistor will cause the program to take longer to respond. The input pin has a 1kohm resistor to prevent electrostatic discharge, putting a piece of clear plastic tape over the copper pad will also prevent this, but this lowers the sensitivity. Grounding the sensor pin with small capacitors ranged from 100pF to 0.1µF will improve stability. This program does not utilize the proximity factor of the sensors due to interference. There are copper tapes running underneath the palm to provide current to the Flexinol in the fingers. The sensors are more than sensitive enough to pick up on these electric discharges, thus making the proximity sensors useless.

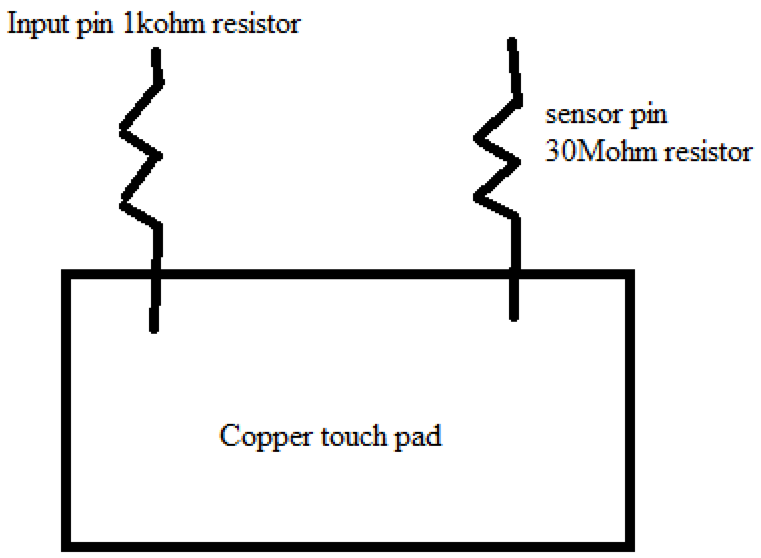


Figure 30. Capacitive sensor diagram, input pin and sensor pin to Arduino digital pins.

The Flexinol in the fingers uses 5 sets of the circuit shown in Figure 30. The battery used is 12V. The LED is a surface mount LED which comes with its own resistor, thus it only requires a common ground and 5V output from the digital pins on the Arduino.

2.4.4 *Test Results*

The original plan was to put this circuit into an origami hand. However the origami hand has too many layers and internal supports which prevent it from bending. The sensors are hyper sensitive, it is possible to put up to 5 layers of paper between the copper touch pads and the user's fingers and still activate the sensors (the code need to be rewritten for a lower threshold). It is also possible to see the surface mount LED through one layer of white construction paper. The flexing fingers are somewhat disappointing. Due to all the fidgeting with the wires and threads, the paper have gotten quite soft and limp, therefore the contraction is rather weak and the fingers bend in strange directions.

3. Conclusion

In this robotic origami project, the team investigated into a variety of technologies on printable circuits, including printing conductive ink, drawing circuits with Bare Paint and using copper tape to make simple circuits. All technologies are found to have major drawbacks. The team also studied the behavior of Flexinol and tried different origami designs with Flexinol. In the end the followings are produced: a piece of robotic dragon origami which can sense ambient light, sound and touch and respond by actuating the Flexinol to move its wings and/or tail and playing a piece of tone; interactive fingers embedded with Flexinol and touch sensors; an Origami Christmas tree with a LED driver circuit that is made to demonstrate the conductive ink technology; a touch piano as a demonstration of the hand-drawn circuit made with Bare Paint.

Valuable experiences have been obtained through the work of the team. The robotic origami is believed to be of great interest and potential, and further work should be pursued.

4. Project Deliverables

4.1 List of Deliverables

The table bellow summarizes the list of various Origami designs that are deliverable at the end of the project.

Original Deliverables	Actual Deliverables	Sensing and actuation of the Origami	Reasons for changes
Origami Flower	Dragon	One light sensor (purchased)	While the type of motion being actuated is similar between the dragon and the flower, there is lot more
		One capacitive touch sensor	

		(designed)	functionality in terms of sensing that can be implemented for the dragon.
		One sound sensor (purchased)	
		Flexinol (purchased)	
Origami Butterfly	Interactive Fingers	Three capacitive touch sensors (designed)	The mechanism for the motion of the butterfly is similar to that for the dragon, hence we decided to make fingers that have similar motion but more interactive with the environment compared to Origami butterfly
		Flexinol (purchased)	
Origami Frog	Paper Piano	Eight capacitive touch sensors (designed)	We believe the dragon and the interactive fingers are sufficient to demonstrate sensing and the possible responses that we can obtain out of Origami designs, so instead we would like to make other paper products that can show the potential of paper based circuits
Work on conductive ink	Origami Christmas Tree	Six Surface mount LEDs (purchased)	No change in the original deliverable plan

Table 2. List of Deliverables.

4.2 FINANCIAL SUMMARY

A table of financial summary is provided below. Note that the shipping and handling fees are not included.

Type	Name	Manufacturer/Vendor	Part Number	Qty	Unit cost per sheet (CAD)	Total Cost (US)
Arduino	Arduino Uno USB Microcontroller Rev 3	Robotshop	RB-Ard-34	2	$ 28.99	$ 57.98
Sensors	LilyPad Light Sensor	Sparkfun Electronics	DEV-08464	5	$ 7.95	$ 39.75
	HDJD-S822 Color Sensor Breakout	Sparkfun Electronics	SEN-10904	2	$ 24.95	$ 49.90
	Color Light Sensor Evaluation Board	Sparkfun Electronics	SEN-10701	2	$ 14.95	$ 29.90
	Sound Detector Board	Robotshop	RB-lnx-33	3	$ 7.67	23.01
	Phidgets Sound Sensor	Phidgets	1133	1	$ 35.00	$ 35.00
	Angalog Sound Sensor	DFRobot	DFR0034	3	$ 6.90	$ 20.70
	LilyPad Temperature Sensor	Sparkfun Electronics	DEV-08777	3	$ 4.95	$ 14.85
	Digital Temperature Sensor Breakout-TMP102	Sparkfun Electronics	SEN-09418	3	$ 5.95	$ 17.85
	Humidity and Temeprature Sensor-SHT15 Breakout	Sparkfun Electronics	SEN-08257	2	$ 41.95	$ 83.90
Speakers	*Buzzer-PC Mount 12mm 2.048kHz*	Sparkfun Electronics	COM-07950	4	$ 1.95	$ 7.80
	Thin Speaker	Sparkfun Electronics	COM-10722	4	$ 0.95	$ 3.80
	Speaker -0.5W 8ohm	Sparkfun Electronics	COM-09151	4	$ 1.95	$ 7.80
	Speaker-PCB Mount	Sparkfun Electronics	COM-11089	4	$ 1.95	$ 7.80
DAC	Breakout Board for MCP 4725 I2C DAC	Sparkfun Electronics	BOB-08736	4	$ 4.95	$ 19.80
Flexinol-Related	Flexinol Wire 0.006'' HT	RobotShop	RB-Dyn-18	6	$ 22.50	$ 135.00
	Flexinol Moving Butterfly (Blue Morpho)	RobotShop	RB-Dyn-29	1	$ 19.95	$ 19.95
Ink-Related	Epson WOrkforce 30Printing kit with Metalon JS silver ink and Photopaper	Novacentrix	JS-011 JS-B15P JS-B25P JS-B35P	1	$ 355.00	$ 355.00
	Bare Paint Pen Pack (Pen with Conductive ink)	Bare Conductive	N/A	2	$ 25.00	$ 50.00
LEDs	Pre-Wired Surface Mount 12V LED(Orange)	Oznium	N/A	10	$ 3.29	$ 32.90
	Pre-Wired Surface Mount 12V LED(Blue)	Oznium	N/A	10	$ 3.49	$ 34.90
Origami	VOG Papers PEARL-crumpled (FireRed, 64cm*64cm)	Origami-shop	N/A	1	$ 3.36	$ 3.36
	VOG Papers -crumpled(Blue, 64cm*64cm)	Origami-shop	N/A	1	$ 3.36	$ 3.36
	Gorilla Fur Tissue Paper (Black, 60cm*60cm)	Origami-shop	N/A	1	$ 3.23	$ 3.23
	Natural Tissue Paper (Green, 60cm*60cm)	Origami-shop	N/A	1	$ 3.23	$ 3.23
	Yellow Tissue-foil paper (Yellow, 60cm*60cm)	Origami-shop	N/A	1	$ 4.52	$ 4.52
	Pack Discovery (Package)	Origami-shop	N/A	1	N/A	$ 17.43
	Kraft Paper (Organge, 60cm*60cm)	Kim's Crane Origami Supplies	N/A	1	$ 3.50	$ 3.50
	Kraft Paper (Green, 60cm*60cm)	Kim's Crane Origami Supplies	N/A	1	$ 3.50	$ 3.50
	Kraft Paper (Charcoal, 60cm*60cm)	Kim's Crane Origami Supplies	N/A	1	$ 3.50	$ 3.50
Total						**$1,093.22**

Table 3. A List of Items Purchased.

5. RECOMMENDATIONS

Our recommendations for the project can be mainly categorized into two parts: general feedback on the experiences working on this project, and specific feedback on the Origami designs or components that are used on the designs.

Our general experience when we work on the project is that the project is somewhat open-ended. Because of the open-endedness of the project, the first two and half months are mostly spent on exploring different means of sensing, actuating, and investigating the feasibility of printable paper-based circuits. For future projects, we recommend that the scope of the project is task-specific (e.g. closer study on the Flexinol, standalone experiment on printable paper-based circuits, and focus on one particular piece of robotic origami with more built-in features)

For more specific feedback on the various designs of this project, the table bellow summarizes the problems that we encounter and the future improvements that can be made.

Origami Design	Sensing and actuation of the Origami	Future Improvements or problems encountered
Dragon	Light sensor	All sensors function as expected. Sensors are used to trigger one particular actuation, they are more of "switch" like, the accurate readings of values are not necessary. Hence, simple two-state sensors made from paper are more preferable.
	One capacitive touch sensor	
	Sound sensor	
	Flexinol	
Interactive Fingers	Three capacitive touch sensors	Problems - The method of attaching Flexinol to paper is time consuming and the flex motion varies due to the paper's stiffness and stitches, causing the Flexinol to lose tension. The Flexinol is not strong enough to lift more weight than the paper it's attached to. - Only use the touch capacitors to sense true touch instead of proximity, because the proximity sensor is too sensitive and unreliable. Unless you can isolate the sensors to minimize interference.
	Flexinol	
Paper Piano	Capacitive touch sensors	Problem of loose connections for the drawing on the paper or between the electrical wires and male pins Using conductive pens to draw the circuits can easily cause smudges on the paper
	Speaker	
Christmas Tree	Surface mount LEDs	Problems: Loose connection becomes a problem as some of the tapes come loose When there is loose connection, it is hard to debug which one is loose because everything is hidden inside the tree Recommendations: Use electrical wires that are not as big and thick as the one we are using now so the tape does not come off very easily Use more adhesive tape to help with the connection problem Maybe explore other kinds of substrate that the printer can print on other than plastic
	Conductive Ink	
	Tape	

Table 4. Problems Encountered and Future Improvements of Origami Designs.

APPENDIX A

ARDUINO CODE

CODE FOR THE PAPER PIANO

```
// Pin to connect to your drawing of piano
int pinNoteC5 = 12;
int pinNoteD5 = 11;
int pinNoteE5 = 10;
int pinNoteF5 = 9;
int pinNoteG5 = 5;
int pinNoteA5 = 4;
int pinNoteB5 = 3;

int pinNoteC6 = 2;
//Speaker pin
int pinSpeaker = 13;

// This is how high the sensor needs to read in orderto trigger a touch.  Non touch reads 1 and a
touch reads any value between 2 to 17, usually 2 or 3.This number usually doesn't require to be
changed.
int touchedCutoff = 2;

//length of note played
int noteDuration = 300;

//define music note's digital values
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
```

```
#define NOTE_C6  1047

//#include <Tone.h>  //tone library, used for speaker

void setup(){
Serial.begin(9600);
}

void loop(){
  // If the capacitive sensor reads above a certain threshold,
  //  turn on the LED and play a corresponding note

int temp8=readCapacitivePin(pinNoteC5);
Serial.println("C5 Reading:");
Serial.println(temp8);
  if (temp8 > touchedCutoff) {
    tone(pinSpeaker, NOTE_C5, noteDuration);
Serial.print("c");
  }

  else if (readCapacitivePin(pinNoteD5) >touchedCutoff) {
   tone(pinSpeaker, NOTE_D5, noteDuration);
Serial.println("d");
  }
  else if (readCapacitivePin(pinNoteE5) >touchedCutoff) {
    tone(pinSpeaker, NOTE_E5, noteDuration);
Serial.println("e");
  }
  else if (readCapacitivePin(pinNoteF5) >touchedCutoff) {
   tone(pinSpeaker, NOTE_F5, noteDuration);
Serial.println("f");
  }
  else if (readCapacitivePin(pinNoteG5) >touchedCutoff) {
    tone(pinSpeaker, NOTE_G5, noteDuration);
Serial.println("g");
  }
  else if (readCapacitivePin(pinNoteA5) >touchedCutoff) {
    tone(pinSpeaker, NOTE_A5, noteDuration);
Serial.println("a");
  }
  else if (readCapacitivePin(pinNoteB5) >touchedCutoff) {
    tone(pinSpeaker, NOTE_B5, noteDuration);
Serial.println("b");
  }
 else if (readCapacitivePin(pinNoteC6) >touchedCutoff) {
    tone(pinSpeaker, NOTE_C6, noteDuration);
Serial.println("c");
  }
  else {
noTone(pinSpeaker);

  }
  //delay(1000);
}

//The function below is from an article named Native Capacitive Sensors without additional
Hardware.
//Retrieved from the Arduino official website from
http://playground.arduino.cc/Code/CapacitiveSensor

// readCapacitivePin
//  Input: Arduino pin number
//  Output: A number, from 0 to 17 expressing
//  how much capacitance is on the pin
//  When you touch the pin, or whatever you have
//  attached to it, the number will get higher
#include "pins_arduino.h" // Arduino pre-1.0 needs this
uint8_t readCapacitivePin(intpinToMeasure) {
  // Variables used to translate from Arduino to AVR pin naming
  volatile uint8_t* port;
  volatile uint8_t* ddr;
  volatile uint8_t* pin;
  // Here we translate the input pin number from
  //  Arduino pin number to the AVR PORT, PIN, DDR,
  //  and which bit of those registers we care about.
  byte bitmask;
  port = portOutputRegister(digitalPinToPort(pinToMeasure));
ddr = portModeRegister(digitalPinToPort(pinToMeasure));
  bitmask = digitalPinToBitMask(pinToMeasure);
  pin = portInputRegister(digitalPinToPort(pinToMeasure));
  // Discharge the pin first by setting it low and output
  *port &= ~(bitmask);
  *ddr  |= bitmask;
  delay(1);
  // Make the pin an input with the internal pull-up on
  *ddr&= ~(bitmask);
  *port |= bitmask;

  // Now see how long the pin to get pulled up. This manual unrolling of the loop
  // decreases the number of hardware cycles between each read of the pin,
  // thus increasing sensitivity.
  uint8_t cycles = 17;
       if (*pin & bitmask) { cycles =  0;}
  else if (*pin & bitmask) { cycles =  1;}
  else if (*pin & bitmask) { cycles =  2;}
  else if (*pin & bitmask) { cycles =  3;}
  else if (*pin & bitmask) { cycles =  4;}
  else if (*pin & bitmask) { cycles =  5;}
  else if (*pin & bitmask) { cycles =  6;}
  else if (*pin & bitmask) { cycles =  7;}
  else if (*pin & bitmask) { cycles =  8;}
  else if (*pin & bitmask) { cycles =  9;}
  else if (*pin & bitmask) { cycles = 10;}
  else if (*pin & bitmask) { cycles = 11;}
  else if (*pin & bitmask) { cycles = 12;}
  else if (*pin & bitmask) { cycles = 13;}
  else if (*pin & bitmask) { cycles = 14;}
  else if (*pin & bitmask) { cycles = 15;}
  else if (*pin & bitmask) { cycles = 16;}

  // Discharge the pin again by setting it low and output
  //  It's important to leave the pins low if you want to
  //  be able to touch more than 1 sensor at a time - if
  //  the sensor is left pulled high, when you touch
  //  two sensors, your body will transfer the charge between
  //  sensors.
  *port &= ~(bitmask);
  *ddr  |= bitmask;
  return cycles;
}
```

Code for the Dragon Control System

```
// Code for controling the Origami System
// Author: Vicky Wang, Jason Niu

//Sensors used: Lilypad Light Sensor; Phidgets Sound Sensor; Self-made capacitive touch sensor
//Actuations: Speaker, LED, PWM for Flexinol

//Jan.4th, 2013
/*
 State I (sleepy state): Sound sensor input (pin A2)
 LEDs - blue ones on
 Speaker - N/A
Flexinol - N/A

 State II (active state): Light input increase by 100% (pin2)
 LEDs - green ones on and shine to the beats of music
 Speaker - Amazing Grace
Flexinol - Run PWM current for Flexinol pin 9 to actuate tail

 State III (excited state): Dragon touched        (Sensor connected to Digital pin 8)
 LEDS - all LEDs on and shine to the beats of music
 Speaker - Jingle Bell
Flexinol - Run PWM current for Flexinol pin 10 to actuate wing

*/

//Pitch and frequency lookup table
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
```

```
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
```

```
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978

#include <Wire.h>
#include <math.h>
//#include "pitches.h"

voidStateI();
voidStateII();
voidStateIII();
floatgetTemperature();
void Speaker(int);
voidTwinkileStar();
voidAmazingGrace();
void Moonlight();
void RLED(intRLEDState);  //Flips the Red LED output; if RLEDState==1, turn on the RLED
and change the RLEDState to 0
void GLED(intGLEDState);
void BLED(intBLEDState);

int tmp102Address = 0x48;
// intledPin = 13;         // LED is connected to digital pin 13
```

```
 //Analog pin connections
intDigPin_lightSen = 1;
intDigPin_TempSen = 2;
intDigPin_SoundSen = 3;

//Light Sensor

intPinlplsensor = 0;     // light sensor is connected to analog pin A0
intlplsensorValue=0;  // variable to store the value coming from the sensor
//Touch Sensor
intPintouchsensor=8;   //A touch sensor is connected to digital pin 8 on the board
//Sound Sensor
intpinSoundS = 2;      //A sound sensor is connected to pin A2 on the board
intSoundSValue=0;
intSoundSValuebase=0;

//LEDs
int pinGLED3=6; //1st Green LED is connected to pin 11 (PWM)
int pinGLED2=5; //2nd Green LED is connected to pin 10 (PWM)
intpinGLED=3; //3rd Green LED is connected to pin 9 (PWM)
intpinRLED=11; //RED LED is connected to pin 6 (PWM)


intRLEDState=0; //0 means Red LED off; 1 means on
intGLEDState=0; //0 means Green LED off; 1 means on
intBLEDState=0; //0 means Blue LED off; 1 means on

intpinSpeaker = 13;    //Speaker

//Flexinol

intPinFlexinol = 10;    //Tail
int PinFlexinol2 = 9;    //Wings

intFlexinolHighWings = 2000; //in milliseconds
intFlexinolLowWings = 2000; //in milliseconds
intFlexinolHighTail = 2000; //in milliseconds
intFlexinolLowTail = 2000; //in milliseconds

//Threshold values that may need to be changed
inttouchsensorcutoff=1;
intlightsensordark=20;  //use this value if want to have the light sensor trigger the function if
sensor is covered
intlightsensorbright=300; //use this value if want to have the light sensor trigger the function if
sensor is exposed to bright light
intlplsensorbase = 50;
```

```
void setup() //It's used to initialize variables, pin modes,start using libraries
{
Wire.begin();
  //Initializing Digital Inputs
pinMode(DigPin_lightSen, INPUT);
pinMode(DigPin_TempSen, INPUT);
pinMode(DigPin_SoundSen, INPUT);

  //Initializing Digital Outputs
pinMode(PinFlexinol, OUTPUT);
pinMode(PinFlexinol2, OUTPUT);
pinMode(pinSpeaker, OUTPUT);
  //pinMode(pinLED, OUTPUT);     //LED to signal Morse Code
 // pinMode(ledPin, OUTPUT);        // sets the ledPin to be an output
 // pinMode(digPin, OUTPUT);    // pin as output

pinMode(pinRLED, OUTPUT);
pinMode(pinGLED, OUTPUT);
pinMode(pinGLED2, OUTPUT);
pinMode(pinGLED3, OUTPUT);

  //Set the data rate for serial data transmission
Serial.begin(9600);

 // digitalWrite(ledPin, HIGH);         // turn the LED on

SoundSValuebase=analogRead(pinSoundS);   //connect mic sensor to Analog 2
lplsensorbase = analogRead(Pinlplsensor);   // read a base value from the sensor
}

void loop()
{
  //Phidget Sound Sensor, connected to pin A2, turns dragon to sleep state
        SoundSValue=analogRead(pinSoundS);   //connect mic sensor to Analog 0
        intdecible=0;
        decible=16.801*log(SoundSValue)+9.872;
        Serial.println("Sound Value Baseline");
        Serial.println(SoundSValuebase);
        Serial.println("SoundValue");
        Serial.println(SoundSValue,DEC);//print the sound value to serial
          //Serial.println(decible,DEC);//print the sound value to serial
        Serial.println(' ');//
        if(SoundSValue>=60)
          {
                StateI();
```

```
            }


   //Get Lylipad Light Sensor Reading
   //digitalWrite(digPin, HIGH); // sets the pin HIGH
lplsensorValue = analogRead(Pinlplsensor); // read the value from the sensor
Serial.println("lightsensorValue");
Serial.println(lplsensorValue);
Serial.println(' ');
   /* Enable the flowing if activing function by cover light sensor is preferred
if(lplsensorValue<lightsensordark)
   {
StateII();
   }
   */

   //The following code actives function when the a sudden bright light shines on the light sensor
if (lplsensorbase<100)
   {
         lplsensorbase=100;
   }
if((lplsensorValue>lightsensorbright) || lplsensorbase*2)
   {
StateII();
   }

   //Get Capactive touch sensor reading
inttouchsensorValue = readCapacitivePin(Pintouchsensor);
Serial.println("Touch Sensor Value:");
Serial.println(touchsensorValue);
if(touchsensorValue>touchsensorcutoff)  //The dragon is touched by someone!
   {
StateIII();
   }



   /* A temperature sensor functions like a touch sensor in our case and is now replaced by the
self-designed capacitive touch sensor
   //TMP Temperatuer Sensor
floatcelsius = getTemperature();
Serial.print("Celsius: ");
Serial.println(celsius);
if(celsius>26)
   {
StateIII();
```

```
   }
   */


 }


floatgetTemperature(){
Wire.requestFrom(tmp102Address,2);

byte MSB = Wire.read();
byte LSB = Wire.read();

  //it's a 12bit int, using two's compliment for negative
intTemperatureSum = ((MSB << 8) | LSB) >> 4;

floatcelsius = TemperatureSum*0.0625;
returncelsius;
}

voidStateI()
{
for (inti=0; i<6;i++)
  {
  // Turn LEDs all on
  /*
  {
sLED(1);
delay(200);
sLED(0);
delay(500);
  */
  //Alternatively, turn the LEDs one by one
digitalWrite(pinGLED,HIGH);
delay(300);
digitalWrite(pinGLED,LOW);
digitalWrite(pinGLED2,HIGH);
delay(300);
digitalWrite(pinGLED2,LOW);
digitalWrite(pinGLED3,HIGH);
delay(300);
digitalWrite(pinGLED3,LOW);

delay(300);
  }
}
```

```
voidStateII()   //Actuate the tail
{

  //BLED(1);
Flexinol(1);  //Actuate the tails
  //BLED(0);
  //speaker();
AmazingGrace();
 // TwinkileStar();
Flexinol(1);
}

voidStateIII()  //Touched
{

  //GLED(1);
Flexinol(2);  //Actuate the wings
  //GLED(0);
  // speaker();
JingleBells();
   //MorseCodeJingleBells();
Flexinol(2);
  //MorseCodeAmazingGrace();
}

voidsLED(intLEDState)
{
if (LEDState==1)
  {

           //RLEDState=0;
           //delay(1000);        //Test
           digitalWrite(pinRLED,HIGH);
           digitalWrite(pinGLED,HIGH);
           digitalWrite(pinGLED2,HIGH);
           digitalWrite(pinGLED3,HIGH);
           //analogWrite(pinBLED,1);
  }
if (LEDState==0)
  {
           //RLEDState=1;
           digitalWrite(pinRLED,LOW);
           digitalWrite(pinGLED,LOW);
           digitalWrite(pinGLED2,LOW);
           digitalWrite(pinGLED3,LOW);
```

```
        //analogWrite(pinBLED,0);
  }
}

void RLED(intRLEDState)
{
if (RLEDState==1)
  {
        analogWrite(pinRLED,128);
        RLEDState=0;
        delay(1000);       //Test
        analogWrite(pinRLED,0);
  }
else if (RLEDState==0)
  {
        RLEDState=1;
  }
}

void GLED(intGLEDState)
{
if (GLEDState==1)
  {
        analogWrite(pinGLED,128);
        GLEDState=0;
        delay(1000);
        analogWrite(pinGLED,0);
  }
else if (GLEDState==0)
  {
        GLEDState=1;
  }
}

/*
void BLED(intBLEDState)
{
if (BLEDState==1)
  {
        analogWrite(pinBLED,128);
        BLEDState=0;
        delay(1000);
        analogWrite(pinBLED,0);
  }
if (BLEDState==0)
  {
          BLEDState=1;
  }
}
*/

//Run different PWM currents to Flexinol for different actuation freq and strengths
voidFlexinol(int state)
{
switch (state)
  {
case 1:
digitalWrite(PinFlexinol, HIGH);   // sets the pin HIGH
delay(FlexinolHighTail);                  // PWM high state
      //digitalWrite(ledPin, HIGH);    // turn the LED on as an indicator
digitalWrite(PinFlexinol, LOW);
delay(FlexinolLowTail);
break;

case 2:
          digitalWrite(PinFlexinol2, HIGH);   // sets the pin HIGH  for tail
delay(FlexinolHighWings);                  // PWM high state
      //digitalWrite(ledPin, HIGH);    // turn the LED on as an indicator
digitalWrite(PinFlexinol2, LOW);
delay(FlexinolLowWings);

          break;

  }
}

//Below are music section. They are based on the Arduino built-in example "ToneMelody"
voidTwinkileStar()
{
  //MorseCodeTwinkleStar();
int melody[] = {
          NOTE_C4, NOTE_C4, NOTE_G4, NOTE_G4,
          NOTE_A4, NOTE_A4, NOTE_G4,
          NOTE_F4, NOTE_F4, NOTE_E4, NOTE_E4,
          NOTE_D4, NOTE_D4, NOTE_C4,
          NOTE_G4, NOTE_G4, NOTE_F4, NOTE_F4,
          NOTE_E4, NOTE_E4, NOTE_D4,
          NOTE_G4, NOTE_G4, NOTE_F4, NOTE_F4,
          NOTE_E4, NOTE_E4, NOTE_D4,
          NOTE_C4, NOTE_C4, NOTE_G4, NOTE_G4,
          NOTE_A4, NOTE_A4, NOTE_G4,
          NOTE_F4, NOTE_F4, NOTE_E4, NOTE_E4,
            NOTE_D4, NOTE_D4, NOTE_C4,
            };

// note durations: 4 = quarter note, 8 = eighth note, etc.:
intnoteDurations[] = {
            2, 2, 2, 2,
            2, 2, 1,
            2, 2, 2, 2,
            2, 2, 1,
            2, 2, 2, 2,
            2, 2, 1,
            2, 2, 2, 2,
            2, 2, 1,
            2, 2, 2, 2,
            2, 2, 1,
            2, 2, 2, 2,
            2, 2, 1,
            };

  // iterate over the notes of the melody:
for (intthisNote = 0; thisNote< 42; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
intnoteDuration = 1000/noteDurations[thisNote];
sLED(1);
tone(pinSpeaker, melody[thisNote],noteDuration);

    //delay(noteDuration);
    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
intpauseBetweenNotes = noteDuration * 1.30;

delay(noteDuration);
sLED(0);
delay(pauseBetweenNotes-noteDuration);
    // stop the tone playing:
noTone(pinSpeaker);
    //sLED(0);
  }
}

voidAmazingGrace()
{
  //MorseCodeAmazingGrace();
```

```
int melody[] = {
	NOTE_D4, NOTE_G4, NOTE_B4, NOTE_G4, NOTE_B4,
	NOTE_A4, NOTE_G4, NOTE_E4, NOTE_D4,
	NOTE_D4, NOTE_G4, NOTE_B4, NOTE_G4, NOTE_B4,
	NOTE_A4, NOTE_D5,
	NOTE_B4, NOTE_D5, NOTE_D5,NOTE_B4, NOTE_G4,
	NOTE_D4, NOTE_E4, NOTE_G4, NOTE_E4, NOTE_D4,
	NOTE_D4, NOTE_G4, NOTE_B4, NOTE_G4, NOTE_B4, NOTE_A4, NOTE_G4
  };

// note durations: 4 = quarter note, 8 = eighth note, etc.:
doublenoteDurations[] = {
	2, 1, 4, 4, 1,
	2, 1, 2, 1,
	2, 1, 4, 4, 1,
	2, 0.5,
	2, 1, 4, 4, 1,
	2, 1, 4, 4, 1,
	2, 1, 4, 4, 1, 2, 0.5
  };

  // iterate over the notes of the melody:
for (intthisNote = 0; thisNote< 33; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
intnoteDuration = 1000/noteDurations[thisNote];
sLED(1);
tone(pinSpeaker, melody[thisNote],noteDuration);

    //delay(noteDuration);
    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
intpauseBetweenNotes = noteDuration * 1.30;

delay(noteDuration);
sLED(0);
delay(pauseBetweenNotes-noteDuration);
    // stop the tone playing:
noTone(pinSpeaker);
    //sLED(0);
  }

}
```

```
voidJingleBells()
{
  //MorseCodeJingleBells();
int melody[] = {
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_G5, NOTE_C5, NOTE_D5, NOTE_E5,
  NOTE_F5, NOTE_F5, NOTE_F5, NOTE_F5,
  NOTE_F5, NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_D5, NOTE_D5, NOTE_E5,
  NOTE_D5, NOTE_G5,

  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_E5, NOTE_G5, NOTE_C5, NOTE_D5, NOTE_E5,
  NOTE_F5, NOTE_F5, NOTE_F5, NOTE_F5,
  NOTE_F5, NOTE_E5, NOTE_E5, NOTE_E5,
  NOTE_G5, NOTE_G5, NOTE_F5, NOTE_D5, NOTE_C5
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
intnoteDurations[] = {
4, 4, 2,
4, 4, 2,
4, 4, 4, 4, 1,
4, 4, 4, 4,
4, 4, 4, 4,
4, 4, 4, 4,
2, 2,

4, 4, 2,
4, 4, 2,
4, 4, 4, 4, 1,
4, 4, 4 ,4,
4, 4, 4, 4,
4, 4, 4, 4, 1
};

  // iterate over the notes of the melody:
for (intthisNote = 0; thisNote< 50; thisNote++) {

   // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
intnoteDuration = 1000/noteDurations[thisNote];
sLED(1);
tone(pinSpeaker, melody[thisNote],noteDuration);

   //delay(noteDuration);
   // to distinguish the notes, set a minimum time between them.
   // the note's duration + 30% seems to work well:
intpauseBetweenNotes = noteDuration * 1.30;

delay(noteDuration);
sLED(0);
delay(pauseBetweenNotes-noteDuration);
   // stop the tone playing:
noTone(pinSpeaker);
   //sLED(0);
   }
}

void Moonlight()
{
  //MorseCodeMoonlight();

int melody[] = {
  //NOTE_C4, NOTE_G3,NOTE_G3, NOTE_A3, NOTE_G3,0, NOTE_B3, NOTE_C4};
  //Test with Moonlight Sonata
  //Replaced BS4 with B4
   NOTE_GS3,NOTE_CS4,NOTE_E4,
NOTE_GS3,NOTE_CS4,NOTE_E4,NOTE_GS3,NOTE_CS4,NOTE_E4,NOTE_GS3,NOTE_C
S4,NOTE_E4, //parallel
   NOTE_GS3,NOTE_CS4,NOTE_E4,
NOTE_GS3,NOTE_CS4,NOTE_E4,NOTE_GS3,NOTE_CS4,NOTE_E4,NOTE_GS3,NOTE_C
S4,NOTE_E4, //parallel

NOTE_A4,NOTE_CS4,NOTE_E4,NOTE_A4,NOTE_CS4,NOTE_E4,NOTE_A4,NOTE_D4,N
OTE_FS4,NOTE_A4,NOTE_D4,NOTE_FS4, //parallel

NOTE_GS3,NOTE_B4,NOTE_FS4,NOTE_GS3,NOTE_CS4,NOTE_E4,NOTE_GS3,NOTE_C
S4,NOTE_E4,NOTE_GS3,NOTE_CS4,NOTE_DS4,NOTE_FS3,NOTE_B4,NOTE_DS4,
//parallel

NOTE_E3,NOTE_GS3,NOTE_CS4,NOTE_GS3,NOTE_CS4,NOTE_E4,NOTE_GS3,NOTE_C
S4,NOTE_E4,NOTE_GS3,NOTE_CS4,NOTE_E4,

NOTE_FS3,NOTE_DS4,NOTE_FS4,NOTE_FS3,NOTE_DS4,NOTE_FS4,NOTE_FS3,NOTE_
DS4,NOTE_FS4,NOTE_FS3,NOTE_DS4,NOTE_FS4,
NOTE_FS3,NOTE_CS4,NOTE_E4,NOTE_FS3,NOTE_CS4,NOTE_E4,NOTE_FS3,NOTE_CS
4,NOTE_FS4,NOTE_FS3,NOTE_CS4,NOTE_FS4,

NOTE_FS3,NOTE_GS4,NOTE_E4,NOTE_FS3,NOTE_GS4,NOTE_E4,NOTE_FS3,NOTE_GS
4,NOTE_DS4,NOTE_FS3,NOTE_GS4,NOTE_DS4,

NOTE_FS3,NOTE_GS4,NOTE_E4,NOTE_FS3,NOTE_GS4,NOTE_E4,NOTE_FS3,NOTE_GS
4,NOTE_E4,NOTE_FS3,NOTE_GS4,NOTE_E4,

NOTE_F3,NOTE_GS4,NOTE_E4,NOTE_F3,NOTE_GS4,NOTE_E4,NOTE_F3,NOTE_GS4,N
OTE_E4,NOTE_F3,NOTE_GS4,NOTE_E4,

NOTE_F3,NOTE_GS4,NOTE_F4,NOTE_F3,NOTE_GS4,NOTE_F4,NOTE_F3,NOTE_GS4,N
OTE_F4,NOTE_F3,NOTE_GS4,NOTE_F4};

  // note durations: 4 = quarter note, 8 = eighth note, etc.:
intnoteDurations[] = {
        //  4, 8, 8, 4,4,4,4,4 };
  //Test
        4,4,4,4,4,4,4,4,4,4,4,4,
        4,4,4,4,4,4,4,4,4,4,4,4,
        4,4,4,4,4,4,4,4,4,4,4,4,
        4,4,4,4,4,4,4,4,4,4,4,4,
        4,4,4,4,4,4,4,4,4,4,4,4,
        4,4,4,4,4,4,4,4,4,4,4,4,
        4,4,4,4,4,4,4,4,4,4,4,4,
        4,4,4,4,4,4,4,4,4,4,4,4,
        4,4,4,4,4,4,4,4,4,4,4,4,
        4,4,4,4,4,4,4,4,4,4,4,4,
        4,4,4,4,4,4,4,4,4,4,4,4};

  // iterate over the notes of the melody:
for (intthisNote = 0; thisNote< 132; thisNote++)
  {
        // to calculate the note duration, take one second
     // divided by the note type.
     //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
        intnoteDuration = 1000/noteDurations[thisNote];
        tone(pinSpeaker, melody[thisNote],noteDuration);

        // to distinguish the notes, set a minimum time between them.
        // the note's duration + 30% seems to work well:
        intpauseBetweenNotes = noteDuration * 1.30;
        delay(pauseBetweenNotes);
        // stop the tone playing:
            noTone(pinSpeaker);
    }

}

//This function is from an article named Native Capacitive Sensors without additional Hardware.
//Retrieved from the Arduino official website from
http://playground.arduino.cc/Code/CapacitiveSensor

// readCapacitivePin
//  Input: Arduino pin number
//  Output: A number, from 0 to 17 expressing
//  how much capacitance is on the pin
//  When you touch the pin, or whatever you have
//  attached to it, the number will get higher
#include "pins_arduino.h" // Arduino pre-1.0 needs this
uint8_treadCapacitivePin(intpinToMeasure) {
  // Variables used to translate from Arduino to AVR pin naming
volatile uint8_t* port;
volatile uint8_t* ddr;
volatile uint8_t* pin;
  // Here we translate the input pin number from
  //  Arduino pin number to the AVR PORT, PIN, DDR,
  //  and which bit of those registers we care about.
byte bitmask;
port = portOutputRegister(digitalPinToPort(pinToMeasure));
ddr = portModeRegister(digitalPinToPort(pinToMeasure));
bitmask = digitalPinToBitMask(pinToMeasure);
pin = portInputRegister(digitalPinToPort(pinToMeasure));
  // Discharge the pin first by setting it low and output
  *port &= ~(bitmask);
  *ddr  |= bitmask;
delay(1);
  // Make the pin an input with the internal pull-up on
  *ddr&= ~(bitmask);
  *port |= bitmask;

  // Now see how long the pin to get pulled up. This manual unrolling of the loop
  // decreases the number of hardware cycles between each read of the pin,
  // thus increasing sensitivity.
uint8_t cycles = 17;
if (*pin & bitmask) { cycles =  0;}
else if (*pin & bitmask) { cycles =  1;}
else if (*pin & bitmask) { cycles =  2;}
else if (*pin & bitmask) { cycles =  3;}
else if (*pin & bitmask) { cycles =  4;}
else if (*pin & bitmask) { cycles =  5;}
else if (*pin & bitmask) { cycles =  6;}
else if (*pin & bitmask) { cycles =  7;}
else if (*pin & bitmask) { cycles =  8;}
else if (*pin & bitmask) { cycles =  9;}
else if (*pin & bitmask) { cycles = 10;}
else if (*pin & bitmask) { cycles = 11;}
else if (*pin & bitmask) { cycles = 12;}
else if (*pin & bitmask) { cycles = 13;}
else if (*pin & bitmask) { cycles = 14;}
else if (*pin & bitmask) { cycles = 15;}
else if (*pin & bitmask) { cycles = 16;}

  // Discharge the pin again by setting it low and output
  //  It's important to leave the pins low if you want to
  //  be able to touch more than 1 sensor at a time - if
  //  the sensor is left pulled high, when you touch
  //  two sensors, your body will transfer the charge between
  //  sensors.
  *port &= ~(bitmask);
  *ddr  |= bitmask;

return cycles;
}
```

//The following code plays MorseCode that signals the name of the music. They are not included in the final code.

```
voidMorseCodeTwinkleStar()   //Signals "TwinkleStar"
{
dash(1); pausebtwletters(1);
dot(1);dash(1);dash(1);pausebtwletters(1);
dot(1);dot(1);pausebtwletters(1);
dash(1);dot(1);pausebtwletters(1);
dash(1);dot(1);dash(1);pausebtwletters(1);
   dot(1);dash(1);dot(1);dot(1);pausebtwletters(1);
dot(1);pausebtwletters(1);
dot(1);dot(1);dot(1);pausebtwletters(1);
dash(1);pausebtwletters(1);
dot(1);dash(1);pausebtwletters(1);
dot(1);dash(1);dot(1);pausebtwletters(1);
}

voidMorseCodeAmazingGrace()  //Signals "AmazingGrace"  //32s for 0.25s per dot
{
dot(2);dash(2);pausebtwletters(2);
```

```
dash(2);dash(2);pausebtwletters(2);
dot(2);dash(2);pausebtwletters(2);
  dash(2);dash(2);dot(2);dot(2);pausebtwletters(2);
dot(2);dot(2);pausebtwletters(2);
dash(2);dot(2);pausebtwletters(2);
dash(2);dash(2);dot(2);pausebtwletters(2);
dash(2);dash(2);dot(2);pausebtwletters(2);
dot(2);dash(2);dot(2);pausebtwletters(2);
dot(2);dash(2);pausebtwletters(2);
  dash(2);dot(2);dash(2);dot(2);pausebtwletters(2);
dot(2);pausebtwletters(2);
}

voidMorseCodeJingleBells()  //Signals "JingbleBells"
{
  dot(3);dash(3);dash(3);dash(3);pausebtwletters(3);
dot(3);dot(3);pausebtwletters(3);
dash(3);dot(3);pausebtwletters(3);
dash(3);dash(3);dot(3);pausebtwletters(3);
  dot(3);dash(3);dot(3);dot(3);pausebtwletters(3);
dot(3);pausebtwletters(3);
  dash(3);dot(3);dot(3);dot(3);pausebtwletters(3);
dot(3);pausebtwletters(3);
  dot(3);dash(3);dot(3);dot(3);pausebtwletters(3);
  dot(3);dash(3);dot(3);dot(3);pausebtwletters(3);
dot(3);dot(3);dot(3);pausebtwletters(3);
}

voidMorseCodeMoonlight()    //Signals "Moonglight"
{
dash(3);dash(3); pausebtwletters(3);
dash(3);dash(3);dash(3);pausebtwletters(3);
dash(3);dash(3);dash(3);pausebtwletters(3);
dash(3);dot(3);pausebtwletters(3);
  dot(3);dash(3);dot(3);dot(3);pausebtwletters(3);
dot(3);dot(3);pausebtwletters(3);
dash(3);dash(3);dot(3);pausebtwletters(3);
  dot(3);dot(3);dot(3);dot(3);pausebtwletters(3);
dash(3);pausebtwletters(3);
}

//Turn on LED or speaker to signal a 'dot'
void dot(intpinLED)
{
if (pinLED==3)
    {
            digitalWrite(pinRLED, HIGH);
            digitalWrite(pinSpeaker, HIGH);
            delay(250);
            digitalWrite(pinRLED, LOW);
            digitalWrite(pinSpeaker, LOW);
            delay(250);
      }

if (pinLED==2)
      {
            digitalWrite(pinGLED, HIGH);
            digitalWrite(pinSpeaker, HIGH);
            delay(250);
            digitalWrite(pinGLED, LOW);
            digitalWrite(pinSpeaker, LOW);
            delay(250);
      }

if (pinLED==1)
      {
  //        digitalWrite(pinBLED, HIGH);
            digitalWrite(pinSpeaker, HIGH);
            delay(250);
  //        digitalWrite(pinBLED, LOW);
            digitalWrite(pinSpeaker, LOW);
            delay(250);
       }
}

//Turn on the LED or speaker to singal a 'dash' in Morse Code
void dash(intpinLED)
{
if (pinLED==3)
      {
            digitalWrite(pinRLED, HIGH);
            digitalWrite(pinSpeaker, HIGH);
            delay(750);
            digitalWrite(pinRLED, LOW);
            digitalWrite(pinSpeaker, LOW);
            delay(250);
      }

if (pinLED==2)
      {
            digitalWrite(pinGLED, HIGH);
            digitalWrite(pinSpeaker, HIGH);
            delay(750);
            digitalWrite(pinGLED, LOW);
            digitalWrite(pinSpeaker, LOW);
            delay(250);
     }

if (pinLED==1)
     {
            //digitalWrite(pinBLED, HIGH);
            digitalWrite(pinSpeaker, HIGH);
            delay(750);
            //digitalWrite(pinBLED, LOW);
            digitalWrite(pinSpeaker, LOW);
            delay(250);
     }
}

//Generate a pause between letters when sending Morse Code
voidpausebtwletters(intpinLED) //Pause the lengths of 3 dots
{
if (pinLED==pinRLED)
    {
digitalWrite(pinRLED, LOW);
            digitalWrite(pinSpeaker, LOW);
            delay(750);
    }

if (pinLED==pinGLED)
    {
digitalWrite(pinGLED, LOW);
            digitalWrite(pinSpeaker, LOW);
            delay(750);
    }

    {
     //   digitalWrite(pinBLED, LOW);
            digitalWrite(pinSpeaker, LOW);
            delay(750);
    }
}
```

CODE FOR THE INTERACTIVE FINGERS

```
#include <CapacitiveSensor.h>

int high=1000;      // PWM high in ms, change this value for different PWM
int low=5000;       // PWM low in ms
int led=8;          //asign led pin
//assign Flexinol pin number for each finger
int thumb=9;
int index=10;
int middle=11;
int ring=12;
int pinky=13;
CapacitiveSensor   left = CapacitiveSensor(2,3);        //30Mresistor between pins 3&2
CapacitiveSensor   mid = CapacitiveSensor(4,5);        // 30M resistor between pins 4&5
CapacitiveSensor   right= CapacitiveSensor(6,7);        // 30M resistor between pins 6&7

void setup()
{
Serial.begin(9600);
pinMode(led, OUTPUT);  //pin as output
pinMode(thumb, OUTPUT);    //pin as output
pinMode(index, OUTPUT);    // pin as output
pinMode(middle, OUTPUT);    // pin as output
pinMode(ring, OUTPUT);    // pin as output
pinMode(pinky, OUTPUT);    // pin as output
}

void loop()
{
   long start = millis();
   long totalm = mid.capacitiveSensor(30);
   long totalr =  right.capacitiveSensor(30);
   long totall =  left.capacitiveSensor(30);

Serial.print(totalr);                  // print sensor output 1
Serial.print("\t");
Serial.print(totalm);                  // print sensor output 2
Serial.print("\t");
Serial.print(totall);                  // print sensor output 1
Serial.print("\t");
```

```
    if ((totall<5000) & (totalm<5000) & (totalr<5000)){
digitalWrite(led,HIGH);
    }
     else if (totall>5000) {  //flex fingers from pinky to thumb
digitalWrite(led,LOW);

digitalWrite(pinky, HIGH);   // sets the pin HIGH
        delay(high);                  // PWM high state
digitalWrite(pinky, LOW);                          // sets the pin LOW

digitalWrite(ring, HIGH);   // sets the pin HIGH
        delay(high);                  // PWM high state
digitalWrite(ring, LOW);                           // sets the pin LOW

digitalWrite(middle, HIGH);   // sets the pin HIGH
        delay(high);                  // PWM high state
digitalWrite(middle, LOW);                         // sets the pin LOW

digitalWrite(index, HIGH);   // sets the pin HIGH
        delay(high);                  // PWM high state
digitalWrite(index, LOW);                          // sets the pin LOW

digitalWrite(thumb, HIGH);   // sets the pin HIGH
        delay(high);                  // PWM high state
digitalWrite(thumb, LOW);                          // sets the pin LOW
        delay(low);
      }
      else if (totalr>5000) {    //flex fingers from thumb to finger
digitalWrite(led,LOW);

digitalWrite(thumb, HIGH);   // sets the pin HIGH
        delay(high);                  // PWM high state
digitalWrite(thumb, LOW);                          // sets the pin LOW

digitalWrite(index, HIGH);   // sets the pin HIGH
        delay(high);                  // PWM high state
digitalWrite(index, LOW);                          // sets the pin LOW

digitalWrite(middle, HIGH);   // sets the pin HIGH
```

```
        delay(high);                // PWM high state
digitalWrite(middle, LOW);                  // sets the pin LOW

digitalWrite(ring, HIGH);   // sets the pin HIGH
        delay(high);                // PWM high state
digitalWrite(ring, LOW);                    // sets the pin LOW

digitalWrite(pinky, HIGH);   // sets the pin HIGH
        delay(high);                // PWM high state
digitalWrite(pinky, LOW);                   // sets the pin LOW
        delay(low);
      }
    else if(totalm>5000) {  //flexx all fingers
digitalWrite(led,LOW);

digitalWrite(thumb, HIGH);
digitalWrite(index, HIGH);
digitalWrite(middle, HIGH);
digitalWrite(ring, HIGH);
digitalWrite(pinky, HIGH);
        delay(high);

digitalWrite(thumb, LOW);
digitalWrite(index, LOW);
digitalWrite(middle, LOW);
digitalWrite(ring, LOW);
digitalWrite(pinky, LOW);
        delay(low);
    }

}
```

APPENDIX B

Bill of Materials for the Dragon Circuit

Label	Part Type	Properties
Arduino1	Arduino	processor ATmega; variant Arduino UNO R3
J1	Piezo Speaker	
LED1	Green LED - 5mm	package 5 mm [THT]; leg yes; color Green (565nm)
LED2	Red LED - 5mm	package 5 mm [THT]; leg yes; color Red (633nm)
LED3	Green LED - 5mm	package 5 mm [THT]; leg yes; color Green (565nm)
LED4	Green LED - 5mm	package 5 mm [THT]; leg yes; color Green (565nm)
Part1	LilyPad Light Sensor	lilypad Yes
Part2	Humidity Temperature Sensor SHT15	
Q1	Basic FET N-Channel	package DPak [SMD]; type n-channel
Q2	Basic FET N-Channel	package DPak [SMD]; type n-channel
R2	150 Ω Resistor	package THT; tolerance ±5%; bands 4; resistance 150Ω; pin spacing 400 mil
R3	150 Ω Resistor	package THT; tolerance ±5%; bands 4; resistance 150Ω; pin spacing 400 mil
R4	150 Ω Resistor	package THT; tolerance ±5%; bands 4; resistance 150Ω; pin spacing 400 mil
R5	100 Ω Resistor	package THT; tolerance ±5%; bands 4; resistance 100Ω; pin spacing 400 mil
R6	15k Ω Resistor	package THT; tolerance ±5%; bands 4; resistance 15kΩ; pin spacing 400 mil
R7	15k Ω Resistor	package THT; tolerance ±5%; bands 4; resistance 15kΩ; pin spacing 400 mil
R8	15 Ω Resistor	package THT; tolerance ±5%; bands 4; resistance 15Ω; pin spacing 400 mil

Label	Part Type	Properties
R9	22 Ω Resistor	package THT; tolerance ±5%; bands 4; resistance 22Ω; pin spacing 400 mil
R10	Trimmer Potentiometer	package THT; size Trimmer - 12mm; track Linear; type Trimmer Potentiometer; maximum resistance 10kΩ
U1	Flexinol	
U2	Flexinol	
U3	Phidget Sound Sensor - 3 pins	
VCC2	Battery block 12V	Voltage 12V

Table 5.Assembly List for the Dragon Board.

REFERENCES

Adafruit Learning System.*TMP36 Temperature Sensor*. Retrieved September22, 2012, from http://learn.adafruit.com/tmp36-temperature-sensor

Adam C. Siegal& Scott T. Phillips. (2010). Foldable Printed Circuit Boards on Paper Substrate. *Wiley InterScience*, *28-35*.

Adam C. Siegal& Scott T. Phillips.(2010). Advanced Functional Materials.*Supporting Information for Foldable Printed Circuit Boards on Paper Substrates.*Retrieved September 19, 2012, from UBC library.

Arduino.*Native Capacitive Sensors without additional Hardware*. Retrieved Jan. 5th, 2013, from http://playground.arduino.cc/Code/CapacitiveSensor

Capsense Library. Retrived November 26, 2012, from http://playground.arduino.cc//Main/CapacitiveSensor?from=Main.CapSense.

DFRobot.*Analog Sound Sensor*. Retrieved from September 22, 2012, from http://www.dfrobot.com/wiki/index.php?title=Analog_Sound_Sensor_(SKU:_DFR0034)

Dimitris C. Lagoudas. (2008). Shape Memory Alloys: Modeling and Engineering Applications. Texas: Springer.

Dragon origami pattern. Retrieved September 20, 2012, fromhttp://www.youtube.com/watch?v=oR10rGLzXp4&feature=watch_response

Dynallyoy, Inc. *Flexinol Actuator Wire Technical and Design Data*. Retrieved September 22, 2012, from http://www.dynalloy.com/TechDataWire.php

Electronics-Tutorials.(2012, September).*Electronics Tutorial about Light Sensors*. Retrieved September 20, 2012, from http://www.electronics-tutorials.ws/io/io_4.html

HowStuffWorks.*How Inkjet Printers Work.* Retrieved September 22, 2012, from http://computer.howstuffworks.com/inkjet-printer.htm

Instructables, Turn a pencil drawing into a capacitive sensor for Arduino. *Retrieved November 22, 2012.* http://www.instructables.com/id/Turn-a-pencil-drawing-into-a-capacitive-sensor-for/

Kamyshny, Alexander & Steinke, Joachim.(2011). Metal-based Inkjet inks for printed electronics.*The Open Applied Physics Journal, 4, 19-36.*
MakeProjects.*The True Meter*. Retrieved September 21, 2012, from http://makeprojects.com/Project/The-Truth-Meter/703/1#.UF4D641lSSE

Magicball origami pattern.*RetrievedSeptember 20, 2012*.http://www.youtube.com/watch?v=tGhcTwIJ4Es&feature=related

MIT Media Lab. *Intro to shape memory alloy actuation using Flexinol.* Retrieved September 21, 2012, from http://fab.cba.mit.edu/classes/MIT/863.10/people/jie.qi/Flexinol_intro.html

National Instruments.*SoundSensors.*Retrieved September 20, 2012, from http://www.ni.com/white-paper/10691/en