Sparky

Project Developed By: Group 2

Report Authors: Andris Zonies, Cameron Leland,

Theodore Davidson, Nicholas Rizzi

Abstract

Our client is creating a mobile museum expo to teach STEM topics. They need us to build an interactive, portable, and entertaining exhibit that can teach high schoolers about STEM topics. The client wants this mobile exhibit to bring STEM to young people to encourage their interest in it and get more young people to go into STEM careers. Our design needed to interact with students in an entertaining way while still teaching them a STEM topic. It should also encourage a cooperative environment like the environment that real engineers and scientists work under.

With these guidelines, we created a customizable circuit with two parallel branches, scaled up to be more readily accessible to students. It teaches concepts of series and parallel circuits by allowing users to place LEDs and resistors into the first branch and see the effect that they have on hardwired components in the second branch, as well as on other components places in the first branch. All of the components are detected and powered from the backend, allowing users to learn about electricity while keeping all of the actual current away from where it can harm them. Our final product did end up with some connectivity issues. However, based on our survey results we can conclude that the prototype succeeded in being interactive, entertaining, and educational, even if it did not encourage cooperation the way that we wanted it to.

Table of Contents

List of Charts

List of Figures

Project 2 Technical Report
Stacked Cornerstone of Engineering
Group 2

Introduction

Our client is building a science museum expo for high schoolers. They have decided to make the expo mobile, so that it can be brought to its target audience and thus reach more people. The expo is meant to encourage young people's interest in STEM and encourage more young people to go into STEM careers. This is because America needs more young people going into STEM, and our client's expo is supposed to aid in that effort by bringing STEM directly to students. The client wants the expo to interact with students in a fun, interactive, and educational way to build STEM interest and understanding. It should also encourage cooperation to mimic the environment that professional engineers and scientists work in.

We have been asked to design a single exhibit of that mobile expo. Based on the expo's overarching goals and observations of people's interactions with exhibits in the Boston Museum of Science, we came up with the following design objectives for our exhibit:

1. Educational
2. Entertaining
3. Interactive
4. Cooperative

We were also given the following constraints by our client, to make sure that our exhibit could be implemented in the expo:

1. The entire exhibit must sit on a 30”x40” tabletop.
2. It must be safe for general use by high school students.
3. It must include educational text about the concept being taught.

4. It must be easily transportable and setup, specifically transportable by two people and set up or broken down in less than 30 minutes.
5. The entire cost must be less than $100.

Methodology

Physical Design:

We started our design process by defining goals for our exhibit. Using observations of exhibits at the Boston Museum of Science, we created four main design goals:

1. Educational
2. Entertaining
3. Cooperative
4. Interactive

We let these principles guide our design process and we ended up with three main concepts: an Arduino step sequencer, a circuit puzzle, and a wind tunnel with paper planes. We decided to rule out the wind tunnel because it felt like too small in scope, while any other adaptation would take away the interactive portion of making your own plane. The step sequencer was ruled out because it wasn't really that educational, step sequencers are too much of a niche piece of equipment to be useful knowledge. We decided to develop the circuit puzzle concept because of how much it aligned with our design goals.

In our brainstorming sessions we very quickly realized that the concept of a circuit puzzle was confining our solution, so we widened our concept out to a full customizable circuit. This would allow the user to start exploring the concepts behind circuitry by building custom circuits before they could be given puzzles that may require prior circuitry knowledge to solve.

Moving forward, we developed a basic concept for the design. We decided that we would build our board to model two parallel branches, each with 3 spaces to put components in. This was done to allow the prototype to show the properties of series circuits within a branch as well as the properties of parallel circuits through the interactions between the two branches. The idea was that each time you put a component into a slot it would connect up with the board such that it would carry information on what kind of component it was as well, as well as receive power if needed. This was done because we wanted to ensure that all electrical connections could be switched off if there were users near them, to prevent anyone from being hurt by our design.

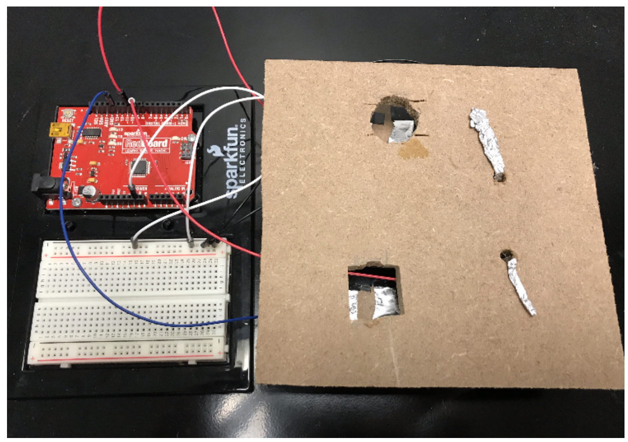

FIGURE 1: PROOF OF CONCEPT COMPONENT SPACE

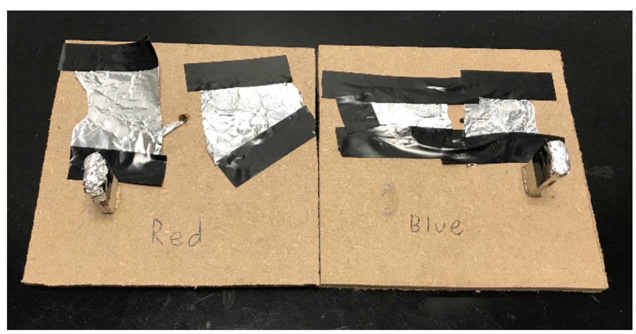

FIGURE 2: PROOF OF CONCEPT COMPONENT PIECE

Our first attempt at creating a working component space was our proof of concept. We came up with an idea for a peg to reach through the component space and complete a circuit down below, with different arrangements of pegs used for different components. This accomplished 2 main goals. First of all, it removed the detection elements from the an area that the user could access them improperly to prevent them from accidentally turning on current on an uncovered space. Secondly, it gave a wide range of flexibility to our design, since there was a very large number of orders of 2 or 3 pegs. We put aluminum foil on the top of the proof of concept and wired it so that it could receive power and give it to a component when one was detected in the space. We then created 2 components with different peg geometries: one red LED, and one blue LED. We programed the circuit to light them to different brightnesses depending on which LED it detected.

We encountered a major problem in our proof of concept in the lack of consistency in electrical connections. Because there was no tolerances built into getting two solid pieces of wood to match up flush, we could almost never get both detection and power to work correctly. We took this lesson into our next design for the electrical connections.

After the proof of concept, we started work on our full board. From that we knew that we would want to reduce variance between components and component spaces as much as possible, so we decided to mill the component spaces into the board. We designed the component spaces with three peg holes in each slot to avoid confining ourselves to a limited number of types of components. We decided to make the board 18” x 24” because that was the maximum cutting area of the CNC mill, and it was cut out of ¾ inch wood to give sufficient depth for recessed component spaces.

We created two types of components for initial testing: resistors and LEDs. The LEDs were meant to give the circuit outputs, and the resistors let the users explore the impact of resistors on the circuit.

For detection, we switched to using copper tape on top of foam to introduce a range of tolerance for connections. The theory was that a metal-bottomed peg would still come down and hit two pieces of metal to complete a circuit, but this time there would be a large tolerance for connection since the foam could just continue to squish inwards and maintain a circuit. We used the same method of copper tape on foam to improve the power distribution method as well. Both detection and power distribution worked effectively at first, but after 20 minutes of use for code debugging the copper tape started to fatigue and connections began to fail. We knew that the exhibit would need to be used for hours on expo day, so we decided to look for a better, hardier design.

Our next solution was conductive foam. This was supposed to work on the same principle of a metal-bottomed peg coming down and completing a circuit; the only difference this time would be that the whole foam would be conductive instead of just the copper on the top. We quickly figured out that that would not work, however, because the foam only consistently conducted electricity if you put leads into it and then put pressure on the foam. This however, played excellently to our purposes for it: we simply put two leads into one piece of foam and let the pressure coming down from the peg turn the foam conductive and complete the circuit. This is what we used in our final design. As for power, we attached copper plates to the top of conductive foam and copper-taped leads to the copper through the bottom of the board. Unfortunately, getting all of the leads hooked up properly so that just the right amount of pressure could consistently activate them took up time, and we were not able to properly hook up and debug a second circuit branch. We decided to hardwire the components there instead. This gave us the function of having a working second branch (thus showing the interactions between the two branches to preserve our ability to teach parallel circuit concepts), but that came at the price of the interactivity.

Code:

The backbone of this project was the code to control the circuit components. In one of our first meetings for the project, we generated the pseudocode. First, we made sure to have a fundamental understanding of the math behind parallel and series circuits. This involved compiling all the equations necessary to calculate the resistance and current of each branch in the circuit, the resistance of each circuit component, and the voltage that should be sent to each component. By calculating these values, we could accurately simulate a circuit with two parallel branches as we could determine the amount of power to send to the output components (in our case LEDs and motors).

The next problem was determining what component was being placed in a component space (we coined this our component detection system). In our physical design, we planned to have three potential positions for pegs on each circuit component. The combination of pegs used would determine what type of circuit component was being placed down (originally an LED or resistor). We used this information to determine whether or not to output power to a space (thus preventing the potential electrocution of the user) and to determine the resistance of each component for our calculations. We planned to use an array to store the detection pin numbers corresponding to each component space, allowing us to check what combination of pins was in use per each component space and assign it all the relevant values.

The first iteration of the code only controlled one branch because we planned to have one Arduino controlling each branch in the circuit. Additionally, writing the code to operate a single branch reduced the number of potential bugs in the code and made it easier to make quick fixes. We knew it would be easy to expand the logic for the calculations to a second branch because the math was essentially the same for each branch.

We wrote all our code for initializing the arrays of pin values and resistance values for each component in the “setup” function. All the calculations for the single branch were performed in the “loop” function (with the resistance value of the second branch hard coded), and the calculations’ results were then used to determine how much voltage to output the component.

We also created a function named “getState” to determine what component was placed in a component space. Every time it was called, it would populate an array with what type of component was in each space. This array was then read by the code to calculate the resistance of each component and the power output in “loop”. To test this code a test rig was built using an Arduino that would mimic the connections of the actual circuit, but have no actual output components. This involved hardwiring power connections to input pins on the Arduino. For this phase of testing, we had a representation of the pin array and the component type array outputted to the serial monitor, along with the value being outputted by the analog outputs to each output component. Here, we were able to see that the code would in theory work properly for a single branch. However, once we moved from the test bed to an initial version of our prototypes circuit, it was soon realized that the values being outputted to the LEDs were inadequate for the user to perceive a difference in brightness. This turned out to be a combination of factors, the way we mapped the voltage values to output values along with the resistances we were setting each component to. To remedy this situation, we dramatically increased the coded resistances of each component type and tried different combinations of mapping values until the LEDs functioned desirably.

In our testing process, we noticed that when pins 0-3 were in use on the red board, the serial monitor would output incoherent characters. We thought it was a result of a broken red board. However, after speaking with a red shirt in FYELC about this matter, we came to the understanding that you cannot use pins 0-3 if you want the serial monitor to function properly, and those pins also don't work very well for taking inputs and outputs. This resulted in a major change in our design. Originally, we planned to have two red boards that would communicate to each other, with each board controlling a single branch, as the number of pins on a single board was too low to support to branches worth of components. This depended on us being able to use more pins than each redboard had, however. This fact led us to purchase an Arduino mega so that we could connect all the components to a single board.

The next version of the code was an attempt to have our logic work with the Arduino mega. This involved redefining what pins corresponded to what components in the pin array and setting up the code to completely calculate the necessary values for two branches. However, once this revision was uploaded to the Arduino and tested with the prototypes circuit, we discovered it was not functional.

We had to go back to a lower level method of testing. We built a test bed that included an array of six lights on a breadboard with the proper power outputs wired to them. To made the Arduino read components as plugged in, we hard wired ground connections to the detector pins. To test the code in attempt to fix it, we would change the combinations of hard wired detection pins to simulate circuit components being switched out, and observe the results on both the serial monitor (we had the code output the same values mentioned before) and the lights. After fixing some problems with the code's syntax, we realized that the LEDs were not lighting up properly again due to the resistance values of the components and the mapping values. Through trial and error using serial.print, we were able to find a combination of mapping and resistance values that resulted in perceivable changes in brightness of the LEDs.

Our final version of the code was simply a cleaner and more refined version of the fourth version. There had been talk to hard code the values for components and resistances for the second branch, as we decided we were only going to have one branch where the user could swap out components. However, in an effort to not break the code the day before the exposition, we simply hardwired the detection and power connections for the second branch and kept the code the same. It already properly doing the calculations for both branches, there was no sense in messing with it.

Testing:

To evaluate our prototype's ability to meet design goals, we created a google form for users to fill out after they finished using the exhibit. In order to evaluate in terms of our design goals, the survey asked the users to rate the project from 1 to 10 in terms of how interactive, entertaining, and cooperative it was. It also asked them to select topics that they thought they had learned something about from a list of topics that the exhibit intended to teach. It then asked them why they left the exhibit, if anything was confusing, and finally what problems they saw. We set this form out next to our prototype and asked users to fill it out when they were done using our exhibit.

Final Design

Physical Design:

Our final design created a functional customizable circuit. We milled a board to represent a circuit with two branches and six component slots. This board rests on top of 4 pegs so it is elevated and the pegs can stick through the slots. We designed three component slots for each branch which act as plugs for lights, resistors, or motors to show how different combinations in series or parallel affect a circuit's outputs (See Figures **Figures)**. This was accomplished through separate detection and a power systems. The method of detection used for our final design uses pressure on conductive foam. A peg attached to the bottom of the component reaches through a slot in the upper board and provides pressure of the conductive foam. This makes the foam act conductive (where it previously did not), and it then completes a circuit (See Figure **FIGURE**) . The arduino then uses that completed circuit like a button press and recognizes the component

FIGURE 3: FINAL BOARD (EMPTY)

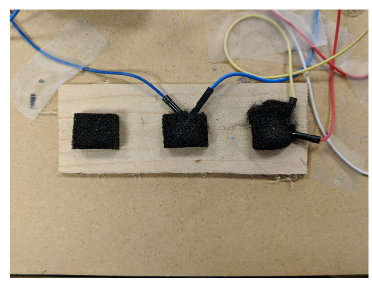

FIGURE 4: FINAL DETECTION SYSTEM

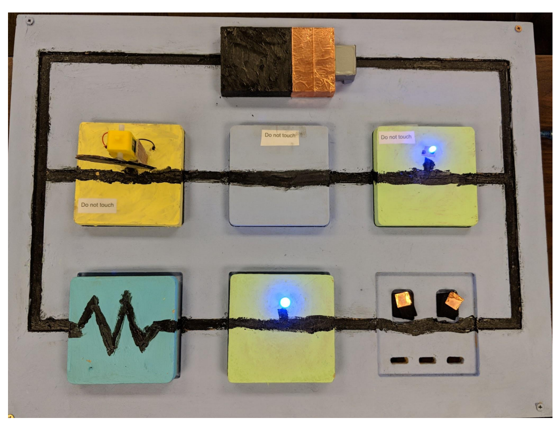

FIGURE 5: FINAL BOARD (FULL)

in that space. The arduino uses this data to calculate the respective voltages and resistances across each branch, then to send correct voltages to each components. The power is sent to the component from two wires connected to the arduino on one side and copper plates on the other. The copper plates sit on top of the board but in the component spaces so when a component is rested on top of it the voltage sent to it powers the output (See Figure **Figure)**. For example, the light pins for our LED are stuck through through holes in the component and than the coppertaped to the bottom of the component so the copper tape connects with the copper plates when placed down, thus completing the circuit and powering the light.

FIGURE 6: FINAL COMPONENT SPACE

Code:

[1]The final version of our code is comprised of three functions. "setup", "loop", and "getState" (with setup and loop being the ones required by the Arduino language). Then, it goes on to initialize three global arrays–one to store the resistance values of each component currently on the board, one to store the numerical values of three detection pins and single power pin for each component space on the board, and one to store what type of component (1 = Motor, 2 = resistor, 3 = LED) is in each component space (the arrays are named resArr, pinArr, and compArr

[1] The physical code and a more indepth version of the pseudocode are included in the appendix B and A respectively

respectively). Then, a global float variable is declared the store the value to output to each output component (outputValue).

Next comes the setup method, marked by the red 1 in figure 7. In the pinArr, the row index corresponds to the circuit component number (0-5), column indexes 0-2 correspond to the three detector pins connected that component space, and column index 3 corresponds to the power pin for that component space. Next is a for loop that populates the pinArr with the power pin values for each component. Pins 2-13 can be used for PWM output on the Arduino mega, therefore, we assigned the power pin values for each of the six component spaces to pins 8-13 and set all of those pins to the pinMode “output”. The following portion of the code is a for loop that populates the detection pins of the pinArr. On the Arduino mega, pins 22 through 52 can be used as digital inputs. Starting at pin 22, we assigned three pins that are in a row numerically as the detector pin for each input (skipping a single pin number between each component to make wiring easier) along with setting each of the pins used to the pinMode “INPUT_PULLUP”, so we could take advantage of the embedded pullup resistors in the Arduino. For testing purposes, we outputted all the pin numbers for each component space to the serial monitor within each of these two for loops.

The getState function, marked by the red 2 in figure 7**,** is where the code detects what type of component is placed in a specific component space and assigns values accordingly. The function loops from the top to the bottom of the pinArr and reads the detector pin values for each component space. Depending on the combination of detector pins receiving input, the code assigns values to compArr and resArr such that they correspond with the correct type of component. For example, if it detected an LED it would put a 1 in the correct spot in compArr and 150 as its resistance in resArr. For testing purposes, the type of component detected was printed to the serial monitor.

The loop function, marked by the red 3 in figure 7**,** is where most of the calculations take place. First, rB1 (resistance of branch 1) and rB2 (resistance of branch 2) are set to initial values of 150 and getState is called to populate compArr and resArr. Then, the resistance of each branch is calculated by adding the resistance values of each component to the resistance of their respective branch. Next, the current of each branch is calculated with the equations listed in figure 8. This is then used in a for loop where that iterates over then elements of compArr that correspond to components in the first branch to determine where to send power in that branch. If a component is a motor or resistor, it uses Ohm's law to calculate the voltage drop across that component, maps that value to values that the component runs on, then outputs to the correct power pin as determined by pinArr. The next for loop corresponds to the 2nd branch and does the same thing as the first, using the current for the second branch instead of the current for the first. Once that is done, the whole loop function repeats to keep adjusting for different combos of placed components.

```
setup()                                                                 1
        populate the pin array with the detector pin values and power pin values for each
        component space
end

loop()                                                                  2
        call the getState function
        calculate the resistances in each branch by adding up the resistance of each component in
        their respective branches
        calculate the voltage drop across the battery with the built-in resistance for each branch
        calculate the voltage drop and current for each branch
        check the component array value for each component space, depending on the type of
        component and send the proper amount of power to that component
end

getState()                                                              3
        populates the component type array with the component type (LED, resistor, motor)
end
```

FIGURE 7: PSEUDOCODE

voltage drop across the battery = total current * 330

voltage drop across the branch = 5 – voltage drop across the battery

current in the branch = the voltage drop across the branch / branch the resistance of the branch

FIGURE 8: CALCULATIONS

Results

All of our formal evaluation data was taken from our google survey, which was administered at the expo. It asked users questions to evaluate our prototype's ability to meet our design goals, to evaluate its overall quality, and to give criticisms to what the exhibit was lacking.[2] On the day of the expo, the survey got 26 responses.

The first question asked users to select topics they felt they learned something about from a list that we intended our prototype to teach. 8% said they did not learn anything new about any of the topics, while 92% selected at least one topic that they learned about and 69% said that they learned about multiple topics.

The next three questions asked users to rate the exhibit from 1 to 10 on how interactive, entertaining, and cooperative it was. The results are as follows:

	Entertainment	Interactivity	Cooperation
Average Rating:	8.5	9.38	7.31
Minimum Rating:	5	6	1
Maximum Rating:	10	10	10
Median Rating:	8	10	7
Percentage of respondents giving an 8 or higher:	84.62%	92.31%	50.00%

CHART 1: SURVEY STATISTICS

The next two questions asked if the users visited as part of a group. 27% of respondents visited as part of a group, with the average group size being 2.4 people.

[2] **For the exact survey questions, see Appendix __**

After that, we asked users what made them decide to leave. 42% said that they left because they were bored or out of combinations to use, and the rest of the answers were highly varied and generally unrelated to our exhibit (“I had to reset my puzzle”, “Rebecca’s”, “I needed to go to class”). When asked if anything about the exhibit was confusing, 3 respondents out of 26 said that they were confused about why the circuit acted in a specific way, 1 respondent was confused about we we used “only two components”, and the rest said some variant of “No”.

When asked to list problems with the exhibit, 5 respondents criticized the simplicity/lack of options, 4 said the board had issues with connectivity, and 1 said the paint job could have been improved.

Final Design Specs:

Dimensions: 18”x24”x4”

Cost: $91.90

Discussion

Based on the data that we collected from our expo survey, our design generally met our design goals but could have been greatly improved upon. In regards to our first design goal, being educational, 24 out of 26 people said they learned something from the project, with 20 out of 26 saying that they learned something about multiple subjects. This proves that our exhibit met our goal, since almost everyone learned something from it. Additionally, based on interactions with users, we believe that it is possible that some of the people saying that they learned nothing may have already known everything that our exhibit was teaching. There was plenty of engineers there, and some of them already generally knew the concepts of parallel and series of circuitry.

As far as our second design goal, being entertaining, 85% of users gave the exhibit a rating of 8 or above for entertainment, with the average at an 8.5 and the lowest rating being a 5. We took that as a success, since the average user was giving ratings in a very high range and no one gave us anything low enough to indicate them finding a serious issue in that department.

Our third design goal and perhaps our most successfully met goal was interactivity. Our average user rating for interactivity was a 9.38, with the lowest rating at 6 and 92% of ratings being above 8. These were all the highest values that we got for any category, so we considered interactivity to be our largest success.

Our fourth design goal, encouraging cooperation, was the opposite. Our average rating was 7.31, but only 50% rated the project above an 8. Three users rated cooperativity at 3 or lower, with one user rating it a 1. All respondents but 2 gave this category their lowest rating. Based on this, we considered this to be a failure. In order for our exhibit to properly encourage cooperation, it should be able to consistently make people feel like they can use it in a group. Our numbers simply don't show our exhibit doing that. In this way, cooperativity is our exhibit's weakness in terms of design goals.

In our free response questions, we asked users to give us information about problems that they noticed and why they left. Here, our exhibit's other main weaknesses were shown. Firstly, users generally said that they finished using the exhibit because of "Lack of options" or "I did it all", and many said the exhibit's main problem was something like "lacking greater options" or "little variety in the parts". This points towards a lack of customizability in the portions of our exhibit that are used by the user, particularly a need for more component options and more component spaces to create a greater variety of possible circuit designs for the user to explore. Secondly, multiple users complained about "connectivity" or "weak connections", pointing to issues with our component detection and power systems consistently hooking up to components. We concluded afterwards that this was likely due to fatigue in the foam in the detection system. This shows that despite of our many iterations of build, our design still had weaknesses in the detection and powering system that caused problems for users during the expo.

Conclusion

Firstly and most importantly, our project met all of the design requirements from our client:

Design Requirement	How the requirement was met:
Must fit on a 30”x40” tabletop	Prototype has a 18”x24” footprint
Contains an interactive component	Contained interactions in the placement of circuit components
Contains at least one 3d printed, CNC milled, or laser cut part	The top board was CNC milled
Safe for general use	All electrical current turned off when exposed to users, and hidden behind components or the main board when in use
Includes Educational Text	Educational text on resistance, current, and series and parallel circuits was included.[3]
Stored outside of FYELIC	Entire exhibit stored in a team member’s dorm room
Easily transportable by two people	Main board can be carried by one person while another person holds the components in a bag or basket
Set up in less than 30 minutes	Setup is simply plugging the board into power, setting the components on the table, and setting up the informational text. Takes no more than 10 minutes
Costs less than $100	$91.90 spent

CHART 2: DESIGN REQUIREMENTS MET

Beyond reaching our design requirements, our prototype was a general success with a few flaws. It met our design goals for by being educational, interactive, and entertaining, although it lacked an element of cooperation. It had issues with connectivity and at some level lacked the level of customizability that we may have wanted, but it still had enough proper functionality to teach the

[3] **See Appendix ___ for actual slides**

STEM topic at hand. The drawbacks that the project had were mainly due to the level of difficulty that we encountered in getting the detection and powering systems to work. We spent so much time trying to get those working to achieve basic functionality that we didn't have enough time to achieve the levels of customizability and cooperation that we wanted to.

The struggle with the electrical lessons taught us the three biggest lessons of this assignment. First of all, we did the right thing by prototyping and testing early, and by not assuming that the project was going to work the first time. Testing early and failing early gave us time to work on our weaknesses and achieve a functionality by the time that the expo rolled around. If we had waited longer to construct and test those connection methods, we would have been in an even bigger time crunch in the week before the expo and we may not have attained the level of functionality that we did (especially considering the fact that each new iteration of the connections required ordering new parts and waiting for them to ship). The moral of the story; test early and fail early instead of failing on the day of the deadline.

Secondly, it's important to create effective ways for you to test as you're going. The most important step in debugging our code was the creation of a full test bed to mimic the board. We could not have gotten the electrical connections working without our extensive use of Serial print, and our issues with mapping values were only solved by plugging things into the testbed and serial printing values. Ongoing testing tools are essential to give you immediate feedback on things so you can find solutions to small problems quickly, before they affect the full project.

The connection issues also taught us to always properly research materials before making assumptions about how they will function. We encountered issues with this in both the aluminum foil tape that we ordered for the proof of concept and the conductive foam that we used in the final design. We originally thought that the aluminum foil tape would be conductive on both sides and that the conductive foam would be conductive on contact, but neither one of those assumptions turned out to be true. These two assumptions being false caused us to have to change our design both times; we're lucky that we managed to get the conductive foam to work in a different way, or we might not have been able to get our prototype working as well as we did.

Recommendations for Future Work

The first step in developing this prototype further would be to activate a second branch for component placement, allowing more interactivity and reducing the occurrence of our number 1 reason for people leaving ("I ran out of combinations to try"). The detection methods could also still use another evolution. Future improvements could involve detecting components by connecting to pegs with the brushes used in DC motors, with solid copper plates on top of foam, or a more standardized plug. Detection could also be done through an entirely different, non electrical method, like RFID sensors, optical sensing (having the peg block a laser) or a form of ultrasonic range sensing (detecting the presence and/or length of a peg). In any case, the detection and power methods were still not perfect at the end of the process, and are still up for improvement.

Beyond that, more user freedom could be given by building a wider range of parts. Other potential placeable circuit components could include resistors of different resistances, motors, different color LEDs, speakers, or buzzers. Adding any of these would improve the freedom of exploration given to the user and improve interactivity.

To better meet design goals, a cooperative element could be introduced to the exhibit. This could be done by creating multiple size-by-side circuits with puzzles to race through, thus returning to the original “Circuit Puzzle” concept, but really any addition to increase cooperation would result in the prototype better reaching its design goals.

Appendix A: Deeper Level Pseudocode

Initialize the variables for the resistance, current, and voltage across branches one and two

Initialize arrays to store the resistance values of the components in the board, the power and detection pins that correspond to each component space, and the type of component in each component space

Initialize a variable to store the value we output to each powered component

Setup()

Loop from component 0 to 5

Store the power pin value for each component in the pin array (pins 8-13 respectively) and set those pins to output

End

Loop through pins 22 to 44

Store the detection pin values for each component in the pin array (assigned in groups of three to each component with a pin skipped in between groups) and set those pins to input pullup

End

End

Loop()

Set the resistances of each branch to initial values of 150

Call get state

Loop through components 0-2

Calculate the resistance of branch one by adding up the resistance values of components 0-2

End

Loop through components 3-5

Calculate the resistance of branch two by adding up the resistance values of components 3-5

End

Calculate and save the total resistance of the entire circuit

Calculate and save the total current of the entire circuit

```
Calculate and save the voltage drop across the battery
Calculate and save the voltage drop and current in each branch
Loop through components 0 – 2
        If the component is an LED as shown by the component array
                Calculate and save the voltage drop across the component
                Output a mapped version of that value to the component’s power pins
        End
Else If no component is plugged in as shown by the component array
                Output nothing to that component ‘s power pins
        End
End
Loop through components 3 – 5
        If the component is an LED as shown by the component array
                Calculate and save the voltage drop across the component
                Output a mapped version of that value to the component’s power pins
        End
        Else if the component is a motor as shown by the component array
                Calculate and save the voltage drop across the component
                Output a mapped version of that value to the component’s power pins
End
Else If no component is plugged in as shown by the component array
                Output nothing to that component ‘s power pins
        End
End
End

getState()
        Loop through components 0-5
```

If the detector pin combination for that component identifies it as a motor

Set the resistance array value for that component to 200

Set the component array value for that component to 1

End

Else If the detector pin combination for that component identifies it as a resistor

Set the resistance array value for that component to 500

Set the component array value for that component to 2

End

Else If the detector pin combination for that component identifies it as a LED

Set the resistance array value for that component to 150

Set the component array value for that component to 3

End

Else If the detector pin combination for that component identifies it as nothing

Set the resistance array value for that component to 0

Set the component array value for that component to 4

End

End

Appendix B: Raw Code

```
 double rB1 = 0; //Resistance of Branch One
double cB1 = 0; //Current of Branch One
double vB1; //Voltage of Branch one

double rB2 = 0; //Resistance of Branch Two
double cB2 = 0; //Current of Branch Two
double vB2; //Voltage of Branch Two

double resArr[6]; //An array that stores all the resistance values for the components in the branch
int pinArr[6][4]; //An array that stores all the pins connected to each component in the branch
int compArr[6]; //An array that stores what each component is (1 = Motor, 2 = resistor, 3 = LED)

float outputValue; //For storing the brightness of the LED's

void setup() {
  //COMPONENT 0: INPUT = 22, 23, 24 | OUTPUT = 8
  //COMPONENT 1: INPUT = 26, 27, 28 | OUTPUT = 9
  //COMPONENT 2: INPUT = 30, 31, 32 | OUTPUT = 10
  //COMPONENT 3: INPUT = 34, 35, 36 | OUTPUT = 11
  //COMPONENT 4: INPUT = 38, 39, 40 | OUTPUT = 12
  //COMPONENT 5: INPUT = 42, 43, 44 | OUTPUT = 13

  int pinCounter = 8;
  for(int i = 0; i <= 5; i++)       //Initialize output pins for each comp. in column 3 of pinArr
  {
    pinArr[i][3] = pinCounter;
    pinMode(pinCounter, OUTPUT);
    pinCounter++;
  }

  int counter = 0;
  int compNumber = 0;
  for(int i = 22; i <= 44; i++)   //Initialize input pins for each comp. in columns 0-2 of pinArr
  {
      pinArr[compNumber][counter] = i;
      pinMode(i, INPUT_PULLUP);
      counter++;
```

```
      if(counter == 3)    //If you've gone through a full row of the array (and thus 1 component)
      {
       i++;
       counter = 0;
       compNumber++;    //Move on to the next one
      }
  }
}

void loop() {
  double rB2 = 150;      //Set baseline resistances for the branches to prevent
  double rB1 = 150;    //calculating a short circuit

  //for(int i = 0; i <= 2; i++)
  //{
   getState();                //Get the info about what is in each component space
  //}

  for(int i = 0; i <= 2; i++)
  {
   rB1 = resArr[i] + rB1;        //Calculate the resistance of branch 1
  }

   for(int i = 3; i <= 5; i++)
  {
   rB2 = resArr[i] + rB2;        //Calculate the resistance of branch 2
  }



  double rT = 330 + (1 / ( (1/rB1) + (1/rB2) ));          //Calculate the total resistance
  double cT = 5 / rT;                                      //and current for the whole circuit

  double dVR1 = cT * 330;     //Calculate voltage drop across battery with built in resistance

  vB1 = 5 - dVR1;      //Calculate resulting voltage drop across 2 parallel branches of circuit

  cB1 = (vB1 / rB1);    //Calculate the current in branch 1
```

```
double dVR2 = cT * 330;     //Calculate the voltage drop across built in resistance of branch 2

vB2 = 5 - dVR2;             //Calculate the voltage drop across branch 2

cB2 = (vB2 / rB2);          //Calculate the current through branch 2


for(int i = 0; i <= 2; i++)      //For the first branch
{
 if(compArr[i] == 3)           //For all of the components detected as LED's
 {
  double voltage = cB1 * resArr[i] * 100;  //Calculate the voltage drop across them
  outputValue = map(voltage, 0, 110, 0, 255); //map to values the LED can understand
  analogWrite(pinArr[i][3], outputValue);  //Write those values to the LED
 }
 else if(compArr[i] == 4)     //Otherwise if you just took an LED off, turn the power to it off
 {
  analogWrite(pinArr[i][3], 0);
 }
}

for(int i = 3; i <= 5; i++)      //For the second branch
{
 if(compArr[i] == 3)           //For the hardwired LED
 {
  double voltage = cB2 * resArr[i] * 100;             //Calculate the voltage drop across it
  double outputValue = map(voltage, 30, 90, 0, 255); //map to values the LED can understand
  analogWrite(pinArr[i][3], outputValue);             //Write those values to the LED
 }
 if(compArr[i] == 1)           //For the hardwired motor
 {
  double voltage = cB2 * resArr[i] * 100;             //Calculate the voltage drop across it
  double outputValue = map(voltage, 0, 150, 170, 255); //map the values the motor can run on
  analogWrite(pinArr[i][3], outputValue);             //Write those values to the motor
 }
 else if(compArr[i] == 4)
 {
  analogWrite(pinArr[i][3], 0);       //If there's nothing in the space, write turn the power off
```

```
    }
  }
}

void getState()
{
  for(int i = 0; i <= 5; i++)       //For all of the components
  {
    if( (digitalRead(pinArr[i][0]) == LOW) && (digitalRead(pinArr[i][1]) == HIGH) &&
(digitalRead(pinArr[i][2]) == HIGH) )          //If just the 1st peg got pressed
    {
      resArr[i] = 200;
      compArr[i] = 1;                          //It's a motor
    }

    else if( (digitalRead(pinArr[i][0]) == HIGH) && (digitalRead(pinArr[i][1]) == LOW) &&
(digitalRead(pinArr[i][2]) == HIGH) )          //If just the 2nd peg got pressed
    {
      resArr[i] = 500;
      compArr[i] = 2;                          //It's a resistor
    }

    else if( (digitalRead(pinArr[i][0]) == HIGH) && (digitalRead(pinArr[i][1]) == HIGH) &&
(digitalRead(pinArr[i][2]) == LOW) )           //If just the third beg got pressed
    {
      resArr[i] = 150;
      compArr[i] = 3;                          //It's an LED
    }
    else if( (digitalRead(pinArr[i][0]) == HIGH) && (digitalRead(pinArr[i][1]) == HIGH) &&
(digitalRead(pinArr[i][2]) == HIGH ) )         //If no pegs are pressed
    {
      resArr[i] = 0;
      compArr[i] = 4;                          //the space is empty
      Serial.print(" NOTHING\t");
    }
  }
}
```

Appendix C: Survey Questions

Museum Exhibition Feedback

* Required

1. **Which of the following topics do you feel that you learned something new about from this exhibit? (Check as many as apply)** *
   Check all that apply.
   - [ ] Resistance
   - [ ] Current
   - [ ] Series Circuits
   - [ ] Parallel Circuits
   - [ ] None of the above

2. **On a scale of 1 to 10, how entertaining did you find this exhibit?** *
   Mark only one oval.

1	2	3	4	5	6	7	8	9	10
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

3. **On a scale of 1 to 10, how interactive did you find this exhibit?** *
   Mark only one oval.

1	2	3	4	5	6	7	8	9	10
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

4. **On a scale of 1 to 10, how much did you feel the exhibit encouraged cooperation?** *
   Mark only one oval.

1	2	3	4	5	6	7	8	9	10
◯	◯	◯	◯	◯	◯	◯	◯	◯	◯

5. **Did you visit the exhibit as part of a group?** *
   Mark only one oval.
   - ◯ Yes
   - ◯ No

6. **If so, how many people were in your group? (If did not visit in a group, don't fill this question out)**

Mark only one oval.

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8

7. **What made you decide to leave the exhibit?** *

8. **Was anything about the exhibit confusing to you?**

9. **What problems with the exhibit did you notice? Was the it lacking anything?**

Appendix D: Survey Data

	Which of the following topics do you feel that you learned something new about from this exhibit? (Check as many as apply)		On a scale of 1 to 10, how entertaining did you find this exhibit?	On a scale of 1 to 10, how interactive did you find this exhibit?	On a scale of 1 to 10, how much did you feel the exhibit encouraged cooperation?
	Resistance, Current, Series Circuits, Parallel Circuits		9	10	10
	Resistance		8	8	4
	Resistance, Current, Series Circuits, Parallel Circuits		10	10	10
	Resistance, Current, Series Circuits, Parallel Circuits		8	10	10
	Resistance, Current		10	10	9
	Resistance		8	10	7
	Resistance, Current		9	9	1
	Resistance, Current, Parallel Circuits		10	10	10
	Resistance, Parallel Circuits		9	10	7
	Resistance, Current, Series Circuits, Parallel Circuits		8	10	6
	Resistance, Current, Series Circuits, Parallel Circuits		10	10	9
	Resistance, Current, Series Circuits, Parallel Circuits		10	10	10
	Resistance, Current, Series Circuits, Parallel Circuits		10	10	10
	Resistance, Current, Parallel Circuits		8	9	6
	Resistance		8	10	7
	Resistance, Current, Series Circuits, Parallel Circuits		10	10	8
	Current, Series Circuits, Parallel Circuits		8	10	6
	None of the above		5	7	3
	Resistance		7	9	5
	Resistance, Current, Series Circuits, Parallel Circuits		10	10	10
	Resistance, Current, Parallel Circuits		5	6	3
	Resistance, Current, Series Circuits, Parallel Circuits		7	8	6
	Resistance, Current, Parallel Circuits		8	10	10
	Resistance, Current, Series Circuits, Parallel Circuits		8	10	7
	Resistance, Current, Series Circuits, Parallel Circuits		10	10	10
	None of the above		8	8	6

			Entertainment	Interactivity	Cooperation
Resistance:	23	Average:	8.5	9.38	7.31
Current:	19	Minimum Rating:	5	6	1
Series Circuits:	13	Maximum Rating:	10	10	10
Parallel Circuits:	18	Median:	8	10	7
None of the Above:	2	Percentage rating 8 or higher:	84.62%	92.31%	50.00%

Did you visit the exhibit as part of a group?	If so, how many people were in your group? (If did not visit in a group, don't fill this question out)	What made you decide to leave the exhibit?	Was anything about the exhibit confusing to you?	What problems with the exhibit did you notice? Was the it lacking anything?
No		I saw all the interactions that individual components had with the circuit	No	Could use more explanatory text for how interactive parts function
No		Tried all of the possible combinations	Why the top light got dimmer, but it was explained to me	Some of the electric connections were difficult to make
No		I decided to leave	No	Nope
No		I finished playing with it	nothing happened when i added more than one resistor	simple
No		I did everything possible	No	No
No		Lack of options	Self explanatory, nicely done	Lacking greater options
No		no more options	no	It would be more fun with more options
No		I had tried out everything there	Nope!	Loud music, like the one in front of it
No		I was just browsing everything and had played with everything		Weak connections between the blocks and the board
No		We used all the ftuff	The windmill	One cinnection wsas poor.
No		Time		
No		I never did	How tf it worked	The paint job was a little shoddy
Yes	2	I needed to leave for a class, I would have stayed longer if I could	I'm not very good at physics or engineering but Andris did I good job explaining it to me	Nope!
No		i got bored	the back row of components	
Yes	4	i did most of the combinations	It was straighforward	There was little variety in the parts
Yes	2	I did it all	no!	more variation of circuits/things to plug in
Yes	2	A little simple in design, not that varied	No, it was easy to comprehend	N/A
No		other exhibits	why only two components	needed to press down on led to make work
No		I had to reset my puzzle		
No		Rebecca's	No	No problems. Nah.
Yes	3	I had finished playing with it		
Yes	2	Go to another one	nothing	nothing
No		I got bored?	Nope	Nope!
Yes	2	we done	nope	connectivity
No		my friend made it		
No		Done experimenting and understanding		

Total People Visiting in a Group:	Average Group Size:
7	2.428571429

Appendix E: Educational Text

These were placed next to the baskets of components that users could use. They showed the resistance and output values of each type of component, and they were color coded to match the color that each component was painted.

DC Motors
Output: Motion
Resistance: 150Ω

Resistors
Output: Heat
Resistance: 1000Ω

LED's
Output: Light
Resistance: 150Ω

Following is the rest of the educational text. This was placed on the table next to the prototype, so the users could see it while they were looking down at the board.

Components in Series: What's happening here?

Current can only go forwards, so once it's in a branch it goes straight through. The more resistance is added to a branch, the slower the current goes.

- What happens if you put a resistor in series (in the same branch) with an LED?
- What happens if you add another resistor?

Parallel Branches: What's happening here?

Current is coming out of the battery and splitting when it gets to the two branches. It wants to take the path of least resistance, and changing which path has more resistance affects how the current splits.

- What happens to the components in the second row if you put a resistor in the first row?
- What happens when you take that resistor back off?

Key Concepts: Current

Current is the flow of electrons through a wire. Think of the electrons like water and the wires as the pipes.

Current flows out of the battery, through the wires, through the components of the circuit, and then back into the other side of the battery.

The more current is flowing through a component, the more power it is getting.

Key Concepts: Resistance

The resistance of a component is a set value that tells how much it resists the current going through it. The more resistance a component has, the more it slows down the current going through it.

Think of resistance as clog on the wire: it slows down all the current on that pathway.

Appendix F: Cost of Construction

Item	Cost ($)	Quantity	Usage
24” x 24” x ¾” Plywood	$8.99	3	Base, Supports, Interface, Components
Arduino Mega Board	$14.99	1	Main Circuit Board
2” Screws	$3.00	1 Pack of 50	Binding Interface and base to supports
12” x 12” x ¼” Conductive Foam	$6.00	1	Detection Wiring
Wires	$8.50	1 Pack of 120	Internal Wiring
Paint	$3.50	5	Aesthetics
Copper Plates	$4.95	1 Pack of 10	Power Wiring
Copper Tape	$9.99	1	Power Wiring
Total	$91.90		