Smart Skate

Abbreviations Shorten Stuff

Karla Zaragoza, Sebastian Rivera and Kaleb McClenahan

ECE: 4890, ECE Senior Design

Spring 2023

Introduction

Learning a new skateboard trick requires a significant amount of time, effort and dedication. It requires balance, the correct timing and the correct motion. This is usually accompanied by watching and rewatching videos and tutorials to learn correct foot placement. In addition, it requires repeatedly recording oneself to make sure your feet are in the correct position and that you are moving your body and board at the correct time.

Our solution for this issue is to create a skateboard deck that can be used as a visual tool for riders and will help them to correct their foot placement. The board will use LED pixels on the top of a skateboard deck to show animations of a trick to the user. The animation would show the correct foot placement for both the right and left foot of the user. With this tool, the user will lessen the amount of time it takes to learn a new trick.

With Smart Skate, the amount of time it takes to learn a new skateboarding trick has the potential to be lessened through a system that allows a person to see where their foot placement should be in real time. This system will lessen the time a user spends referencing internet sources and recording themselves to determine if they are performing the steps of a trick correctly. Our system will aid someone in their learning of a new trick.

While creating the prototype, we realized that this tool can not only be an aid, but also act as a cool customization feature that can personalize the skateboard. Most skateboards allow for the customization of the wheels, bearings, and the underside, but the top usually stays the same. Our tool gives the user the ability to personalize the top of the board along with helping them to learn tricks.

Project Outcome

Overall, with Smart Skate we were successful in proving our concept. For our project, one of our objectives was to design a way for the LEDs to be controlled. Originally, we planned to control the LEDs with an Arduino, but in our final implementation we programmed them with a Raspberry Pi. Our second objective was to implement a way for the LEDs to be displayed to the user. For this, the LEDs were stuck to the top of the board in a grid pattern. This allowed the user to see where their foot placement should be. The third objective was to implement an interface that can be used to change the LEDs for a specific trick. We created a website in which the user can choose to display a selected trick. Our project was able to meet our three main objectives; however, we weren't fully able to meet our requirements and constraints.

Our requirements involved creating a system that would control the LEDS. We were able to successfully do this by creating a circuit that connected all the LED strips and was controlled by a Raspberry Pi. Another one of our requirements included the ability of the user to see the animation of the steps of a specific trick. We were able to implement this successfully by creating the LED grid, and by programming the LEDs to display specific animations that correspond to the movement and timing of a specific skateboard trick. Our next requirement was that the animation would be changeable by the user. We accomplished this by creating presets for the user to select on our website. Another one of our requirements was that the different LED arrays would be accessible within a database. Upon our implementation of the trick, we realized it would be more sensible to host the trick locally and use the database as a buffer between our website and local code. This requirement was not technically met; however, in retrospect, it is not practical for our project. Our last requirement was that the display should be able to be stepped on and work with full functionality. Because of the overall structure of the skateboard deck, we were not able to test our project for this requirement. We did not want to risk damage to our prototype.

One of our constraints for this project was that the system will be responsive to user input. We were able to follow this constraint with the creation of our website and the ability it

gives users to interact with the animations on the board. Another constraint was that the power supply needed to be protected and to be safe to mitigate electrical hazards. We were able follow this guideline, by covering our circuitry with electrical tape, and by storing our battery and microcontroller underneath the deck in a space that would have minimal interaction with the user. Our next constraint was that the board must be ridable with all the modifications made to it. Again, as was mentioned previously, we were unable to test the rideability of our prototype as we did not want to potentially damage the prototype before our final presentation of our project. Our next constraint was that the board must be relatively light weight and maintain a "normal" skateboard design. The board also needed to weigh between 14-16 lbs. We were able to meet this constraint by designing our skateboard deck to be thinner so that the LEDS would make less of a contribution to the weight and thickness of the deck of the board. Our last constraint was that the board parts needed to be less than $400. We were also successful in implementing this constraint as we were able to maintain our budget.

For our project, Karla took on the role of team leader. She provided the initial idea, selection of parts and overall design of the skateboard deck. She worked on the physical layout of the deck, and the layout of the LCDs. She codesigned the test bench and circuit of the project. She designed the flexible circuit board, including the layering system used to connect the microcontroller and battery to the circuit. Her work included soldering the flexible circuit board to the LED strips and testing each of the strip's connection. She also worked on the assembly and soldering of both the temporary and final prototype. She also modeled all of the schematics, worked on documentation and created the designs of the animations. Sebastian worked on the communication between the hardware and the software. He created the website and the database, as well as wrote the code that controls the LEDs. He also engineered an alternative solution using the Raspberry Pi when we ran into issues with the Arduino and worked on the new hardware software communication used in the final design. Sebastian also worked on the creation and soldering of each of the flexible circuit board pieces. This includes the testing of each PCB to LED connection as well as the assembly and soldering of the final prototype. Additionally, he worked on the coding of the animations, and made substantial edits to the animation design tool. He also worked on the documentation. Kaleb assisted in the initial design

of the PCB and subsequent testing. He also codesigned the test bench and circuit. Kaleb also designed the power supply system and created an animation design tool.

The primary project management process used was the Agile design process. As two of the members were familiar with this system, we were able to set up a reasonable sprint schedule. The project consisted of six total sprints, each two weeks long. These sprints were lined up with the biweekly meetings to hold the team accountable for the objectives of the current sprint. The Agile process helped keep the team on task as it forced us to work consistently throughout the semester. We also planned an extra three weeks of time after the sprints finished to work on any overdue tasks.

To reiterate, the two requirements that were not met from the initial proposal were the storing of tricks in a database and the rideability of the board. For the database design, we realized the use of a database was not necessary for the sole purpose of acting as a database. Instead, we used a real-time database which acts as a buffer between the website and the microcontroller. Its primary function is to be the middle ground between these two components, it can be written to by the site and read from by the microcontroller. The second requirement not met was the rideability of the board. While not necessarily unrideable, we did not want to risk breaking the board before video and documentation production. The components needed to rebuild the prototype would have been expensive to replace, and the deck of the board required substantial changes. The components, specifically the LEDs, would also have to be reapplied to the board. This requires not only money, but also time that we felt was not worth the risk for testing.

In comparison to the hardware component, the software component is significantly smaller than the hardware component. We have chosen not to supply the GitHub commits, since the image lacks any real meaningful information. Source code will be provided in the Appendix.

Design Documentation

Design Concept

Initial Design

The design of our project consists of multiple components. The initial design used an Arduino Nano as the microcontroller. This was selected due to its small size, Bluetooth capability, and low power consumption. This piece was then accompanied by a power supply that powered the microcontroller and the LEDs. The LEDs had two types of connections. Ground and power were planned to be connected in parallel, and data connected in series. The connections were going to be linked by a custom PCB that was designed and printed for this project. The LEDS were to be controlled by FastLED, an Arduino library that is for the programming and animation of addressable LED strips. This would then be linked with a web application that could communicate with the microcontroller and Arduino Nano via Bluetooth.

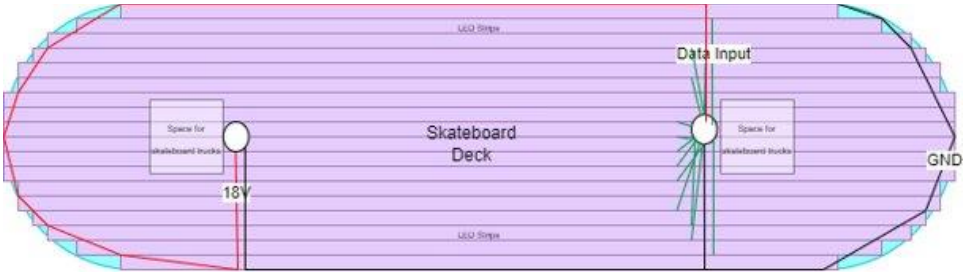

Figure 1: This shows the initial design for the circuit of the LEDs, microcontroller, and power supply.

Figure 2: This shows the profile view of the initial design for the circuit of the LEDs, microcontroller, and power supply.

Final Design

Microcontroller:

The Nano was unable to change the LEDs utilizing the FastLED library. We found this in initial testing and switched to an Arduino Uno as its replacement. In later testing, we found that the Uno is incapable of supporting more than 490 LEDs. In our case, we had 612. This was due to the using all the available SRAM in the microcontroller. To fix this, we switched to a Raspberry Pi, which is similar in size to the Uno, but much more powerful. This microcontroller can control 700+ LEDs. As shown in Figure 1, the power supply lies under the board. This is then powered by an external power bank that provides 5V and 2A to the microcontroller. The two wires from the microcontroller are the ground and data lines that connect to a PCB. These wires are then taped, trimmed, and covered with electrical tape.

Figure 3: This is the microcontroller case powered by an external power bank.

Power Supply:

The power supply was replaced by two different pieces: 9V batteries and a power bank. This design switch made it much easier to power each of the components as each is capable of safely powering their respected pieces. To ensure the LEDs do not reach above the suggested voltage of 5V, we use a buck converter which scales the 9V from the battery to 5V. This piece is also located under the board. This change was made to shrink the power supply and to keep weight down. Figure 2 shows the actual implementation of our prototype. We can see that the battery and the converter are kept in a case and are linked up by soldered wires. These are then covered in electrical tape.

Figure 4: This is the power supply for the battery. It is connected by two cases, one that holds the battery and one that holds the buck converter.

LED Array:

Ideally, we would like to connect all components in parallel to ensure that if a connection is broken, the product is still functional. Unfortunately, the direction of power and ground do

matter in the case of these LEDs. When testing the parallel approach, the LEDs were unable to work in the parallel circuit we created. Due to time, we stuck with the full series approach and ensured that all connections were very sound and protected.

Figure 5: Top view of the prototype

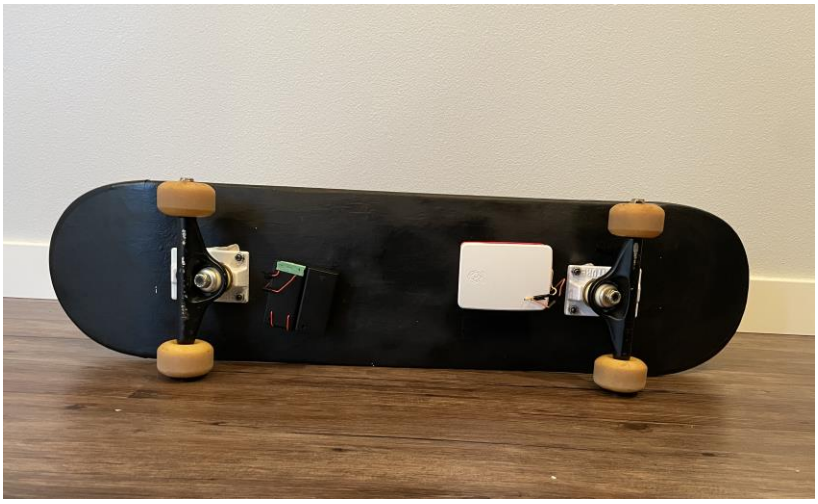

Figure 6: Underside view of the prototype

Figure 7: Side view of the final prototype

Figures 5-7: These figures show the final prototype design. This design as seen contains the necessary controlling and power along the underside of the board. Each component is protected by specific cases and is high enough to where they are a good distance off of the ground.

Hardware and Software Communication:

When working through communication, it was realized that a direct connection via Bluetooth would require the specific device to communicate with the board. This would render the web application useless as the web application is hosted in some location, nowhere near the board. This would mean that a mobile app that someone can download would have worked better in this case. Unfortunately, no group member has any experience with mobile computing or mobile app creation. To work around this, we decided to create a buffer between the hardware and software rather than direct communication between the two. We chose the real-time database Firebase as our buffer. When a user decides to select a trick on the site, the site will send this data to Firebase, the board constantly checks the database for any data put into it and pulls out the new data when the database is updated. Since this requires internet connection, we opted to use Wi-Fi communication rather than Bluetooth as the Pi can do both. This can be visually displayed in Figure 8.

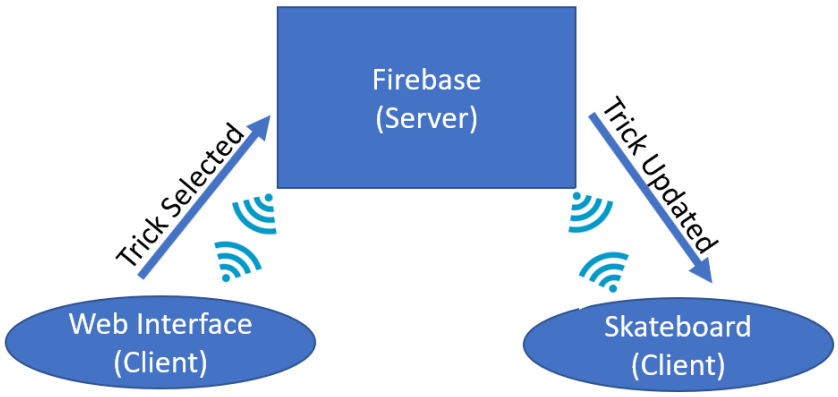

Figure 8: Shows the acyclic connection between the board and the website

FastLED to NeoPixel:

Due to FastLED being an Arduino Library, the switch to the Pi requires a new library to be used. Since the Pi can utilize multiple programming languages, we decided to go with the

Python library NeoPixel by Adafruit. This library functions the same as FastLED where it can control the LED strips under different syntax.

Flexible Circuit Board Design:

Since the board is shaped and curved, it would be hard to implement a standard PCB. Most PCB boards are not meant to be bent and curved. Since we need some flexibility, we decided to create our own custom PCB. These PCBs have three layers within them: one for Power, one for Ground, and one for Data. These connections are done using copper tape. The copper tape sits on top of thick paper and each new copper tape section is connected by solder. Each connection is tested using a multimeter and the layers will be separated and covered by electrical tape to ensure no shorting. These are created in a large quantity, as we need many different shapes and sizes that can fit on the board properly. Figure 9 shows the actual image of each type of PCB. The reason for the different types is due to the strip separation. These separations vary across the board because of the shape of the skate deck. An LED strip will be either shorter or the same length as the adjacent strip. To properly fit the LED array on the board, these needed to be sized in this specific orientation.

Figure 9: These are the shapes of the custom PCB that are responsible for the connection between each strip

The second PCB design is located under the LED array. Figure 8 shows the actual implementation of this second PCD. There are 4 total connections needed that come from the power supply and microcontroller: the battery power, battery ground, microcontroller data, and microcontroller ground. These follow the same approach using copper tape. This PCB links the components under the board with the LEDs. Figure 10 shows the real-life design used on the board. We can see that there are 4 strips that run under the LEDs. Two of them run to the battery power and ground, and the other two run to the microcontroller data and ground. A small section is cut from the LED array, giving access to one end of the strip. This strip is then soldered onto the custom PCB.

Figure 10: This is the PCB located underneath the LED array. This is responsible for connecting the LED array with the power supply and the microcontroller

Website:

Due to the large amount of time spent on the Hardware, we decided to make one page as our web application. Though React includes many helpful and creative features, we realized our project does not need to utilize these. We decided to move to GitHub pages. GitHub pages give us a URL that is accessible to anyone on the web. The pages can easily be made using HTML and JavaScript scripts with the help of CSS. This greatly reduced the amount of time needed in software while preserving functionality and good hardware software communication.

Analysis of Possible Solutions and Tradeoffs

Alternate Design 1: Creating a mobile app

This was one of the choices mentioned in the above section. For this design approach, rather than creating a web application, we replace it with a mobile application. This would have enabled a user to connect to the Microcontroller using their phone and establish that communication rather than a middleman. This would not require a user to be connected to the internet as well. Since the group has no experience with mobile app development, we decided to skip this entirely as it could not have been finished in time.

Alternate Design 2: User Customization

To ease the creation of images and animations, a tool was created to visualize the exact pixels that could be lit up on the board. The tool was able to be interactive and provide code that can be used for the visual. This tool could have been used to enable user customization of the LEDs. This would give the user total control over what they can display on their board. Unfortunately, this solution was not implemented due to the timing of this realization. If this was thought of earlier in the design process, the tool could have been integrated nicely.

Alternate Design 3: Ground and Power in parallel with Data in series

Rather than connecting all of the LEDs in series, attempt to split them up. When testing, we ran into the issue of direction of current mattering. If we oriented the LEDs in the same direction, we may have been able to avoid this issue. This solution was thought of after the custom PCB idea which disregarded this possible solution.

Alternate Design 4: Raspberry Pi Pico

Since the change from Arduino products to Raspberry Pi was short notice, we did not have time to attempt to test the Pico. The Pico is shaped similarly to the Nano but has much more computational power. This means that in theory, this microcontroller would be able to function just like the Pi but shrink its size significantly.

Alternate Design 5: Power Supply Design: Power Switch

In the final prototype, the 9V battery will continuously power the LEDs, draining the battery until either taken out or completely drained. A good solution to this problem is to add an electrical switch to the battery case. There was actually a switch located on the original case, but its location happened to be on the face of the case and not the side. This means that the switch Velcro-ed onto the board and is inaccessible to the user. The new switch would be located on the side of the case for easy access for the user to turn on and off.

Constraints

These are constraints from the proposal:

1. The system shall be responsive to user input
2. System power supply must be protected and safe
3. The board must be ridable with all modifications on it

4. The board must be relatively light weight and maintain a similar normal skateboard design so around 14-16 lbs.
5. Board must be less than $400 to maintain budget as well as positive cost efficiency

Standards

These are our standards from the proposal:

1. NeoPixel
2. GitHub Pages
3. FIrebase

NeoPixel is the library utilized for controlling LEDs. The web application is hosted off of GitHub Pages, which made the app easily accessible to users with internet connection. The database used is Firebase, which requires a google account.

Architecture

Below are some schematics describing the design of the hardware of the system:

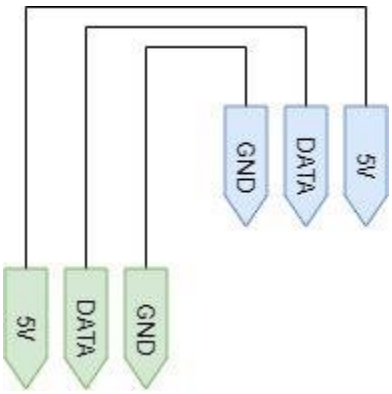

Figure 11: This is the schematic for the LED PCB. This shows the circuit that connects two LED strips

Figure 11 shows the schematic for the PCB designed by us to connect pairs of LED strips. The final prototype consists of about 17 of these PCBs. The PCBs are made of copper wire and solder and separated by electrical tape to avoid cross connections and shorting.

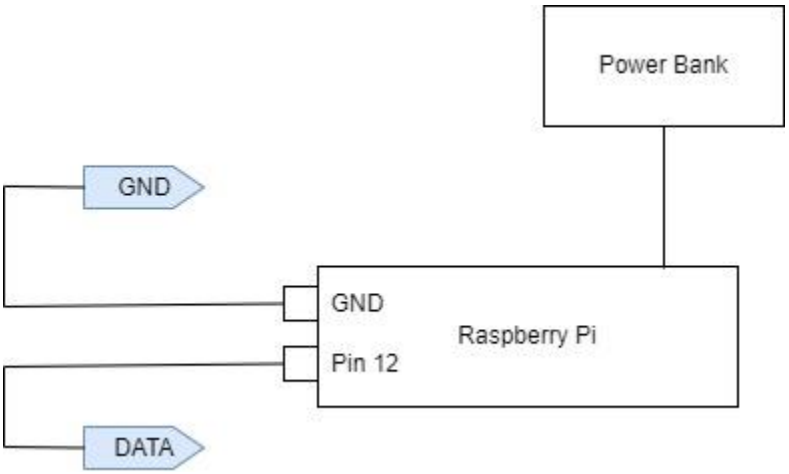

Figure 12: This is Raspberry Pi schematic. This shows what connections are being made between the LEDs on the op and the microcontroller under the board

Figure 12 shows how the Raspberry Pi supplies its ground and data to the board. Since the LEDs run off of the battery, there is no need for the Pi to supply power to the strips. For data synchronization, the ground needs to be a part of the battery ground which is why we have a connection there. As mentioned before, these two lines are then connected from under the board to another custom PCB that lies under the LED strips.

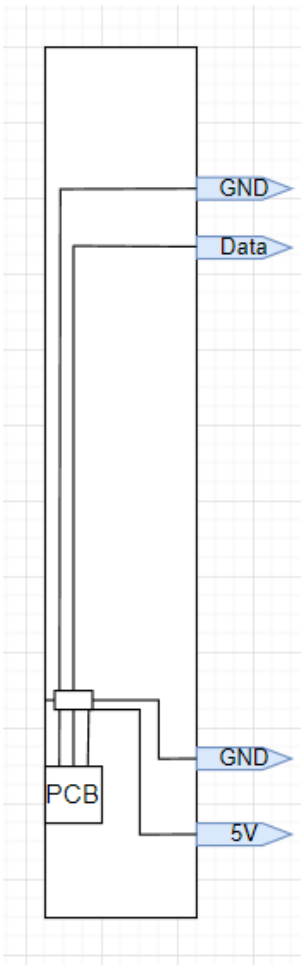

Figure 13: This design is the PCB that connects the board with the power and controls located under the board

Figure 13 shows how power, ground, and data are connected to the circuit. This PCB is responsible for providing connections from the underside of the board to the LEDs. This PCB is located under the LED strips. It is covered in electrical tape and is soldered onto the first PCB located on the right side of the board.

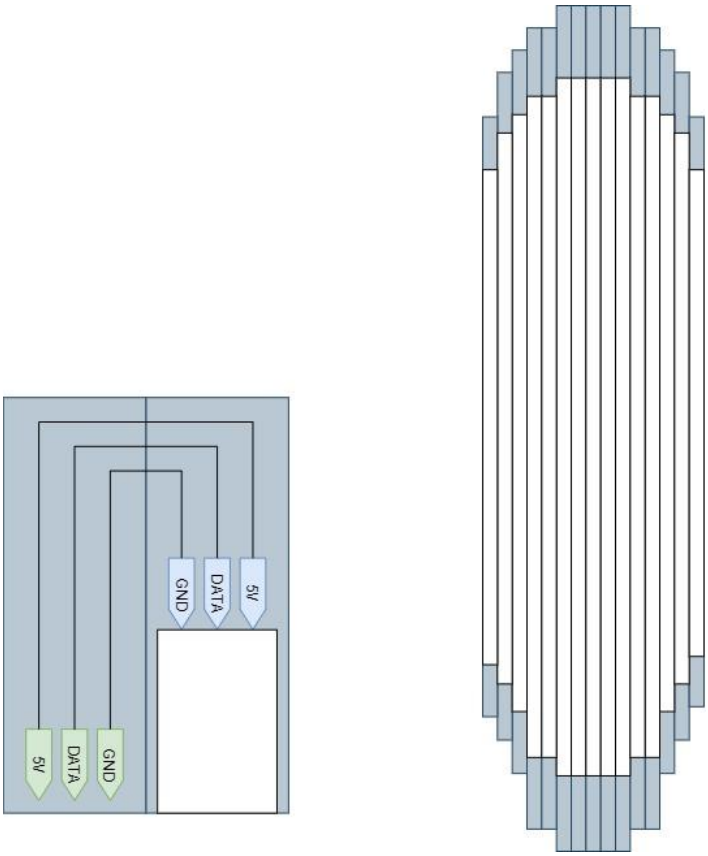

Figure 14: This shows example of flexible PCB Green on led strip and blue another LED strip

The bird's eye view of the board is shown in Figure 14. This shows how the LEDs are shaped as well as the 18 PCB connections at the end of the strips. These PCBs are highlighted in blue and are covered in electrical tape. The left part of the Figure shows the exact connections that occur under the strips.

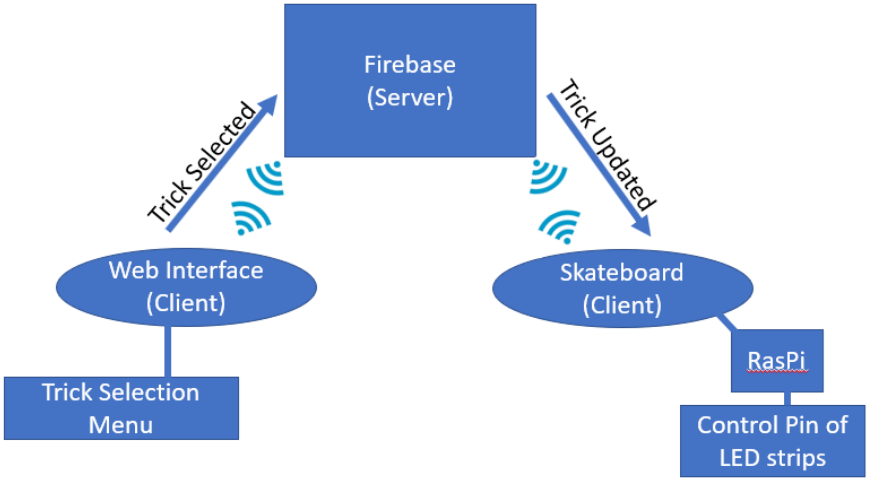

Figure 15: This is system overview of the hardware and software connection

The system works as an acyclic path in communication. As Figure 15 shows, the system starts with selecting a trick from the web application. Once the trick is selected, the data is sent to Firebase. While this occurs, the skateboard is constantly reading from the Firebase and looking for any changes. Once the change occurs, it pulls that data out from Firebase and uses it to select the proper LED lights for the selected trick.

UI/UX

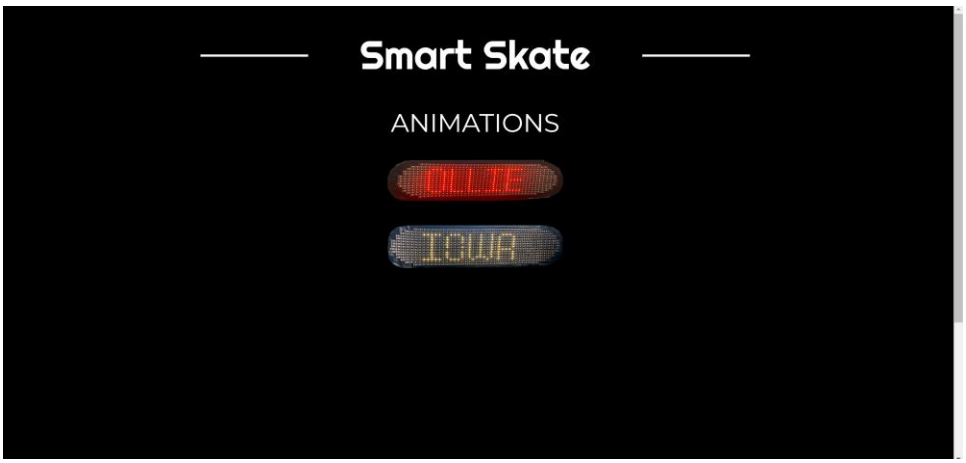

Figure 16: This is the final UI of the Website

The website utilizes GitHub pages as its host and is built on HTML, CSS, and some JavaScript. Figure 16 shows the final prototype of the Website. Here a user is able to select a preset that can then be shown onto the board. The user can do so by clicking the image of the preset.

Maintenance

For the software to continue working as intended, Firebase must be upkept and paid for. The user must change batteries if the board were to die. The device also requires to be connected to the internet, so the user must be able to connect the board to the internet.

Possible Changes

Some of the solutions mentioned in the Analysis of possible solutions and tradeoffs can be possible changes to the design. For example, if found earlier, the switch from a Pi to a Pico would have been doable. Another possible design change is utilizing rechargeable batteries rather than normal 9V batteries. This would limit battery waste and avoid unnecessary maintenance duties for the user. Ideally this would work similarly to today's electric scooters, where a user can plug in the board to recharge it.

Test Report

Traceability Matrix:

Req No	Req Desc	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12

1	User shall control LED		**x**	**x**	**x**								
2	User can see animations	**x**											
3	Animations can be changed	**x**	**x**	**x**									
4	Display can be stepped on					**x**							
5	System can access database											**x**	**x**
6	System establishes Hardware and Software connection						**x**	**x**	**x**	**x**	**x**		**x**

Req No	**Req Desc**	**T13**	**T14**	**T15**	**T16**	**T17**	**T18**	**T19**
1	**User shall control LED**	**x**				**x**		
2	**User can see animations**	**x**					**x**	**x**
3	**Animations can be changed**	**x**					**x**	**x**
4	**Display can be stepped on**							
5	**System can access database**				**x**			
6	**System establishes Hardware and**		**x**		**x**			

	Software connection							

Test ID	**Test Description**	**Expected Outcome**	**Observed Outcome**
1	Microcontroller can utilize NeoPixel	LEDs will light up	LEDs lit up red
2	Microcontroller can control more than 3 LEDs at a time	Microcontroller controls the small test strip	5 LEDS lit up red
3	Microcontroller can control more than 2 LED strips that have been custom attached	Microcontroller controls custom made LED strips	Microcontroller was able to light up 4 strips red
4	Microcontroller can control entire LED array	Microcontroller can turn on all 611 LEDs	All strips and LEDs lit up red
5	Microcontroller can function under board and off external power supply	Microcontroller powers on off of power bank	LEDs were powered using external power supply
6	Custom PCBs establish strong connection with each LED strip	No issues in connection when turning LEDs on	LED strips were able to light up due to the soldered connection on the PCBS
7	Underlayer Custom PCB can hold 4 connections with no issue	Ground, Data, and Power are all linked to the LED array	LEDs were able to light up using

			underlying custom PCB
8	Skateboard is modular and can detach LED mat if needed	LED Array and other components (battery, microcontroller) can be taken off	LEDs were able to be detached
9	Skateboard is within reasonable weight (under 20 lbs)	The skateboard weighs about 18 lbs.	The skateboard weighed under 18 lbs.
10	Parts are well integrated within the board and do not obstruct riding	Microcontroller and power supply do not hit the ground or stick out	The parts were placed under the board and do not obstruct the user
11	Website can send information to database	When the user presses button, data is sent	LEDs light up when user presses button on website
12	Microcontroller can pull information from database	Microcontroller reads an int from database	Microcontroller receives correct input made by the user
13	When a user selects and submits trick, LEDs change	Microcontroller detects data and displays accordingly	LEDs light up when user pushes button on website
14	Microcontroller can connect to the internet	Microcontroller connects to hotspot	The microcontroller was able to connect to the internet

15	Website is hosted and can be accessed anywhere	GitHub pages allows the hosting	Public URL works on multiple devices
16	Website delay from submitted changes are within reasonable time (maybe between 1-2 seconds)	Timed change submission is under 2 seconds	Delay is about 1-1.2 seconds
17	Skateboard updates when Firebase updates	Skateboard changes when user inputs a trick	Skateboard displays proper trick when Firebase updates
18	LEDs change patterns according to the animation specifications	When Ollie is selected, the Ollie animation occurs	Ollie is successfully displayed

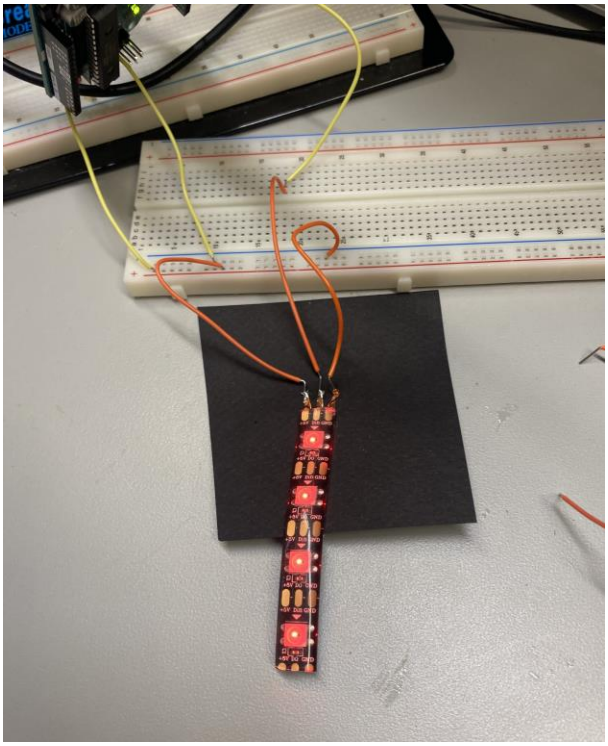

Figure 17: Initial LED Library Test

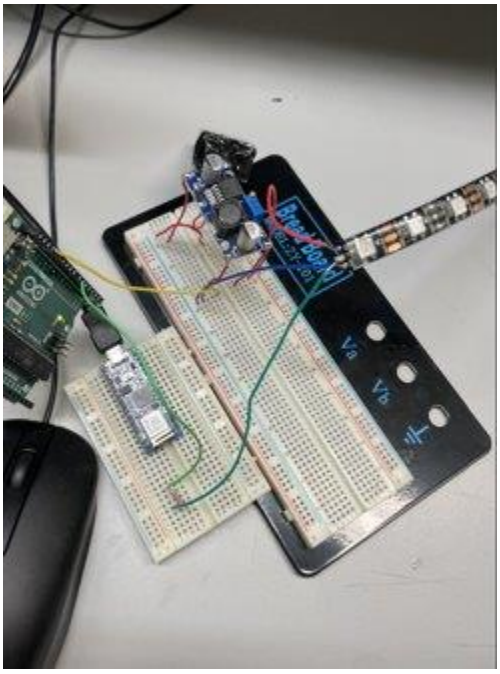

Figure 18: Initial Power Supply Test

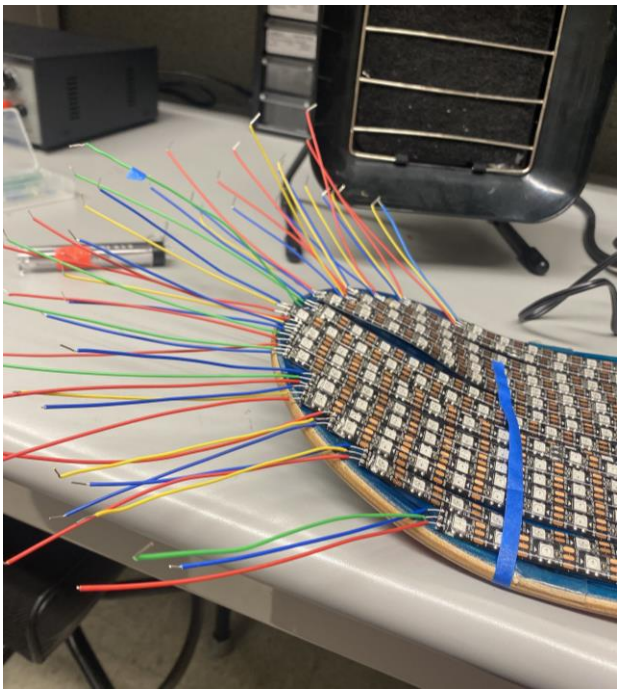

Figure 19: Initial set-up of full function LED test

Figure 20: Full function LED test

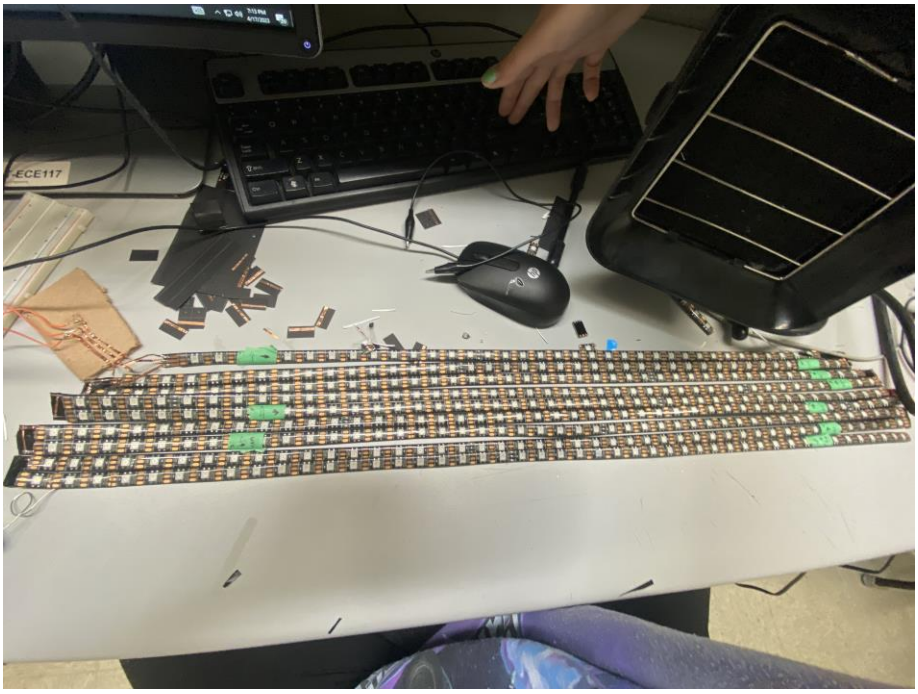

Figure 21: Circuit Direction and shape test

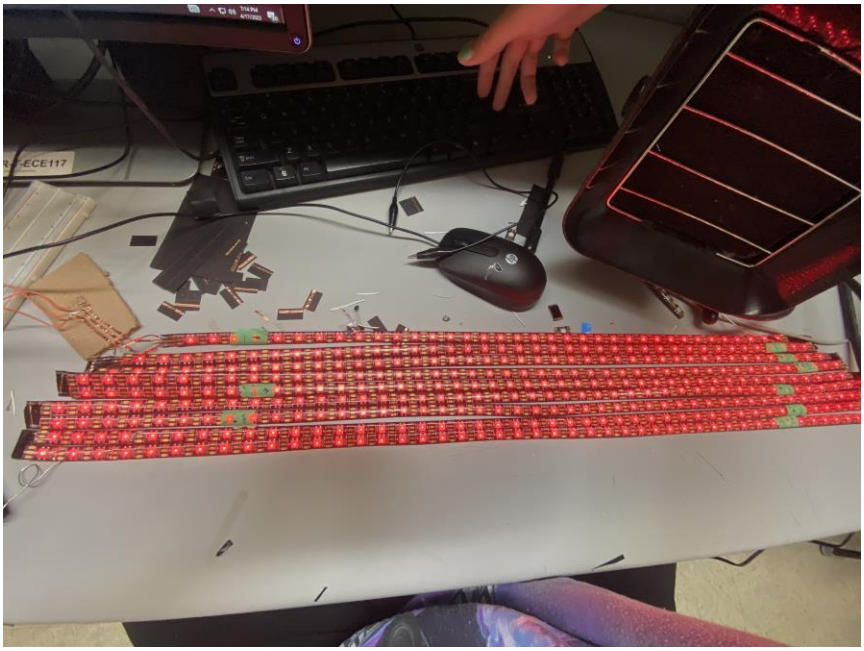

Figure 22: LED connections via PCB test

Figure 23: On-board LED connection test with all components connected

Most of the testing done was on hardware. As this project was heavily hardware oriented, we needed to make sure all hardware was tested and connected as best as possible. This gave us a series of tests that were built from the ground up. We first need to test the library for the functionality, then slowly work our way up to all the components acting as one cohesive unit. Figure 17-23 are some images of the test that were done on the system. Once all hardware testing is complete, software testing can begin. The test done on software was focused on the hardware software connection. These needed to be consistent in order for the prototype to function properly.

Appendices

Source code and Gantt chart is provided in a separate zip file. Gantt chart is located here due to its size.

Web path for the user interface: https://skate-site.github.io