TypePaste: Design Document

Ben Haswell - IMS 253, Section A

// Project Overview

TypePaste is a macro device that helps users save time and effort by allowing them to store text snippets and paste them to their connected computer using physical buttons. Users can easily input and retrieve text, simplifying their workflow and improving their productivity.

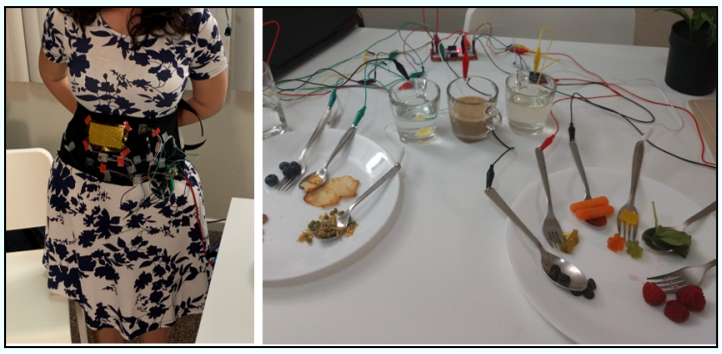
After plugging the device into their computer via USB, users can enter up to four text Snippets, tabs and newlines included, to one of the four buttons on the device using the digital interface, built with HTML/CSS/Js. Once the user enters and saves a Snippet, a serial connection is made between the app and the Arduino connected to the device, storing the Snippet into the associated button and illuminating the associated LED green to indicate a successful storage. Users can also modify the name of the Snippet in the app to fit their needs and preferences.

TypePaste is designed to simplify workflows and create a more productive world by saving the digital labor of typing the same things over and over again. It was designed for programmers, but can be used by anybody who has frequent texts that don't necessarily need to be typed out every single time. From code to chat logs, email formatting, or just simple keyboard shortcuts, TypePaste makes your life easier.

// Brainstorming and Idea Generation

A living timeline detailing the brainstorming and idea-construction process that led to the current version of the project.

NIME/CHI/DIS: What Not To Do

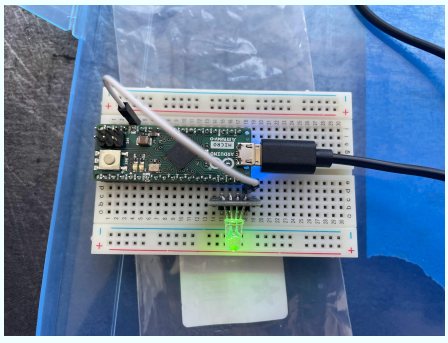


One of the first exercises we completed was a look through an archive of interactive devices and art installations. I decided to look at an archive called “Exploring Food based Interactive, Multi-Sensory, and Tangible Storytelling Experiences”. While most projects in this archive were interesting and thought provoking, I was skeptical about a few of them being labeled as ‘interactive’. Some of the projects were spiced up to be ‘interactive’, and ‘multi-sensory storytelling experiences’, but I don’t feel that they quite reached the heights they were set out to reach in that regard.

What I got from this activity was the understanding that I was more interested in the actual functionality of the projects - how did they work? How did the sensors communicate with one another? It was this curiosity which led me to feel that I would be more interested in building a device which leaned away from the artistic and interpretive and more towards the functional and utilitarian.

RGB Fading

During our first couple weeks of learning about Arduino, we had an assignment where we were tasked with lighting an LED Module (with RGB 0-255 capabilities) and having it smoothly fade between the three colors instead of the digital outputs (on and off, 0/1) we were used to.

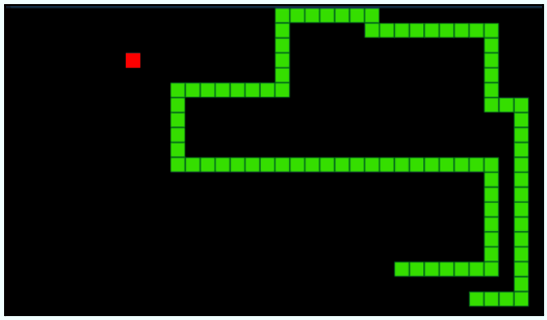


The finished product was something that seemed like a piece of cake to an inexperienced (Arduino) user, physical wiring and connections included, but was actually more complex in back end implementation than you might think. It was nothing crazy, but it took more lines of code than I thought were necessary to

accomplish a result which seemed simple on paper. It continued to spark my interest in the idea of encapsulating the device and hiding implementation details from the user.

Computer-to-Microcontroller Connection

One of the biggest inspirations for my project was well into our Arduino learning where we were tasked with building a 'controller' for the famous "Snake Game", in other words, wiring up four buttons and writing the code so that each button corresponded to a control in the game (up, down, left and right) so that you could use nothing but the controller to play the game.



This was our first time being exposed to Arduino Libraries, in particular the Keyboard Library which allowed for the Arduino microcontroller, if connected to a laptop, to send keyboard strokes as output, being received and executed by the computer. This greatly expanded my understanding of what would be possible with my final project, as I was not aware that some sort of data connection between a computer and an Arduino microcontroller was in our scope. It was from this point on that I decided I would be most interested in finding some way to establish not just connection, but *communication* between a computer and an Arduino, having them send and receive data from one another and respond to it accordingly.

After some more brainstorming, considering that the final project needed to target a specific group or community in function (game controller for physically disabled people, braille keyboard, etc.), I came up with the idea of making a device for a community I was already part of, one I could see myself using.. The idea for a device which programmers (or 'digital writing enthusiasts') could use to speed up their work and increase their productivity by saving time with Arduino-pasted snippets was born.

Rapid Prototyping: Less is More

In the beginning stages of our planning, we did a class activity where we were each given a concept of a device and were supposed to draw it and try to convey its functionality with just the drawing - no words, numbers, or 'obvious' iconography was allowed.

My team had a particularly challenging device to draw (compared to that of the other group), so needless to say we went all out trying to add as much detail that would convey the device's weird and non-traditional

function. When it came time to present, however, the other team struggled greatly with understanding and coming up with an idea of its functionality, ultimately pitching an idea which couldn't have been more off from the device concept we were tasked with drawing.

What this exercise reinforced for me was the truth that presentation is, arguably, the most important aspect to consider when designing a device (or any other kind of interactive object for that matter). When it comes to conveying information, less is more, and in order to have the smoothest user experience possible you must consider what kinds of things are absolutely necessary to display, and what the best way to display its associated information is.

Buttons, Buttons, Buttons

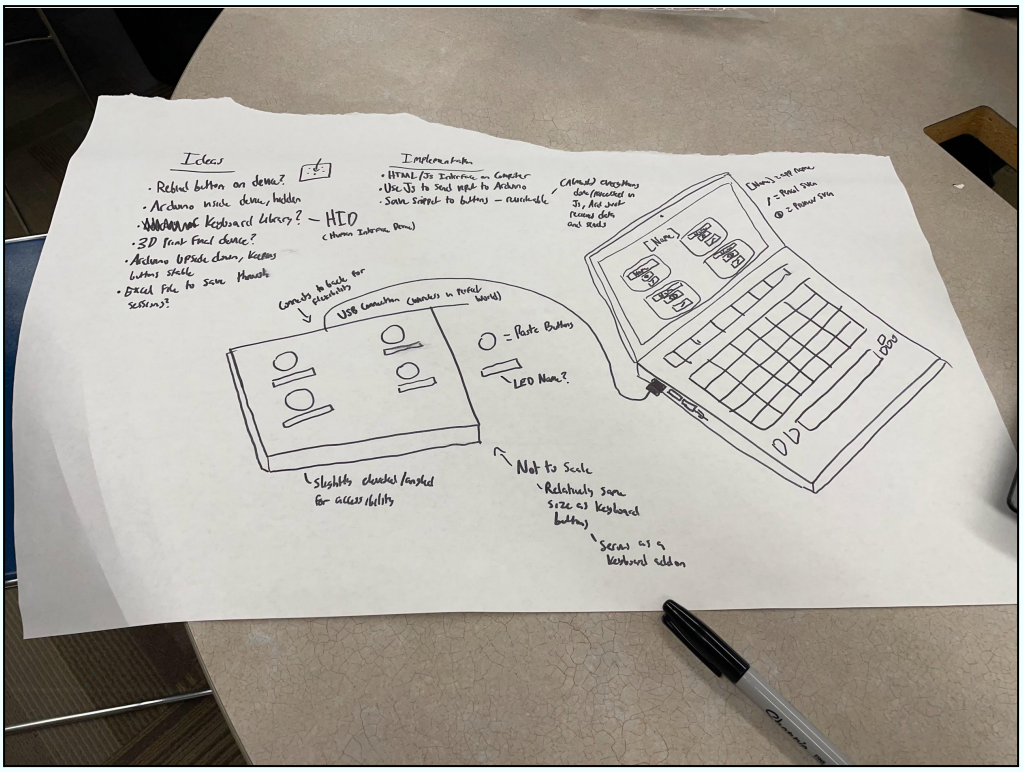
As a side note, one of our days was spent learning how to correctly solder wires together. During this activity, I was able to solder a bunch of things, but more importantly had the opportunity to see what kinds of buttons would be available for me to use for my final project.

There were many different kinds of buttons, each of them with their own characteristics (colors, sizes, shapes, pressure, stickiness, functionality). I made a point to analyze what kinds of buttons would be best fit for my project, eventually landing on the arcade-style buttons since they were satisfying to press, on the larger side to compliment the standard keyboard experience, and would work well in regards to dimensions/sizing with the design I had in mind.

// Sketching and Construction

A chronologically-sorted series of project sketches and models highlighting the evolution of the concept into a concrete artifact.

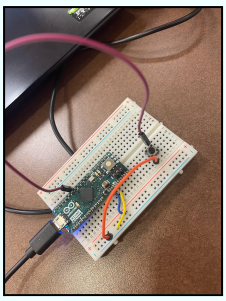
Initial Sketch



In my initial sketch, I made a point to highlight every detail of the device's design, both physical and digital, possible, without worrying too much yet about the implementation. I wanted a design which complimented the regular keyboard experience: something you could just plug into your computer and have ready to use.

I went for a slanted design to make buttons easier to access when typing (instead of having them flat, which would make them harder to press). I sketched out the idea of having some kind of indication on the physical device (an LED) which would inform the user of the name of the snippet set to the corresponding button, but was unsure of whether or not this would be in the budget (both time and money wise). I also decided, in this stage, to implement the digital interface using a vanilla HTML/CSS/Js application, since I was very proficient in making such apps and saw it as the most logical way to perform such a communication, creating an 'app' people would download to their computers.

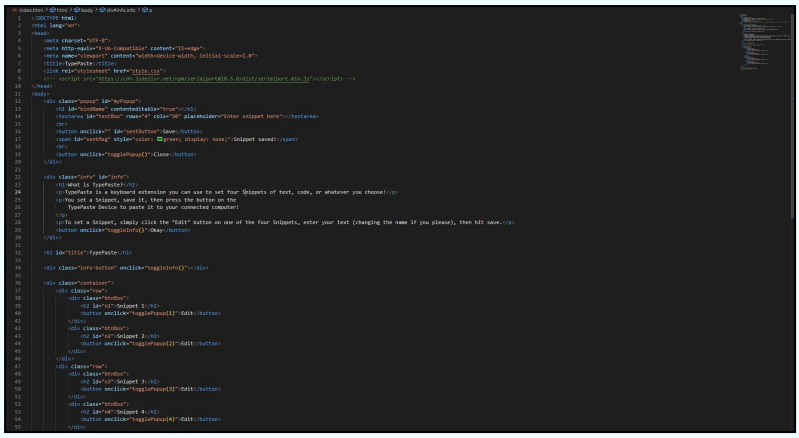
Positioning



In beginning my construction of a prototype, I quickly ran into a thought puzzle in how I would wire up the buttons inside the encapsulation, taking into consideration where the actual Arduino and breadboard would be located and how to best optimize that space. I was still pretty new to wiring, so I had not yet discovered the trick of sharing ground connections or disassembling/reassembling wires to best fit the space inside, but that was to come.

Digital Skeleton

At this point, I had figured that the most important step to take would be to make sure that my project is actually possible. Like mentioned earlier, we had not gone over serial communications between a computer and Arduino, let alone a locally hosted, static JavaScript file. I had sketched out my idea, now it was time to start building.



I wanted to tackle the digital interface and functionality before going any further, so I decided to set up a minimalist 'skeleton' of the app interface I thought would be most effective. I followed along closely with my sketches, creating four distinct modules to associate with the four buttons on the device, each with their own names and edit buttons. I had not yet implemented the functionality, just put my design onto paper, so to speak.

I also decided on the branding and name of the device. Inspired by Arduino's color palette, the app background was a bluish-green to white gradient with white boxes and black text. I came up with the name for the device: TypePaste.

Coding Communication

The biggest breakthrough in development was, by far, establishing the previously uncharted communication between the computer and the Arduino via USB serial connection. Using JavaScript's built in 'navigator' and

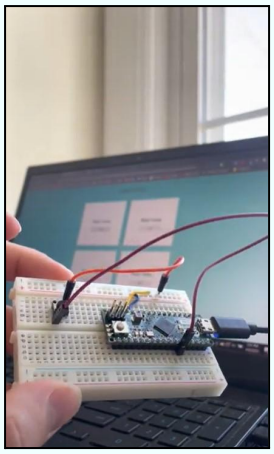
its serial functionality, I was able to open the port with a 9600 baud rate (as well as other specifications like dataBits, stopBits, parity, and flowControl) to match with the connected Arduino and establish the connection. Then, I set up a function which would take a text input from the front end and send it through the serial connection to the Arduino, then read it bit by bit back to the app through a TextEncoder object and the port's writer functionality. Once the data returned in byte form, I used a TextDecoder to read it character by character, then print the output to the console.

```
// Testing
async function sendTextToArduino(id) {

    // If port is not already open, open it
    if (!port) {
        port = await navigator.serial.requestPort();
        if (!port.readable) {
            await port.open({ // open before function - trying to open when already open (repeated inputs will cause error)
                baudRate: 9600,
                dataBits: 8,
                stopBits: 1,
                parity: "none",
                flowControl: "none",
            });
        }
    }

    // Get snippet from text box
    const textInput = document.getElementById("test").value;

    // Encode the data
    const encoder = new TextEncoder();
    const data = encoder.encode(textInput); // worked with "Hello, Arduino!", not textInput

    // Write to port
    const writer = port.writable.getWriter();
    await writer.write(data);
    writer.releaseLock();

    // Read from the port
    let decodedString = "";
    const reader = port.readable.getReader();
    const timeout = setTimeout(() => {
        console.log("Timeout reached, exiting loop");
        reader.cancel();
    }, 1000); // 1 second timeout

    while (true) {
        try {
            const { value, done } = await reader.read();
            if (done) {
                break;
            }
            if (value) {
                const decoder = new TextDecoder(); // Decode array
                const decodedValue = decoder.decode(value);
                decodedString += decodedValue;
            }
        } catch (error) {
            console.error(error);
        }
```

Originally, this caused problems because each character read would come back as a String preceded by "Received: ". To solve this, I created a regex function which would filter the received data character by character and return the character without the "Received: " or any newlines which resulted from lag or connection issues, adding those characters to a final string.

This was pretty much the deciding factor of whether or not this project would be possible - stepping into new waters and seeing if communication would be possible through nothing but a USB and a JavaScript file - and it was a huge relief once I finally saw the correct text, without breaks or errors, being sent to and from the Arduino and printed to the console.

Finally Functional



Successfully save snippet to Arduino and paste it using keyboard library
After the communication breakthrough, making the project a functional reality was a matter of allowing text to be entered in the digital interface (with new lines and spaces included), sending it to the Arduino with an id (specifying which of the four Snippets it was to be saved to), filtering out the id using a temporary String, and awaiting a pushdown of one of the buttons to print the associated String.

It was really exciting to see a stripped down but functional version of the project

work as intended, as it confirmed that I was on the right track and that all I needed to do was implement the rest of the buttons and user interface, prototype and squash bugs, and use my learnings to design the best version of the device as my final.

Cardboard Crafting

With proven functionality, I was focused on creating an enclosed (hiding implementation/wiring) cardboard prototype I could use to gather user feedback.

My prototype included two buttons and an LED for the first button; I decided I was going to implement LEDs for each of the buttons which would be red if there was no Snippet saved and green if there was so that it's easy for users to remember which ones they've set in each session and for more visual clarity.

I also finalized the design for the app UI as well as its functionality, including interactive elements like hovering boxes, prettier buttons and titles, and editable Snippet names.

// User Testing and Iteration

Notes and iterative adjustments taken from user feedback inside and outside the lab.

It's The Little Things

Due to time constraints, user testing and feedback only came from one person - my roommate who also happens to be a regular programmer. I described to him what my project was, the current state of its functionality (and encapsulation), where I was planning on heading, and asked him to give it a test run on one of his assignments and give me any feedback he can think of.

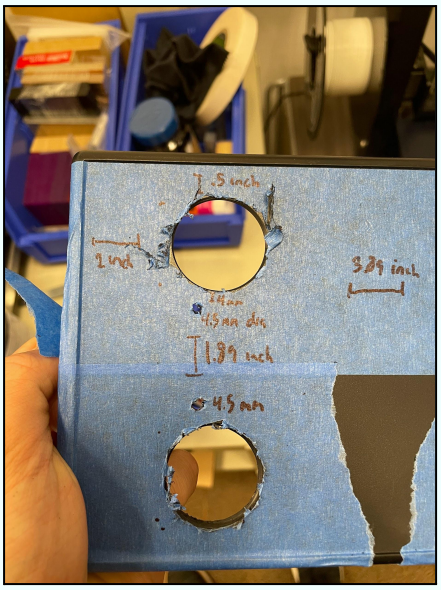
The overall reception was positive, as he told me it worked well and could potentially be very useful to programmers in its final form. He had some nitpicks here and there which I took into consideration, but the most valuable piece of feedback he gave me was to implement features in the text area where users enter

their snippets to match common shortcuts on programming IDEs like tabbing to add a tab character, shift-tabbing to remove two spaces, entering to insert new lines, and more. I hadn't fully thought about this before, but it made so much sense to me that a device targeted towards programmers should take into account the muscle memory and expectations they already have, making the user experience as seamless as possible. I was extremely grateful for this input and went to implement it right away, adding a secondary JavaScript file to the app purely for styling and UI-side functionality.

// Polishing and Refinement

Log of adjustments and final touches made before the final presentation.

Encapsulation



The final encapsulation for the device had arrived, and I was shocked by how close it was to the design I had sketched out and planned on building/3D printing myself; complete props to my professor for finding exactly what I needed.

I began by measuring out the dimensions of the box, the exact distances between each of the buttons and the LED modules. I did this by taking measurements of the items and mapping out the optimally spacing for a symmetrical and consistent layout. Once these measurements were taken, I labeled the locations of each of the holes to drill with diameter measurements and, with my professor's help, made it happen. What I was left with was an encapsulation with perfect holes for all eight implements (4 buttons, 4 LED modules).

Building The Thing

Building the final device was a difficult and rewarding process. There were some imperfections in the holes that were drilled, so several of them needed hot glue in order to secure their spot and not have them fall inside upon being pressed. Little to my knowledge, however, I had accidentally squirted hot glue into the springs of one of the buttons, giving it a horrible case of sticky keys: taking 10-20 seconds to rise after being pressed. With work from a heat gun, I was able to melt the glue and use pliers to dig out most of the glue from inside; my professor was kind enough to finish the job after I had gone home, so that issue was fixed.

There was also an issue with spacing, as I had dozens of wires with no clear way to organize and compress them so that they don't interfere or push against the encapsulation. As my professor said, there is almost nothing that can't be fixed with hot glue and some other sticky implements. I used sticky clips to organize the branches extending from each LED module and ran them along the sides of the encapsulation so that they stay out of the way of the buttons and don't push against each other and risk internal failure or circuit blending.

In a similar vein, the Arduino and breadboard themselves took up space that was going to be hard to account for. The solution was actually working against gravity by hot gluing them to the top of the enclosure so that the 'floor space' could be reserved strictly for cable organization and so that no implements ram into the top like they would have if the breadboard would have stayed on the base of the enclosure.

With all implements in place and all circuity problems fixed (trust me, there were lots) I was proud to have officially turned my idea into a reality.

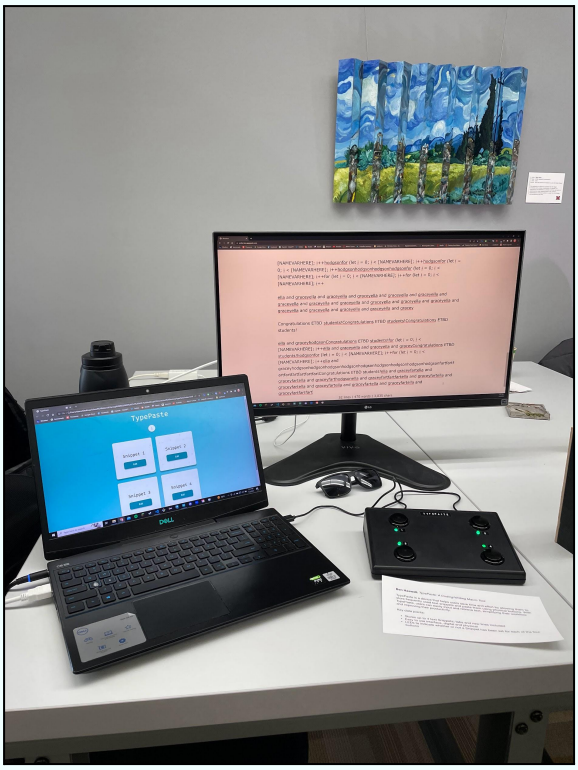
Still Confused?

As a precaution for the possibility of me not being present at my station during the expo as I make my way around to check out the other projects, I implemented an "info" button which describes the device and how it is used.

// The Expo

Showing off TypePaste at the ETBD Expo was a great success. All kinds of people came up to my station and seemed genuinely interested in what I had on display.

I enjoyed watching people enter their own Snippets and paste them into the provided text box; it was interesting what kinds of things people decided to save. For the most part, they would be simple things like “hi!” or “your mom”, but it was a relief to have some fellow programming enthusiasts approach my station and save/paste simple Java methods and code for loops; these people seemed particularly pleased with the device, giving me a sense of great satisfaction as these were the kinds of people I built the device for. Many of these people told me that they would like to buy one, or that they would definitely see themselves using it every day. I even had one of my professors from another class come up to me and tell me that he would use it to save time writing the same emails to students over and over again. These were all nice to hear.



I had one or two people come up to me and say things like “isn’t this just four extra copy-paste buttons?” or “doesn’t something like this already exist?” I go further in detail on this point in my retrospective, but I simply told these people that yes, functionally TypePaste is something that ‘already exists’, but in a new and optimized form. Explaining my design choices and how the device fits into people’s workflows seemed to quell their suspicions.

Overall, I am satisfied with how the expo went. I may not have had an interpretive multimodal art piece, a song made from nothing but trash, or a skeleton that has a heart that beats faster when you hug it (these were all creative and super interesting projects my fellow classmates presented), but I had a device that was functional, utilitarian, and came out exactly how I had wanted it to.

// Retrospective

I went into IMS 253: Building Interactive Devices with high expectations. With the same professor, I had previously taken Media Archeology, a deeply analytical class based around studying the state of media and technology in the past and present, and what it means for the future. That class culminated in a final project I was extremely proud of and still think about today. Needless to say, I was going in expecting a similar experience.

My expectations were both met and not met. My professor, once again, gave her all in every aspect of the class and remains my favorite professor at the university by far. However, I found this class to be the more hands-on variant of Media Archeology; we were still making interesting things with unusual materials, but I felt that this class had a greater focus on the making than the thinking (philosophically), if that makes sense. This was a double edged sword, as I went through each brainstorming session with this mindset; I had to leave the class with a project that was more practical than analytical. I believe I accomplished this, making a device for a community I'm a part of that I could actually see myself adding to my workflow. However, I can't deny a slight feeling of regret with choosing this kind of device.

Don't get me wrong, developing and building TypePaste was an incredibly rewarding experience and I am extremely grateful for the hands-on experience and learning in all directions which resulted from the process. I just feel that we learned about so many different kinds of sensors, so many different possibilities with Arduino, so many creative ways to use this technology, and I used four buttons and four LEDs to make a 'copy-paste device'.

If I could take this class again, I would try to find some way to test my limits further, reaching into uncharted territories far deeper than learning how to communicate between a computer and a microcontroller via serial connection. I would use all kinds of sensors to make a project that is both useful and thought provoking, rather than focusing too much on the former. Luckily for me, everything I learned is still with me and will be with me for years to come, so if I do end up wanting to scratch that itch as a personal project, I'm confident that I would be able to.

I feel that this experience gave me the chance to learn about creating in both physical and digital spaces, and how to weave the two together to create something interesting. Beyond that, it has taught me how people interact with technology, and how I can use that information for good.