Group 31 Final Report

Noah Feinberg - nbf2, Ashwin Mukund - amukund2
TA: Charles Ross

Abstract

The operation of a computer from a distance or on the go is not optimized causing inefficiencies while working with computers. The original solution was to carry only portions of the keyboard that were needed. Our solution was to create an application that could operate the computer from a distance using Bluetooth technology, removing the need for any hardware. We redefined the scope of the problem, since the original group aimed to convenience frequent travellers. We instead opted to focus on teaching applications and modified the original design to suit this new problem.

Second Project

Problem Statement

In most cases, teachers are using the technology to teach their students using screens and powerpoint as a medium. However this inhibits the ability for the teacher to interact directly with the students, since they are required to be near their computer. Meanwhile, smartphones, which are not generally used in the classroom, are instead kept in pockets or bags by both teachers and students. However we believe it is possible to optimize the teacher-student experience by simply improving the mobility of the teacher through the use of their smartphones.

Solution

Our solution is to use the smartphone found in the pockets of any teacher, to act as a keyboard or a mouse for their computers, allowing them to operate their computer from a distance. Our product allows a user to maintain constant operation of their computer while being a distance away from their computer. This product will enable teachers to engage with their students without compromising their ability to use technology for their lessons. We plan to use a created piece of hardware to serve as an intermediary between the phone and computer, translating the data sent from the phone into a form understandable by the computer’s keyboard driver. The phone itself would have a displayed keyboard and mouse, with its configuration able to be changed to multiple modes to suit the needs of the user.

Group 10’s solution was to create a plug and play modular keyboard in order to operate a person's computer. This keyboard consisted of several small portions of a regular keyboard that could be assembled to make a complete keyboard. The advantage of this solution allowed a user

to only carry the parts of a keyboard required to optimize their workflow. The target of this product was very general and meant to appeal to a large audience.

We opted for a wireless and limited solution in order to optimize the workflow of a presenter for a large venue. In classrooms and lecture halls, lessons involving teachers/professors interacting with their computer require them to face their students while looking at their computer. The teacher is forced to choose to either interact with the students directly or focus on the material presented on the computer. By allowing the teacher/professor to operate their computer at a distance without losing the quality of their computer's connection, we would help their workflow.

Currently, the only type of product that has similar functionality to our product would be a handheld laser pointers/slide clicker. Laser pointers/slide clickers allow for the user to highlight locations with a laser pointer, and flip between slides in a powerpoint presentation[1]. Our design would be vastly extending those capabilities to beyond just paging through slides and documents. On top of the full keyboard functionality, we would allow the user to freely control their mouse.

High-Level Requirements

1. The phone application should be able to operate at a distance of up to 15 meters, allowing the user to operate their computer from anywhere within a moderately sized classroom or lecture hall.
2. The user is able to customize at least 3 tile configurations, such as a mouse, keyboard, and miscellaneous configuration, and switch between them in order to optimize the users workflow when communicating with the computer through the phone application.
3. The phone application sends inputs to the computer in less than .5 seconds.

Physical Design

One of the key features to our product is the multiple configurations that will allow the user to operate the computer. In figure 1 the default configuration is shown with a keyboard, mouse, and number pad configuration. In order to switch between the different configurations a user would simply need to click the appropriate tab located at the top of the configuration. Figure 1 also illustrated the USB connection between the product to the computer and the general direction flow of data from the phone, to the bluetooth module, to the microcontroller, and finally to the computer.

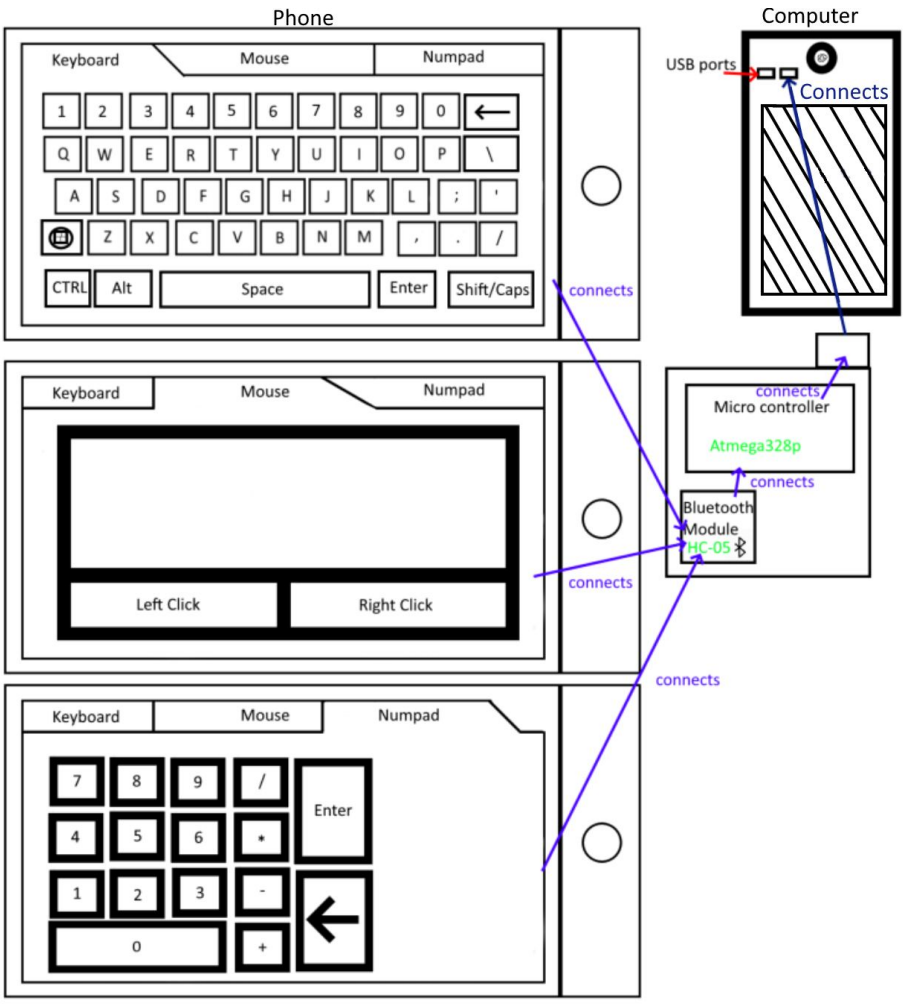

Figure 1. Physical Design Diagram

Circuit

Figure 2 below illustrates the connections between the communication module, microcontroller, and computer. Power for the entire system is received from the vbus of the PCB's USB connection to the computer. The bluetooth module sends data it receives from a cell phone using the TXD pin. On the other hand the RXD pin broadcasts data it receives from the microcontroller. After data is processed by the microcontroller it passes through the USB connectors to the computer using Data+ and Data- pins which are used for noise immunity.

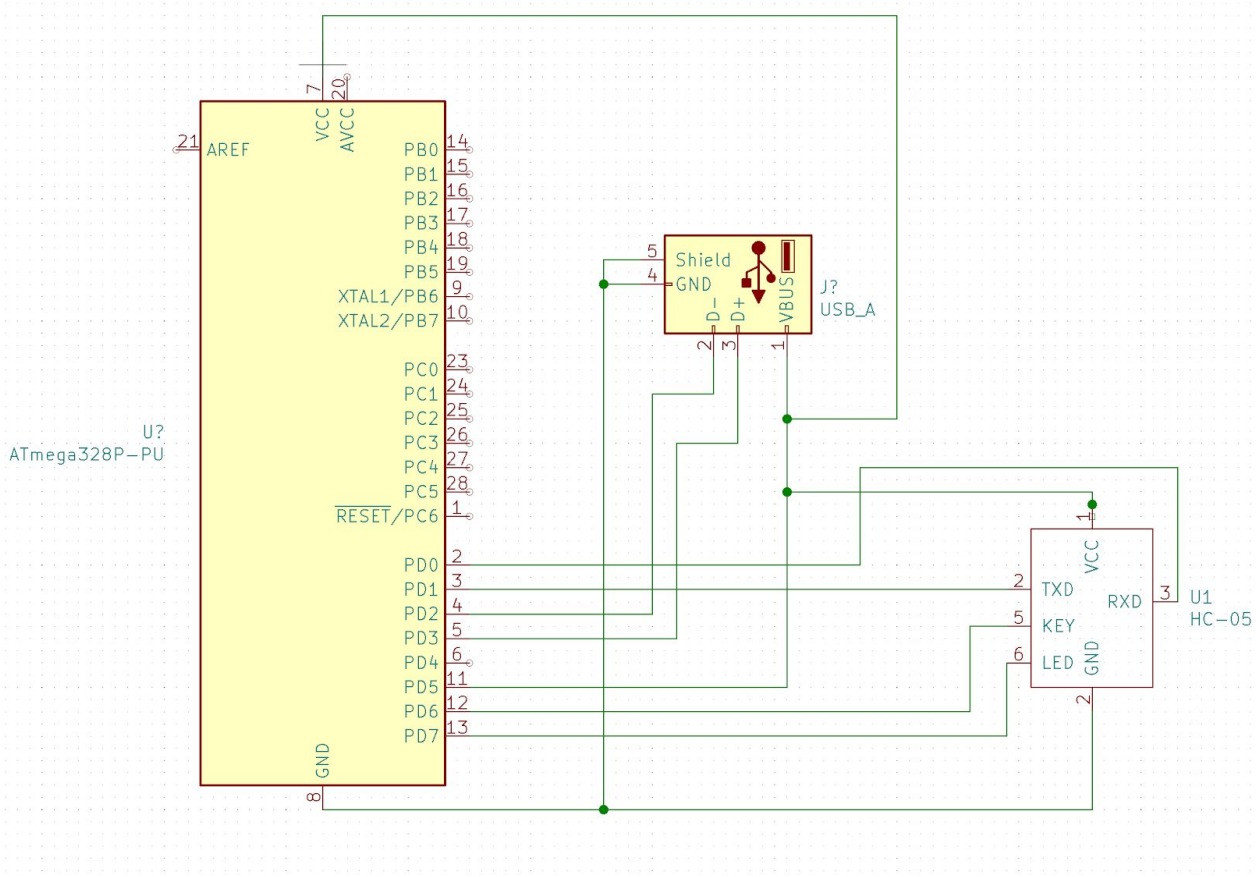

Figure 2. PCB Schematic

Block Diagram

Our project consists of 4 major blocks , the phone application, the computer application, the communication module, and the microcontroller. The primary circuit board contains the communication module and the microcontroller. A 5V power will be supplied to the PCB through the USB vbus via its computer connection. Using a bluetooth connection with the phone the communication module is responsible for receiving data inputs from the phone. Data from the phone will be sent based on a modular setup on the phone using various tile configurations. Received data from the communication module will then be made into appropriate data signals by the microcontroller which will process the computer application as necessary. This process is illustrated in the block diagram below in figure 3 as well as in the physical design in figure 1.

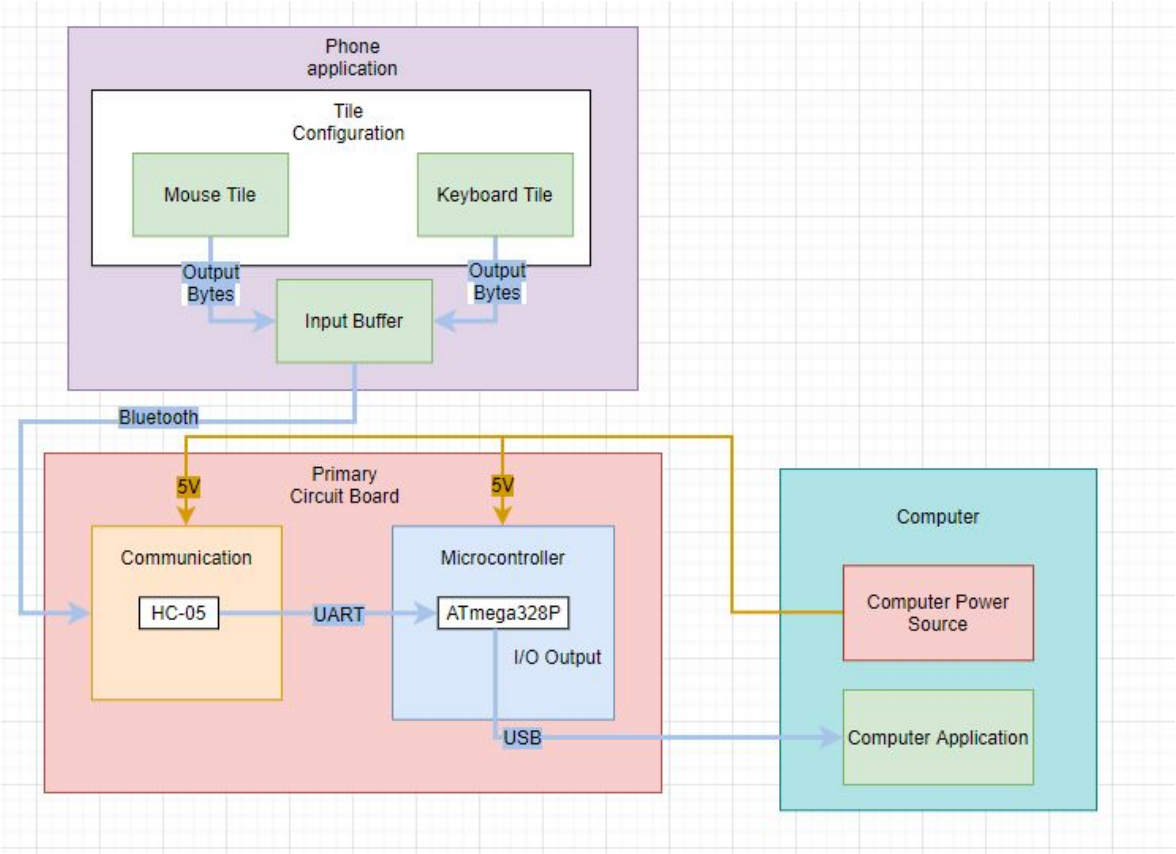

Figure 3. Block Diagram

Second Project Implementation

PCB

The printed circuit board has three main components: the USB connector, the Atmega328 chip, and the HC-05 chip. USB connector is responsible for powering the entire PCB and moves data between the microcontroller and the computer. The microcontroller consists of the Atmega328 chip which is responsible for processing data received from the bluetooth module, so that the computer can easily use the data. Atmega328 chip will be prepared with the necessary software to function ahead of time, so that most of the computation required for communicating with the computer can be accomplished on this piece of dedicated hardware. This would hopefully allow the system to finish computations within the desired .5 seconds range. Code for the Atmega328 chip would take in data that is identified as either keyboard data or mouse data and then process the ASCII sent from the phone into data packets that could then be used by the native drivers of a computer to manipulate the cursor or send keyboard data. The code would also consist of a basic script that will check an environment variable on the users computer which will consist of the keycode for bluetooth pairing. This will consist of a program that we would have tested on an arduino ahead of time and then translated into c in order for the code to work the chip as a stand alone. In order to provide some security, the pair code used to connect to the computer via bluetooth will be set from a prompt in the computer frontend application. The circuit used for the PCB can be viewed in figured 2. The HC-05 chip will be responsible for receiving data from the phone application. 50mW of power should roughly be supplied to the HC-05 chip for the bluetooth range to properly encompass a radius of 15 meters[3]. This will ensure the user can operate their computer from a range as expressed in the high level requirements.

Windows Application

The computer application would be built for windows, since we do not currently have a Macbook computer to test on. The application would be programmed in C and run in the background only displaying a small icon on the toolbar. When the icon is clicked it would simply display a generic windows prompt with text asking if the user wanted to change the keycode on the PCB for pairing the device. After responding to the prompt the system would simply update the environment variable that was created for the bluetooth pairing key. The symbol would be green if a phone was connected to the computer and the symbol would be red if no device is connected. Since the system does not require the computer to do any calculations it should operate with minimal strain on the computer's resources.

Phone Application

The phone application would have been programmed in react native and would consist of three screens: a bluetooth connection screen, a configuration screen, and an operating screen. When the app first boots up the user should immediately encounter a screen asking if the user would like to connect to a computer or configure their settings.

On the configuration screen they will be able to look up and add tiles consisting of keyboard tiles, and mouse tiles which can be used to operate a computer from a keyboard or mouse stand point. There will be 3 kinds of mouse tiles : a left click mouse tile, a right click mouse tile, and a mouse movement tile. A mouse movement tile will allow a user to operate a computer mouse by placing their finger on the tile and moving the finger within the tile. The app will check the differentiation of the positioning of the user's finger in order to move the mouse appropriately. The mouse left click and right click tiles will consist of squares that can be used to left and right click perspectively. Keyboard tiles will also consist of squares with symbols that will represent keys that you could find on a keyboard. There will also be special keyboard tiles that usually require multiple key presses to activate, but can be activated just by clicking the tile. One such tile example would be the bracket symbol.

By default there will be three tile configurations already setup: a mouse, a keyboard, and a numpad tile configuration. This configuration can be viewed in figure 1. From the configuration menu it is possible to adjust which configurations are displayed as well as the size of all the tiles used in each configuration.

On the connection screen the user can view what bluetooth devices are available to connect to and be asked for the bluetooth pairing code. When on the operation screen a user can click multiple keys in order to in essence press the same key or keys on a keyboard, and operate their mouse using the mouse tiles. Whenever a tile is pressed data is sent to an action buffer, first the type of data either mouse or keyboard data is stated. Then if the data is keyboard data the appropriate ascii is stored, but if its mouse data position and click status of the left and right mouse button data is sent.

Pseudocode

This is a pseudocode implementation of what would be running in the backend of the phone application

This method will get called whenever a user clicks a tile on the frontend interface.
Input contains two fields: a list of data and an enum stating if it is keyboard data or mouse data.
Action is a parent class of KeyboardData and MouseData
KeyboardData contains an id of 0 and has

MouseData contains an id of 1 and hase left click, right click, x difference position, and y difference position

```
QueueAction(Event input):
        if(input.type == Keyboard):
                #data[0] for a keyboard contains the characters value
                Action a = new KeyboardData(0,input.data[0])
                buffer.add(a)
        else (input.type == Mouse)
                #input.data[0] is  left click,
                #input.data[1] is right click,
                #input.data[2] is x differnce position
                #input.data[3] is y difference position
                Action a = new MouseData(1,input.data[0],input.data[1],input.data[2],input.data[3])
                buffer.add(a)
```

This method will be called after connecting the phone to the PCB, to send data.
Frequency is how often the system will send packets to the PCB.

```
LoadBalancer(int frequency):
        Int counter = 0
        while(true):
                if(counter == frequency):
                        Counter = 0
                        send(buffer.remove())
                else:
                        counter++
```

This is a pseudocode implementation of what would be running on the Atmega328 chip to interpret incoming packets from the phone application

```
MpacketConvert(Action input):
        #mousePositionX and mousePositionY are local variables
        mousePositionX = mousePositionX +input.mouseX
        mousePositionY = mousePositionY +input.mouseY
        Packet1 = 0+1 +Input.mouse_leftclick + Input.mouse_rightclick + (mousePositionX XOR
0x11000000)>>4+(mousePositionY XOR 0x11000000)>>2
        Packet2 = 0+0+(mousePositionX XOR 0x00111111)
        Packet3 = 0+0+(mousePositionY XOR 0x00111111)
        send(Packet1)
        send(Packet2)
        send(Packet3)
KpacketConver(Action input):
        Packet1 = input.keyvalue
        send(Packet1)
```

Tolerance Analysis

To get a rough idea of how large the key tiles for the phone application should be, we need to take into account the dimensions of the phone (Galaxy S9) itself. The phone's resolution is 1440x2960, with a physical viewport coordinate system of 360x740[2]. This means that each viewport coordinate corresponds to a 4x4 pixel block. For the purposes of our application, we will assume the phone is in landscape orientation. Knowing that on a regular phone keyboard, 10 keys fit within the width of the phone screen, and that 5 rows of keys takes up half the screen, we can get an idea of how large standard phone keyboard tiles are:

$$(\textit{width of viewport}) - 10 * (\textit{width of key tile}) - 11 * (0.2 * \textit{width of key tile}) = 0$$

The ⅛ keytile term accounts for the spaces in between keys that are present in all phone keyboards. Solving for the width of key tile nets us a key tile width of 60.6 in viewport coordinates. Moving onto the height, we get this equation:

$$0.5 * (\textit{height of viewport}) - 5 * (\textit{height of key tile}) - 6 * (0.2 * \textit{height of key tile}) = 0$$

Once again accounting for space between rows of key tiles, solving for the height of the key tile term gives us a value of 29.03 in viewport coordinates. Knowing now that regular phone key tiles are 60.6x29.03 (we shall make this 60x29 to make things easier for us), we have a baseline when designing the size of our keys for the phone application.

Subsystems

In order to test our software/hardware and ensure our system will function properly, we would use the following tests and verifications. The tests and verifications are separated by subsystem.

- Phone application
    - Consists of the modular user interface that the user will enter keystrokes into, and the mouse interface used to manipulate the computer's mouse. It also contains the input buffer that keeps track of which keys would be pressed correspondingly on the computer. See table 1 for explicit requirements and verifications.
    - **Requirement**: The user can have up to 50 key tiles of various sizes, allowing the user to take full advantage of the entire screen surface
        - **Verification**: For the purposes of this project we are assuming the phone used is a Samsung Galaxy S9. It's resolution is 1440x2960 , with a viewport coordinate system of 360x740 [3]. This corresponds to 10 keys

width if the phone is in a vertical orientation. Using a horizontal orientation, this means we can potentially have (2960/1440)*10 = 20.5 or 20 keys per row, easily allowing us to have 2.5 or 3 rows on the phone application screen.

- **Requirement:** The user can have up to 3 different configurations of key tiles for general use,allowing for the user to quickly switch between them
    - **Verification:** Program 3 different configurations that can be swapped through swiping on the phone screen. If the user can enter a key in one configuration, and switch to another configuration and enter a key for all other configurations, this requirement is satisfied.

- Communication Module
    - Specifically a HC-05 bluetooth chip to communicate with the phone software user interface. Will be part of the primary circuit board. It will be connected to an ATMega328p microcontroller. See table 2 for explicit requirements and verifications.
    - **Requirement:** The phone application can repeatedly send a key tile packet (10 bits) without loss of data to the primary circuit board. Loss of data is quantified by having more than 3% of packets having incorrect bits.
        - **Verification:** Write a basic program for the phone to send a packet of a random 10 bits to the computer through the primary circuit board. We can test accuracy by sending 5,000 packets. If 150 packets or more are incorrect, the requirement is not met.
    - **Requirement:** The phone application can send a key tile packet to the primary circuit board via the HC-05 chip in under a tenth of a second
        - **Verification:** To send a key tile packet, we will use two ten bit packets. The first will be identifying that is a key packet, as well as what configuration is being used. The second packet will contain the specific key that the user has inputted. The baud rate of a HC-05 Chip by default is 9600[4], or 960 bytes per second, so 2.5/960 = 2 milliseconds. We are giving a buffer of 50 times this to account for connection issues,etc.
    - **Requirement:** The phone application can send 3 packets to the primary circuit board for mouse movement[5].
        - **Verification:** We will be testing this requirement by sending 4 packets, the first telling the primary circuit board that this is a mouse packet set. We will write a program that sends 3 packets in a group to the primary circuit board. We will check for loss of data and ensure there is none.
    - **Requirement:** The phone application can send packets to the primary circuit board from up to 15 meters away.

        - **Verification:** Have the phone user stand 15 meters away from the primary circuit and attempt to send a key tile or mouse packet via the phone app. If the primary circuit board receives this packet, the requirement is met.
- Microcontroller
    - This subsystem consists of the ATMega 328p microcontroller, capable of interpreting the incoming data into inputs resembling keyboard and mouse inputs. Will be part of the primary circuit board, and is connected to the HC-05 bluetooth module in the communication subsystem. See table 3 for explicit requirements and verifications.
    - **Requirement:** The microcontroller can correctly interpret the 3 mouse packets[5] into a form usable by the computer application.
        - **Verification:** We will send the 3 mouse packets in a form similar to what is used by serial mouses, the most common type of mouse. We will create a test program that is considered a success if it can correctly interpret test mouse packets.
    - **Requirement:** The microcontroller can correctly interpret the key tile packets into a form usable by the computer application.
        - **Verification:** Similarly to the mouse packets, we will check if the test program can correctly identify the pressed key from the key packet. We will test this using a configuration of the maximum number of 50 keys.
- Computer Application
    - An environment application that will run in the background that processes the data from the microcontroller as signals for a keyboard.See table 4 for explicit requirements.
    - **Requirement:** Given proper key tile and mouse inputs, the computer application can output characters or mouse movements to its native computer screen.
        - **Verification:** We will conduct a test specifically testing the movement of the mouse in 8 cardinal directions and observe whether or not movement is mirrored on the computer screen. We will also do the same with key tiles in the phone application to check for desired output.

Second Project Conclusions

Implementation Summary

In order to implement our solution our product would require the creation of a hardware bluetooth fob, a phone application, and a windows application. The PCB would be broken down into the microcontroller and the communication module. The computer application would consist of a small front end interface and a driver in order to process data from the PCB. The phone application would just be a simple application that will mainly be used to operate the computer and setup configurations.

Our product allows a user to maintain constant operation of their computer while being a distance away from their computer. Using a created piece of hardware to serve as an intermediary between the phone and computer, we can offload the computation from the computer, allowing our product to provide a quick response time. The phone itself would have a displayed keyboard and mouse, with its configuration able to be changed to multiple modes to suit the needs of the user.

Noah would have made the PCB, the computer end application, and the mouse configuration tiles. Ashwin would have made the phone application, code for the PCB, and the keyboard configuration.

Unknowns and Uncertainties

At the moment, we are unable to create any of the physical hardware required for the project, which would be the primary circuit board that would connect to the computer and included the HC-05 chip, the Atmega328 chip, and the USB connector. In order to build the primary circuit board we would need access to the lab's soldering equipment. We also currently do not have the time necessary to build any of these components. Another problem is we never received the Arduino we planned on using to test and build programs for the Atmega328 chip, so we can not build any of the translation or packet building programs needed for the programming on the Atmega328 chip. In the design document I specified a schedule which would detail the process for building all of the components. We currently do not know react native which is necessary for building a phone application. Therefore it is impossible for us to build the phone application at this current point in time. One last thing is we currently do not have access to an open space to

test the bluetooth range of our chips. Thus we cannot test the speed, range, or configuration settings of our product so it is currently impossible for us to test any of our high level requirements for any parts of the project.

Ethics and Safety

Since this is a consumer product there is a certain level of safety we must ensure when creating the device. Immediately apparent is ensuring the safety of the primary circuit board connected to the computer. We have to ensure there are no potential hazards for users operating the device.

In the design of this project we must aim to adhere to the 3rd rule of the IEEE code of ethics: “to be honest and realistic in stating claims or estimates based on available data” [6]. In this final report and in the actual fabrication of the project, we should strive to clearly define the capabilities of this system. This includes the physical specifications, programming requirements, and accuracy of its operation. Also concerning design, we should also keep in mind the first rule of the IEEE code of ethics: “to hold paramount the safety, health, and welfare of the public, to strive to comply with ethical design and sustainable development practices, and to disclose promptly factors that might endanger the public or the environment”[6]. Considering that we are working with electrical circuits, we must take proper care when constructing and testing these circuits to ensure that we are not potentially harming ourselves.

Finally, because we are creating a product that interacts with both a user’s phone and a computer, we must ensure the security of the data being transferred from the phone to the computer, not allowing any malicious parties to gain access to this connection and consequently either the phone or computer.

Project Improvements

One improvement to our project would be creating an application that would allow this project to work on a raspberry pi. Currently the system manipulates the user interface developed for a computer's operating system in order to operate the computer. However if we wanted the system to work on a raspberry pi we would need to create our own keyboard and mouse drivers.

We could also create programmed macro tiles in order to further optimize the users workflow. For instance if the user wanted to open up a specific application on their computer, they could create a macro tile in order to open the application. Then the user could open the application without having to use a mouse configuration or keyboard configuration. Another good example would be a volume up and down macro tiles to control the computer's volume. To create a macro tile, there would be an interface within the phone application that, when prompted, would allow

the user to create a new macro, specify what key presses it corresponds to, and name it for their use.
Another improvement would be allowing multiple devices to connect to the computer at once. If a teacher can have all of their students connect to their computer, students could instantly respond to a prompt from the teacher in real time. This would require us to figure out how to handle multiple inputs at the same time and most likely require some version of an input buffer. We would also need to expand our primary circuit board, adding multiple HC-05 Bluetooth modules to allow for multiple inputs/multiple Bluetooth connections to the computer. Having multiple inputs would also necessitate having a load balancing setup, probably through checking the order in/when people sent responses.

Extended Applications

If a teacher/professor is drawing a schematic they can interact with students about how they think it should be drawn and possibly designate the drawing to the student, by having the student connect to their computer. This would allow multiple students to collaboratively work on a single design with real time feedback from their teacher/professor.

Another use case would be with games, specifically in the style of tabletop RPGs. These games require collaboration between multiple members of a group, and while certain applications out there already exist for this exact purpose, our application could potentially negate the slow time of interaction that many of these softwares have, as they use wifi to connect multiple players together.

Working on both of these extended applications would require some of the add ons that we talked about in the previous section regarding project improvements, such as adding functionality for multiple inputs at once. The two projects would diverge from there, since for a schematic drawing application, the 3 panel setup in our original project would be helpful, but for the tabletop rpg application, only a mouse setup and a macros configuration would be required.

Progress made On First Project

In the process of making our first project we had a lot of issues with regards to the creation of the modular light array. The complication resulted in my partner and I doing a lot of tests on chips and LEDs in order to prove that the way the lights would operate was how we expected them to. One such test involved the frequency at which we were updating lights, we had to prove that we were updating the lights fast enough to the point that it looked like the lights were all on at the same time. In order to test this we set up a basic circuit with a single voltage source and two LEDs in parallel each with its own 100ohm resistor. We set the voltage to cycle at various frequencies to see at what point the lights flickering would be noticeable. Our findings suggested

that we would have to set the frequency below 1 hz before the flicker became noticeable. Aside from flicker we also tested to see if the brightness of the light would decrease due to the increase in the number of lights in use. We reused the setup from above and simply held the frequency at 1Mhz and recorded the brightness when there were two LEDs in parallel and after removing one of the LEDs and resistor. Our findings told us that the brightness did not seem to change based on the number of LEDs connected in parallel. Lastly my partner and I had already begun the construction of the modular light array, in the 445 lab there is currently a box which has been constructed and tested to work with our modular light array design. Also in our cubby contains a 4 by 4 light circuit which we constructed to do testing once we received the Atmega328 chip. We however never got to use the Atmega328 chip and the circuit was only used to test the operation of the light array itself.

References

[1]Amazon.com. 2020. *BEBONCOOL RF 2.4Ghz Wireless Presenter Remote Presentation USB Control Powerpoint PPT Clicker*. [online] Available at: <https://www.amazon.com/BEBONCOOL-Wireless-Presenter-Presentation-PowerPoint/dp/B00WQFFZ9I?ref_=s9_apbd_omg_hd_bw_bAqqRX&pf_rd_r=J4N06NZ4CD4C8SHNZ699&pf_rd_p=278954d0-440b-5f00-b724-8e213dd4a804&pf_rd_s=merchandised-search-10&pf_rd_t=BROWSE&pf_rd_i=160358011> [Accessed 4 May 2020].

[2] J. Kielty, "Viewport, resolution, diagonal screen size and DPI for the most popular smartphones," *DeviceAtlas*, 15-Jan-2020. [Online]. Available: https://deviceatlas.com/blog/viewport-resolution-diagonal-screen-size-and-dpi-most-popular-smartphones#samsung. [Accessed: 02-Apr-2020].

[3]A. Peshin, "What Is The Range of Bluetooth And How Can It Be Extended?," *Science ABC*, 26-Nov-2019. [Online]. Available: https://www.scienceabc.com/innovation/what-is-the-range-of-bluetooth-and-how-can-it-be-extended.html. [Accessed: 04-May-2020].

[4] "HC-05 Bluetooth Module Pinout, Specifications, Default Settings, Replacements & Datasheet," *HC-05 Bluetooth Module Pinout, Specifications, Default Settings, Replacements & Datasheet*. Available at: https://components101.com/wireless/hc-05-bluetooth-module. [Accessed: 27-Feb-2020]

[5] T. Engdahl, "PC mouse information," *PC mouse info*. [Online]. Available: https://courses.cs.washington.edu/courses/cse477/00sp/projectwebs/groupb/PS2-mouse/mouse.html. [Accessed: 02-Apr-2020]

[6] "IEEE Code of Ethics," *IEEE*. Available at: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 23-Feb-2020]