PC Buttonpad

Team 12 – Yicong Dong, Stephanie Jaster, and Adit Umakanth
ECE 445 Design Document – Fall 2019
TA: Mengze Sha

1 Introduction

1.1 Objective

Keyboards and mice work well with simple tasks, but are not the quickest way to interact with computers. There are some actions people perform on a regular basis that take many mouse clicks or keyboard button presses, which are not only tedious but also tiring and time consuming, but could be condensed down into one simple press of a button. Such actions include but are not limited to opening up certain pages altogether in a browser, quickly adjusting volume and brightness of the PC, taking screenshots and sharing to social media with one click, etc. In order to increase productivity, decrease the repetitiveness of working on a computer, and enhance the experience and fluidity of working on a computer, we believe it is necessary to engineer more user-friendly hardware designs of the mentioned characteristics.

We therefore propose to make a compact and portable unit with sensors, buttons and LEDs that can be connected and used on the three major operating systems: Windows, macOS, and most distributions of Linux. These buttons will perform whatever repeated actions the user assigns it, and the pad does not need other power supply except the output from the Universal Serial Bus (USB). Sensors and LEDs shall provide more advanced features that further assist the user with work.

1.1.1 Background Research

Macro buttons are not new and exist on some high-end keyboards and specialized keypads on the market [1]. Workflow of multiple actions are also existing features of many operating systems. One of our team members has experience using the stock Automator application of macOS, and it is still only on the software side with no hardware dedicated to make it easily accessible. In addition, the number of functions are limited and mostly only on the stock applications.

Our project is different because it promises to let users customize the button functionality to their wishes instead of relying on forced presets, spend a much lower cost, and still keep their favorite keyboards on the desk while having our solution in parallel with the existing.

1.2 Visual Aid

As seen in Figure 1 and 2, the PC Buttonpad has been created to offer users with several customizable shortcuts through a series of interactive buttons and sensors. The PC Buttonpad is connected to a PC via USB cable and can be placed anywhere around the computer and keyboard. The user has the capability to program the buttons under separate profiles such as music, internet, etc.. Profiles can be changed just by the swipe of a hand over the PC Buttonpad, where ultrasonic sensors will detect the motion and direction of the hand. The buttons themselves are metallic and sensitive to touch, so there is no need to press down on the buttons with force. PC volume adjustment is available through the use of a rotary encoder, on the left side of the PC Buttonpad, along with an array of LEDs on top for PC battery status indication.

Figure 1. Isometric view of the PC Buttonpad with a desk set-up.

Figure 2. Zoomed-in isometric view of the PC Buttonpad desk set-up.

1.3 High-Level Requirements

- Buttonpad and accompanying software driver should work on the three current versions of mainstream PC operating systems: Windows 7+, macOS 10.11+, and recent versions of popular Linux distributions.
- Button functions should be easily customizable without any specific knowledge on coding or hardware. Someone who only uses a computer for word documents/web browsing should have no problem using the driver software.
- The rotary encoder, touch sensors and ultrasonic sensors should work responsively while the user intends to interact with them while staying idle, i.e. not give erroneous inputs, when one doesn't.

2 Design

2.1 Block Diagram

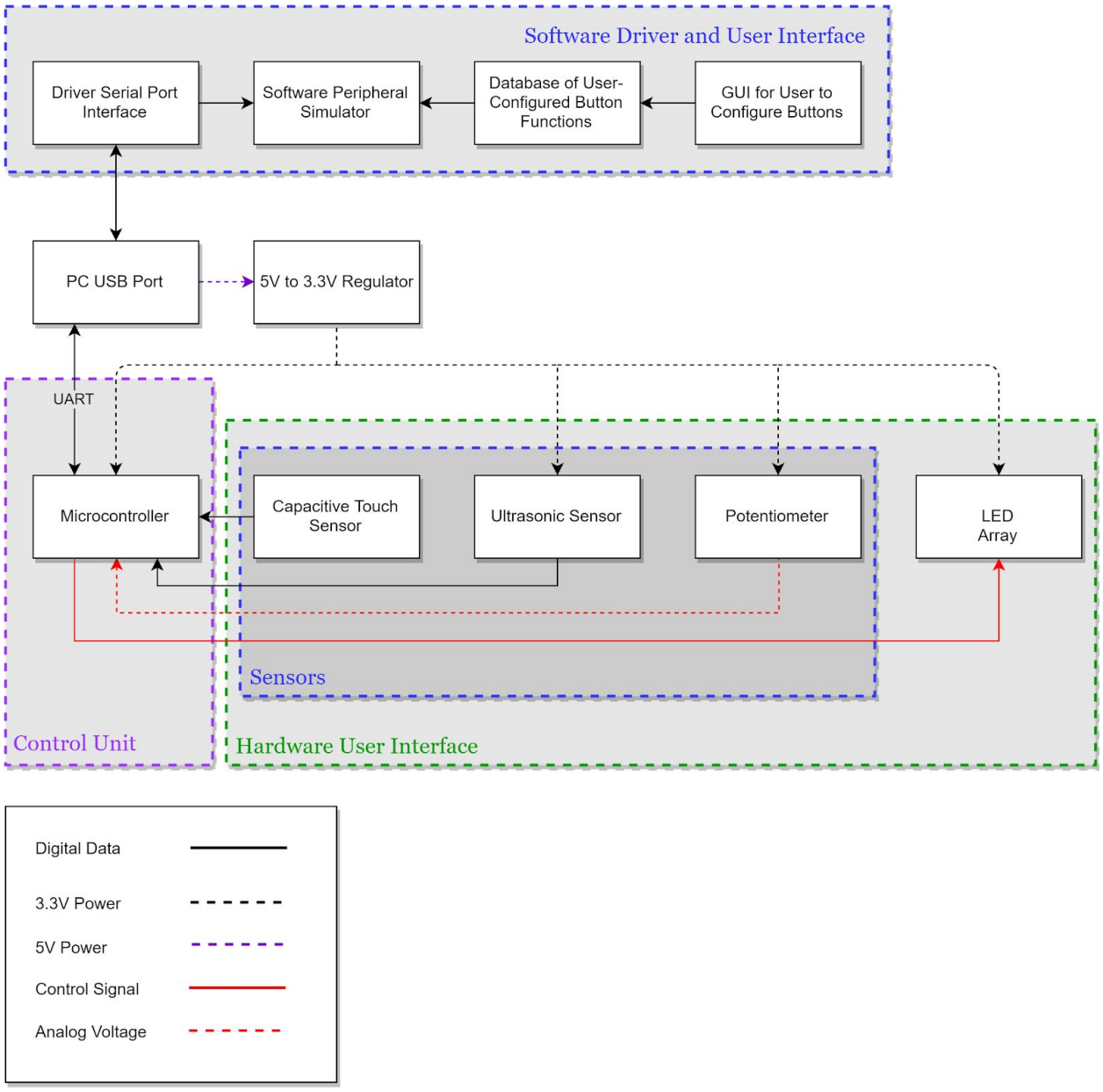
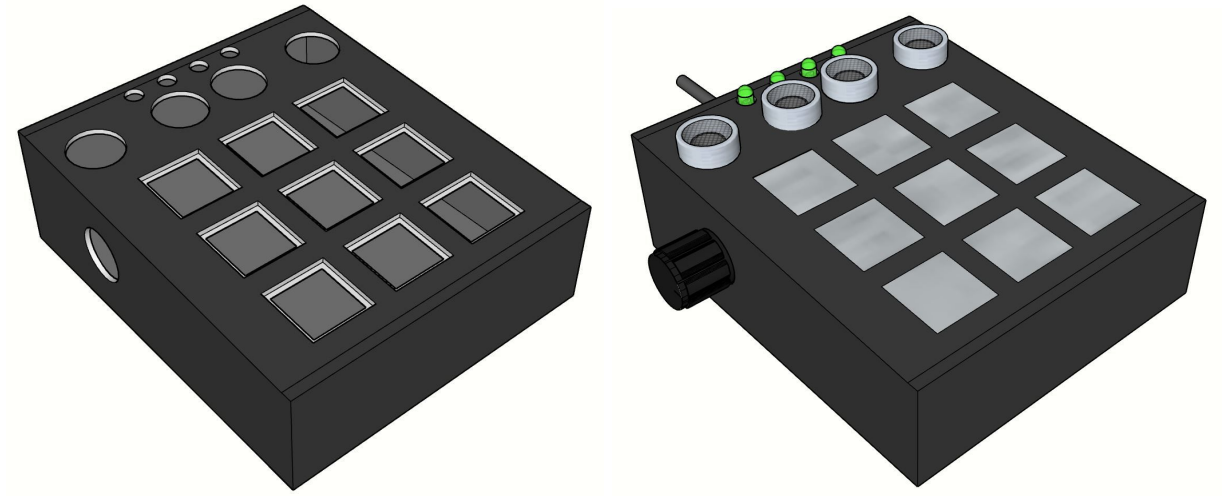


Figure 3. Block Diagram of PC Buttonpad.

As illustrated in Figure 3, the capacitive touch sensor detects button presses and sends a signal to the connected computer through a microcontroller and USB chip. The connected computer has drivers installed which decodes these signals and performs the user-assigned actions. The USB port of the user's computer is expected to power the whole circuit. Additional sensors serve as part of the user interface and provide extended features and functionalities.

2.2 Physical Design

The PC Buttonpad physical design, as shown in Figure 4a, will consist of a 3D printed, plastic shell to encase the PCB, sensors, USB port, and internal wiring. Figure 4b shows the completed design and encasement of all sensors and electronic components, including the metallic buttons. Metallic buttons will be used to enable the capacitive touch sensors placed below the buttons, since touching metal will lead to the greatest change in capacitance. The metal type for this design is to be determined, but we are planning to either use coins or to 3D print using Steel-PLA. This shell will be 3 ⅞" x 4 ½" and roughly 1 ½" thick at most. The dimensions of this design were selected to fit the sensors and maintain a ⅛" wall for durability. Specific dimensions of the design and sensors are displayed in Figure 5, 6 and 7.



a) b)

Figure 4. Isometric views of a) PC Buttonpad outer shell for 3D print and b) finished PC Buttonpad containing metallic buttons, LED array, ultrasonic sensors, rotary encoder, and USB cable.

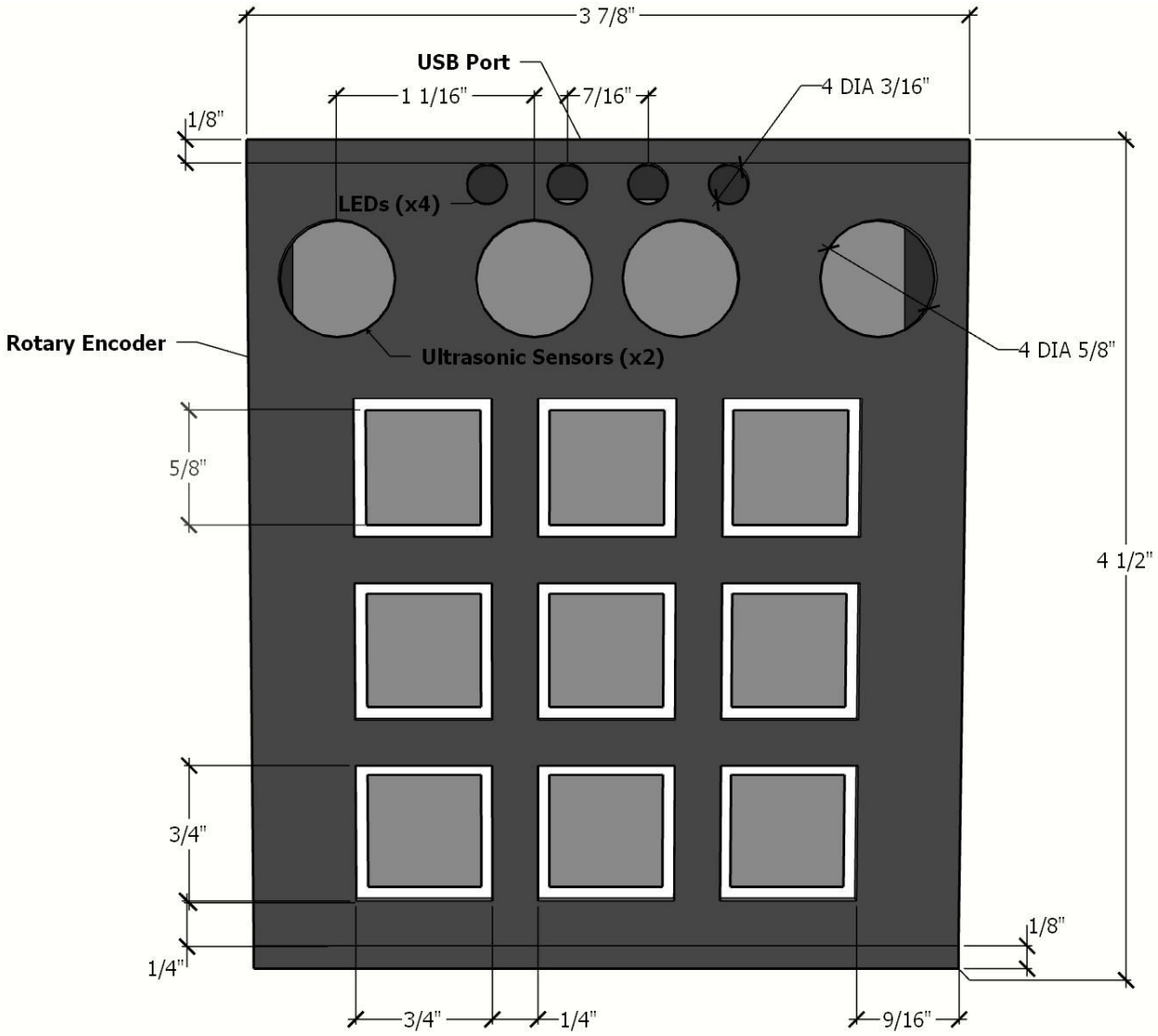


Figure 5. Top view of the PC Buttonpad outer shell with dimensions.

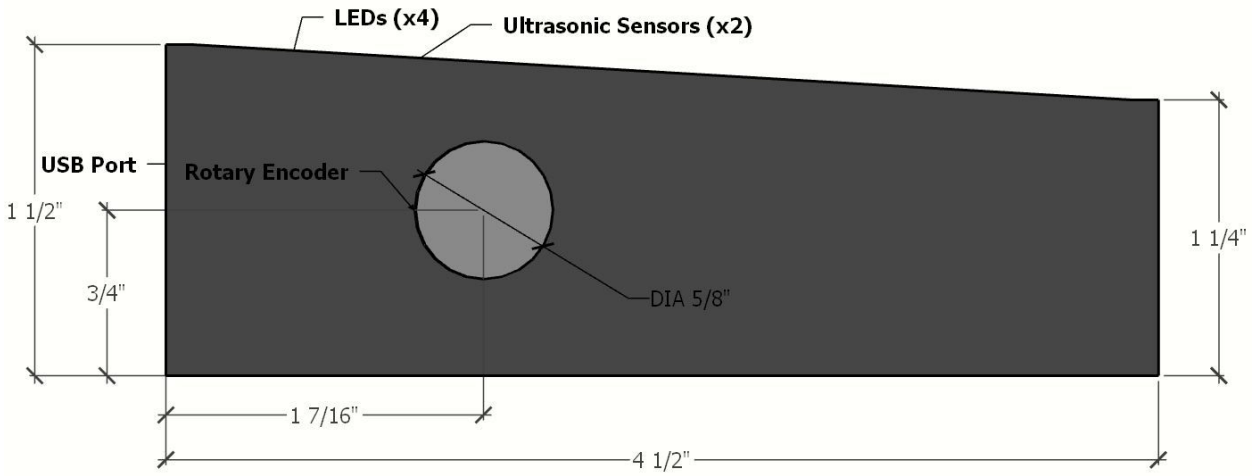


Figure 6. Left view of the PC Buttonpad outer shell with dimensions.

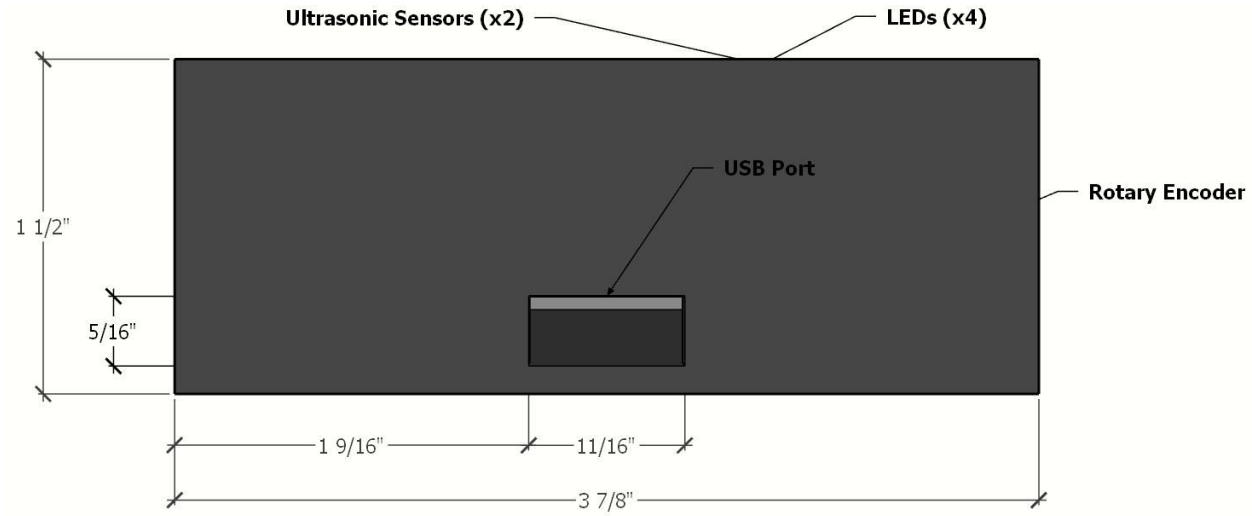


Figure 7. Back view of the PC Buttonpad outer shell with dimensions.

Note that the USB port may in practice be USB-C instead of USB-A if type-C is better at power delivery.

2.3 Subsystems

2.3.1 Software Driver and User Interface

Driver Serial Port Interface: This is the piece of software that allows the microcontroller to communicate with the user's computer. A press of the button or trigger of an ultrasonic sensor is passed on to the microcontroller and sent to the USB Serial Port of the PC, which is listened to by this piece of software. It carries out the relevant actions depending on the signals from the hardware.

Requirements	Verifications
1. The driver should be able to receive 8-bit data values at a rate of 9600 baud.	1A. Set up an Arduino to send a sequence of bytes at a pseudo-random interval (to simulate the human press of a button) and ensure that the driver receives those same sequence of bytes.
2. Receiving a byte should trigger a function depending on what byte is received.	2A. The same Arduino can be used to send the sequence of bytes and some dummy functions can be tested on the driver side to ensure that they are executed.

Software Peripheral Simulator: Instead of having the hardware mimic the signals of a keyboard or mouse, we use the serial port interface to read a custom signal, and then use this Software Peripheral Simulator module to simulate the click of a mouse or typing on a keyboard. This offloads all the limitations of storing information for each button onto the computer instead of our hardware. It also allows further customizations beyond keyboard and mouse interactions such as running shell commands or specific libraries for certain software if required. We will use an open-source library for this module instead of creating this functionality from scratch.

Requirements	Verifications
1. This module should be able to simulate single keypresses and keyboard shortcut combinations.	1A. Attempt to simulate the press of individual letters and ensure they are output. 1B. Attempt to simulate commonly used keyboard shortcuts and ensure that their effect is visible (e.g. close window, open a new tab, etc.).
2. Simulating keypresses should work on the stated operating systems: Windows 10, macOS, and mainstream Linux distributions.	2A. Run the above tests on all three of the operating systems to ensure smooth operation.

Database of User-Configured Button Functions: This is where are all saved keyboard and mouse shortcuts the user sets are stored. Whenever a signal is received through the serial port, this database is checked to find out what actually needs to be executed by the software peripheral simulator. The user does not have direct access to this database, but can freely and easily make changes to it through the GUI for User to Configure Buttons.

Requirements	Verifications
1. The database can hold key combinations and sequences such that it can be read and executed by the software peripheral simulator module.	1A. Store dummy key sequences in the database and attempt to use the software peripheral simulator to read and execute those sequences.
2. Multiple key sequences must be set up to execute sequentially if required, but within reason (< 100 keys).	2A. Test a dummy key sequence of around 100 keys to ensure that this can execute without any glitches or crashes.

GUI for User to Configure Buttons: This module is the core of customizing the buttons to the user's liking. An easy to use GUI will allow the user to choose specific buttons and assign desired keyboard shortcuts or command line executions.

Requirements	Verifications
1. The GUI should allow the user to choose any key combination and sequence of keys, within the reasonable 100-key limit.	1A. Ensure that the GUI allows for easy editing and setting up of the key sequences.
2. A selected key sequence must also be written to the database for further use by the software peripheral simulator.	2A. Set a key sequence using the GUI and then check that the software peripheral simulator can correctly execute that key sequence from the database.

2.3.2 Control System

Since our project is a blend of hardware and software, the main control system will be the Microcontroller on the hardware side and the driver program on the software side. These two modules communicate with each other through the USB Serial Port and a Universal Asynchronous Receiver-Transmitter (UART) to convey data like battery levels of the connected computer and the buttons pressed.

Requirements	Verifications
1. The microcontroller should be able to send a sequence of bytes through the USB serial output. 2. It should be able to receive valid input from the ultrasonic sensor and the capacitive sensor.	1A. Use the driver serial interface to intercept bytes sent by the microcontroller and check for validity. 2A. Check that the microcontroller receives the distance measurement from the ultrasonic sensors encoded as a number. 2B. Ensure that the microcontroller receives different numbers for different buttons pressed on the buttonpad.

2.3.3 Sensors as User Interface

Capacitive Touch Sensor: Since we do not need high resolution for each single button, it is best to build capacitive touch sensors using surface capacitance, where "only one side of the insulator is coated with conductive material. A small voltage is applied to this layer, resulting in a uniform electrostatic field. When a conductor, such as a human finger, touches the uncoated surface, a capacitor is dynamically formed" [2]. We would build nine of these sensors and place them in a 3×3 grid-fashion on the buttonpad. Capacitive touch sensors ensure that there are no mechanical parts in the buttons, making the design simpler and more durable. Right now we have not yet figured out an analog detection mechanism for the touch, but as we found in the capacitive sensing library of Arduino, this would basically be using RC time constant of an external resistor and the capacitance of the PINs [3]. When the finger touches the foil, it is equivalent to a capacitor, and RC constant would increase significantly, and that shall detect the touch.

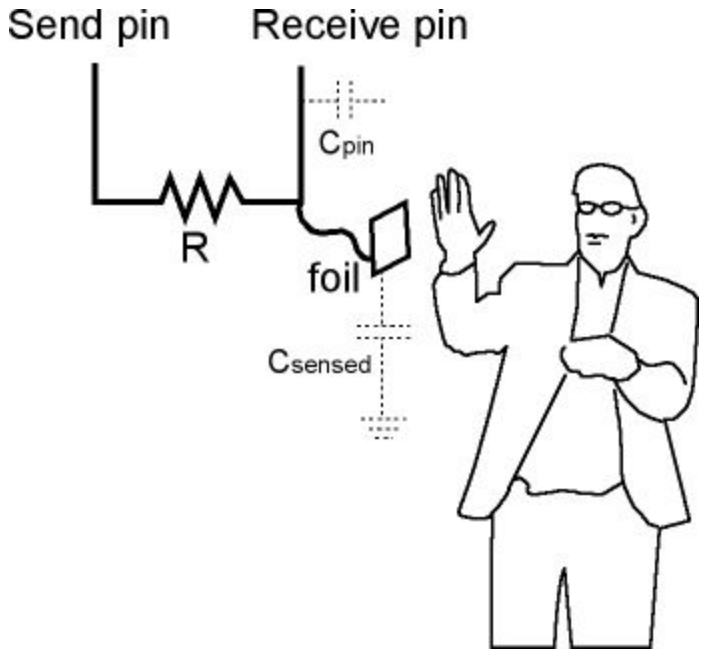


Figure 8. Capacitive Sensing [3]

A sample Arduino code we found online is presented here:

```
#include <CapacitiveSensor.h>
CapacitiveSensor Sensor =
CapacitiveSensor(4, 6);
long val;
int pos;
#define led 13
void setup()
{
Serial.begin(9600);
pinMode(led, OUTPUT);
}
void loop()
{
val = Sensor.capacitiveSensor(30);
Serial.println(val);
if (val >= 1000 && pos == 0)
{
digitalWrite(led, HIGH);
pos = 1;
delay(500);
}
else if (val >= 1000 && pos == 1)
{
digitalWrite(led, LOW);
pos = 0;
delay(500);
}
delay(10);
}
```

Code Block 1. Sample Arduino Code for Capacitive Touch Sensors [4]

What we shall do is somehow tweak this code to fit into a microcontroller; or if the number of digital pins on the microcontroller is not enough, find an analog equivalent schematic that does the same detection of RC constant. For the analog equivalent solution, we would be using a “timer circuit to generate a frequency that is inversely proportional to capacitance and then utilize a microcontroller to count pulses within a given period to calculate the frequency.” See Figure 9 for how to schematic would look like [5].

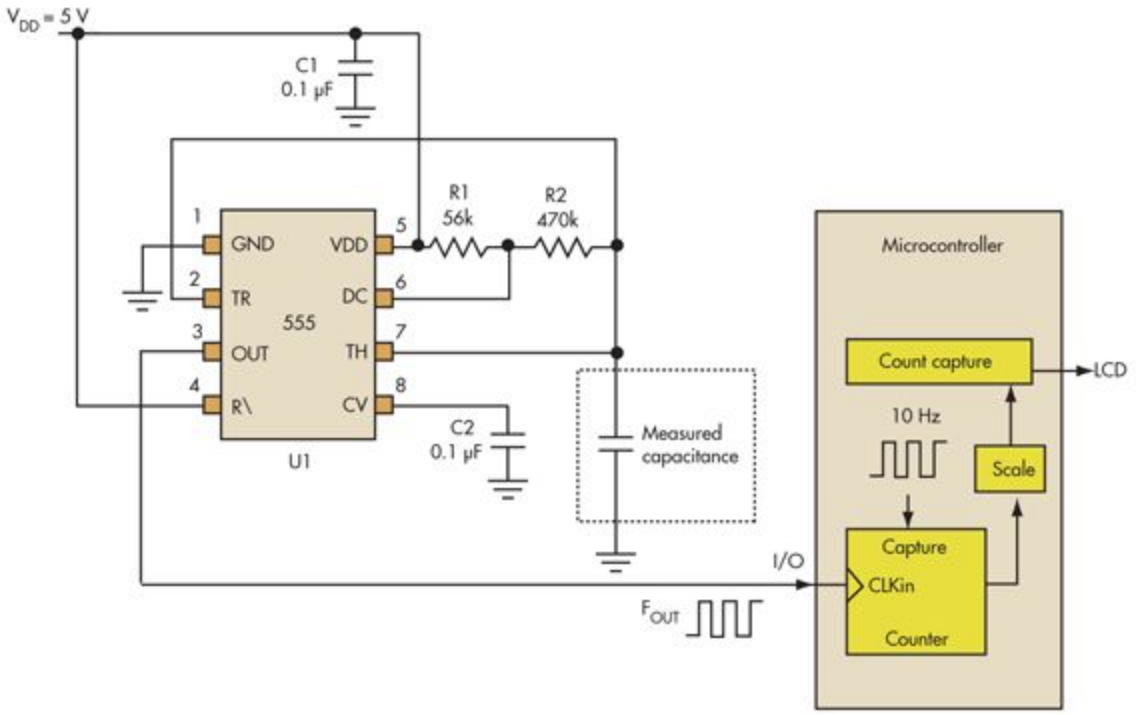


Figure 9. Digitizing the value of a capacitive sensor often involves generating a frequency that is inversely proportional to the capacitance and counting pulses over a fixed period to determine the frequency [5].

And the equation to calculate frequency would be:

$$f = (C \cdot (R_1 + 2R_2) \cdot ln(2))^{-1} \quad \text{(Eq. 1) [5]}$$

Requirements	Verifications
1. Must detect touching situations only, i.e. won't trigger unless the finger is ≤ 0.5 mm away from the button.	1A. Pretest this using Arduino, coins, foils, etc. on a breadboard, and place finger as close to the metal as possible without touching. Observe how the reading changes as we do so. If not desirable, change resistor value.

Ultrasonic Sensor: This sensor really comes down to measuring how far an object is. Using the time it takes for sound to travel to the object and reflect back to get received, it can find the distance from the speed of sound through a simple multiplication. In our case, we would allow a certain range of distance that the user can wave his/her hand passing by two ultrasonic sensors, and this shall allow the controller to figure out if the movement is to the left or to the right. With this movement, the profile of the pad

switches and other functionalities can be activated. These sensors, as shown on the visual aids, will be properly arranged at the farther side of the pad such that using the buttons on the pad would not trigger them mistakenly. Since the ultrasonic sensors output digital signals of the distance information directly, no encoders or any other circuitries are needed.

Requirements	Verifications
1. Distance range for motion detection must be between 2 cm and 1 m.	1A. Test accuracy of data around lower distance threshold at 2 cm and ensure that accuracy is maintained up to 1 m. Data collected between 2 cm to 12 cm will be most important for the uses of this design.
2. Angle of measurement is within 15°.	2A. Test hand placement above sensor at several angles, including 15°, and compare the data. If 15° is too tight of an angle for motion detection, the physical design will be modified to slightly angle the ultrasonic sensor more towards the user.
3. Two adjacent ultrasonic sensors must individually collect data to perform the profile swipe function.	3A. Take and record data from a single ultrasonic sensor. Test two ultrasonic sensors, side by side, to see if their signals interfere or cause any discrepancy in the data compared to the data of a single ultrasonic sensor.
4. Measurement cycle period should exceed 60 ms.	4A. We will set the cycling period to 70 ms in the microcontroller before sending the following 10 μs TTL pulse.

Rotary Switch: This is nothing but a switch that has 12 ON/OFF states, instead of just one ON/OFF. Voltage signal from the rotary switch will take only one of the 12 pins of outputs. By encoding 12 decimal numbers to their corresponding 4-digit binary numbers, we can tell the controller that the switch is being rotated clockwise or counterclockwise. Refer to Figure 10 for the encoding.

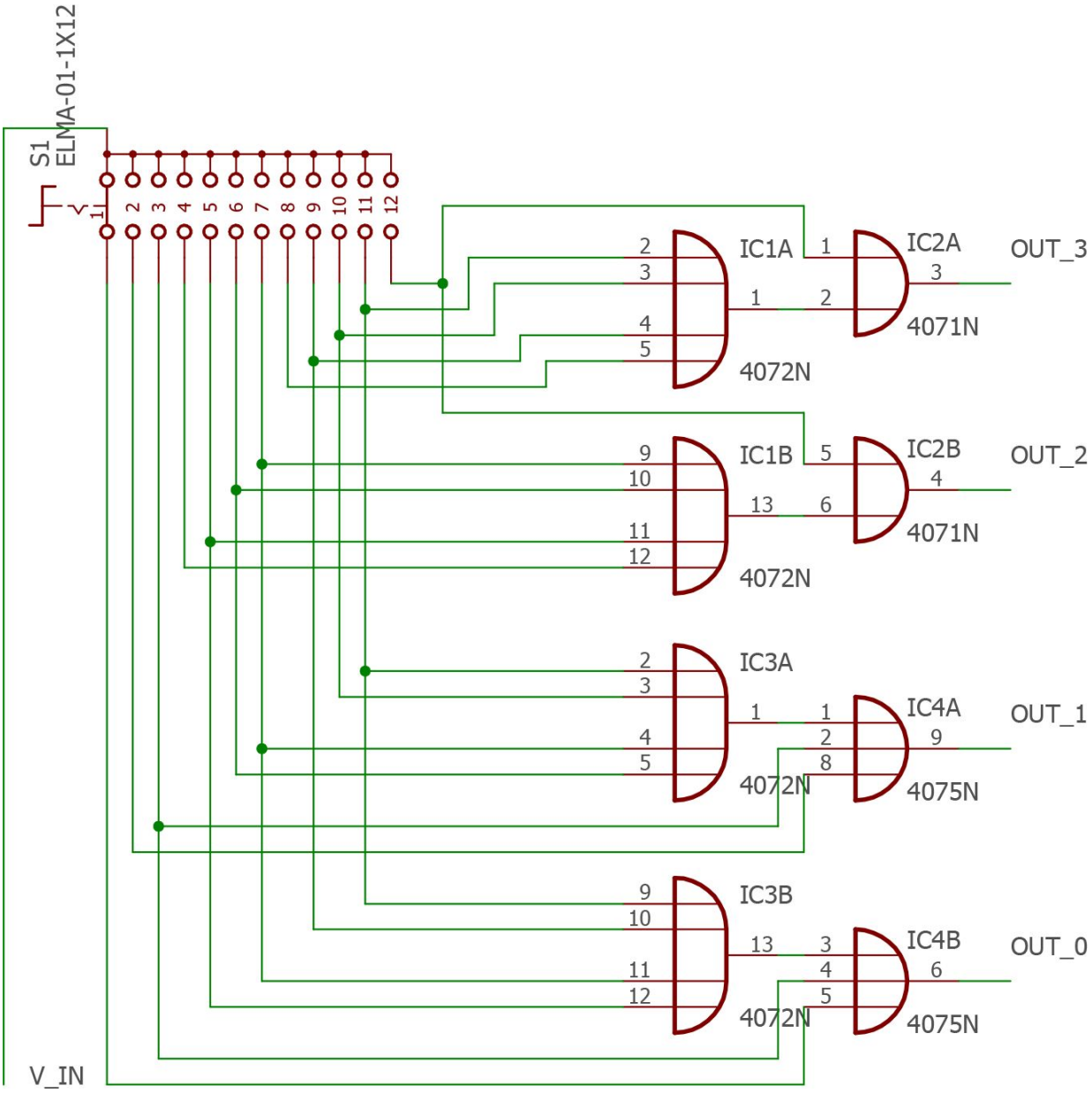


Figure 10. Rotary switch with binary encoder schematics.

Note that here we used ELMA-01-1X12 for the schematic only, and this is subject to change. We are likely going to use what is listed in the parts of the cost analysis. Same for the OR gates. Four input OR gates require more voltage than the 3.3 V supply in a lot of cases, so we would use different ones.

Requirements	Verifications
1. Must output at least 95% of the input voltage, i.e. if 3.3 V goes in, about 3.2 V should come out. 2. Must be able to register the correct binary number of decimal 1~12.	1A. Use a multimeter to measure the voltage coming out of all 12 positions. 2A. Use an Arduino to read the binary outputs and print to the screen so as to check if the encoding is correct.

2.3.4 Additional Hardware User Interface

In addition to sensors mentioned above as part of the user interface, we could also include an LED array to indicate the battery level of the PC. This can be achieved with the design of a 2-to-4 binary decoder that receives two digit binary signals from the PC, and lights up the corresponding LEDs in the array. As for the number of LEDs, we may use four or five LEDs of the same color, and linearly indicate the battery percentage value to the closest proportional ceiling. For example, for four LEDs, 60% battery level would have a digital signal of "10" coming from the controller, and have three LEDs on and one off as it is in between 50% and 75%. A schematic is illustrated in the figure below with input signal being "11", where voltage supply PIN1 and PIN2 are really just the signals from the controller, i.e. not independent power supply. We put voltage sources here for simulation purposes only.

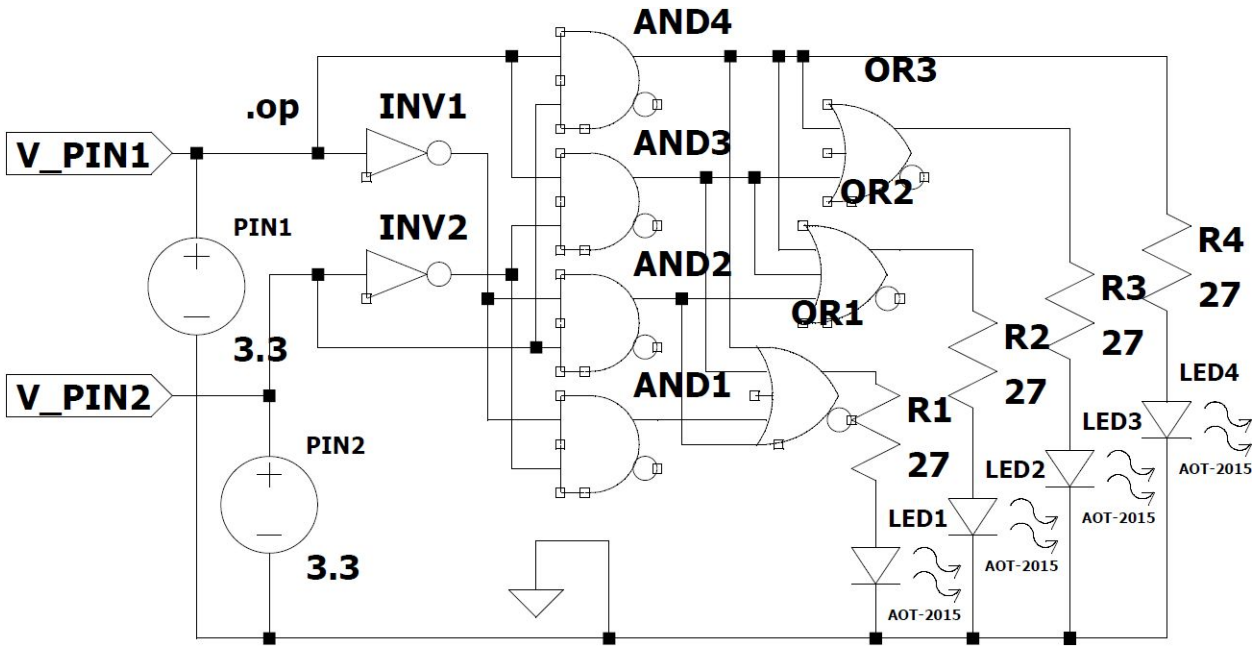


Figure 11. LED array schematics.

The model of LEDs simulated is AOT-2015, which has a forward current maximum of 20 mA and breakdown voltage of 5V thereby being quite representative of indicator LEDs. We would tend to use

different ones instead in the actual circuit as seen in parts table of cost analysis, but this is subject to change due to actual availability.

Requirements	Verifications
1. Illumination of at least 10 mcd at the designed angle of 50° for our LEDs chosen, which translates to about half of the 19 mcd specification.	1A. Make sure that the current through the LEDs is at least more than half of the current limit, i.e. ≥ 10 mA for each LED is expected such that the brightness will be somewhere more than 10 mcd. This can be measured using a multimeter when a 3.3 V power supply goes to PIN1 and PIN2 from the schematics above, and all four LEDs are on at the same time, drawing the largest amount of current.
2. Does not burn out due to exceeding the tested power limit, in this case ≤ 2.6 V, 20 mA is the limit we would be aiming for.	2A. Make sure that the current through the LEDs is strictly less than 20 mA, which again can be measured with a multimeter when 0V goes to PIN1 and PIN2, and only one LED is on.

Simulations gave the same current of 18.1307 mA through the active LED(s) regardless of whether one is on or all are on, which is somewhere pretty close to the 20 mA limit. The real case operating points will need more testing at this moment. As before, parts are subject to changes.

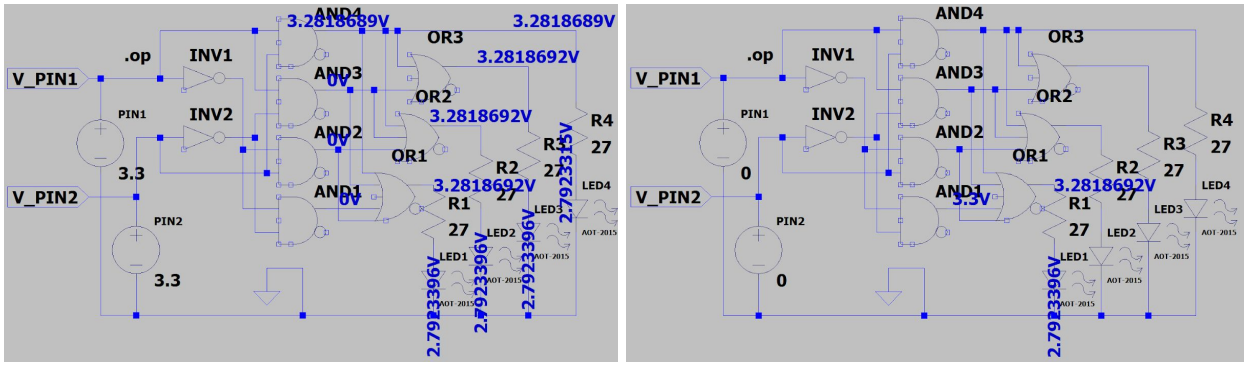


Figure 12. LED array simulation results.

2.3.5 USB Power Supply

We will be using the 5 V power supply from the USB 3.0 ports of a laptop, and use a USB-C breakout to separate power and data. For the power lines, we plan to use a 3.3 V linear voltage regulator such that the voltage works better with the sensors.

Requirements	Verifications
1. The output voltage is no less than 3 V, and the capability of outputting current is no less than 500 mA.	1A. Use a multimeter to measure the output from the regulator when connected with/without load.

2.4 Tolerance Analysis

2.4.1 Capacitive Touch Sensors

The tolerance of the analog capacitive sensing circuit would be of overriding importance if we choose to do it because the touch buttons are the key inputs to our system and the major way users interact with the product. Using what we have from Eq. 1, we could analyze the deviation of the frequency from the ideal situation by accounting for the tolerance of the resistors in the manufacturing process. Using the example values of of 56 kΩ and 470 kΩ as shown in Figure 9, we could assume a typical 5% tolerance for both resistors. We then have the minimum, standard, and maximum frequency calculated below.

Table 1. Tolerance Analysis of Capacitive Sensing Frequency due to Resistance

	Minimum	Standard	Maximum
R1 (kΩ)	53.2	56	58.8
R2 (kΩ)	446.5	470	493.5
f (Hz)	0.000001525/C	0.000001448/C	0.000001380/C

This essentially tells us that there would only be a 10% change in the ultimate frequency for a given capacitance of human fingers whereas our threshold of the detection frequency would always be on the order of multiple times of difference, so there is no issues here.

In addition, we must also acknowledge the possible effects of fluctuating voltage source on this timer circuit. Timer 555 has a minimum supply voltage of 4.5 V, so it would work fine if our USB port voltage output is no less than 90% of the 5 V standard. The 5 V would in many cases be unstable, but in most cases it would certainly be above 4.5 V, so we should be on the safe side of things.

2.4.2 Ultrasonic Sensors

The ultrasonic sensors make up an important hardware feature for our project; they will detect hand motion by the user to swipe between profiles. In order to create the swipe functionality, two ultrasonic sensors are required with each including separate transmitters and receivers. The problem with this is that either of the ultrasonic sensors could potentially pick up the motion signal coming from the opposing ultrasonic sensor. This diagram is shown in Figure 14a, while the diagram of height *d* required for potential signal interference is shown in Figure 14b.

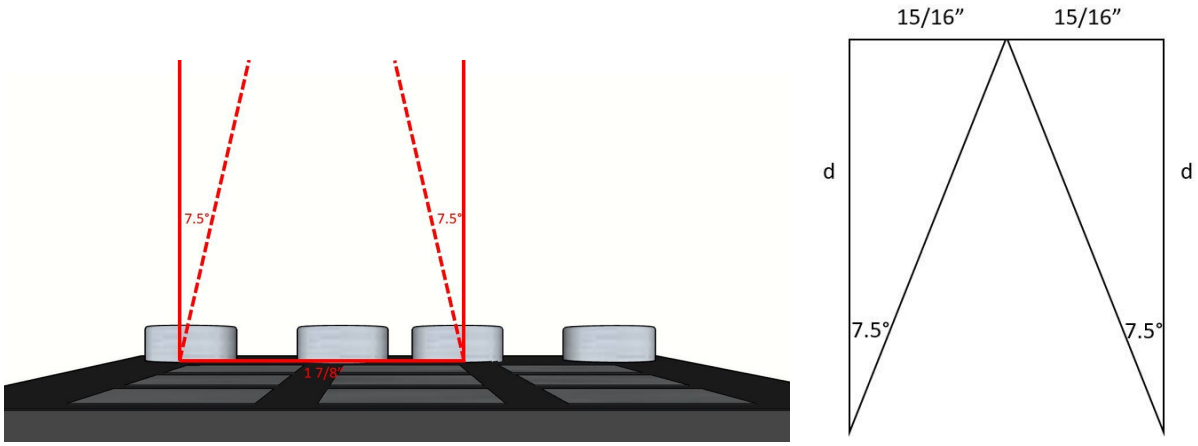


Figure 14. Ultrasonic sensor measurements from a) physical design dimensions and b) distance measurements.

With a measuring angle of 15° and distance of 1 ⅞" separating the two transmitters of the ultrasonic sensors, we can calculate the height *d* of the signal where interference may start to occur. The height *d* can be solved using Eq. 2.

$$d = \frac{15/16"}{tan(7.5)} = 7.121" \approx 18.087\ cm \qquad \text{(Eq. 2)}$$

Given that the range of an ultrasonic sensor is between 2 cm and 4 m, it is clear that motion signal interference may occur within this range. However, with motion signal interference occurring at approximately 18.087 cm above the sensors, there should be no data distortion when the user waves their hand above the sensors. In this design, it is unlikely that users will wave their hand more than 18 cm away from the PC Buttonpad.

If the user decides to wave their hand more than 18 cm above the ultrasonic sensors, motion signal interference can be prevented by controlling the cycle period duration of each sensor. Data collection is initiated by sending a 10 µs TTL pulse to the trigger input, which then outputs a cycle of eight 40 kHz ultrasonic bursts. An echo output signal is received and measures the time of travel, proportional to the distance traveled. The cycle period contains the trigger input signal, ultrasonic bursts, and echo output signal. The length of the cycle period must be greater than 50 ms, so that there is a delay before the next generated 10 µs TTL pulse.We can set the delays for each sensor so that they measure data at slightly different times.

3 Cost and Schedule

3.1 Cost Analysis

3.1.1 Labor Cost

Undergraduates are usually paid somewhere above 10 US Dollars/hour. Assume a 20 $/hr salary for each of the team members, and approximately 10 hours of work per week for each person. For a total of 16 weeks in the semester, the total labor cost would be 9,600 US Dollars.

3.1.2 Parts Cost for a Single Prototype

Table 2. List of Parts in the Project

Description	Manufacturer	Digikey Part #	Unit Price (Prototype)/$	Unit Price (Bulk)/$	Quantity	Proto Sum/$
LED GREEN DIFFUSED T-1 3/4 T/H	Lite-On Inc.	160-1706-ND	0.36	0.04416	4	1.44
SWITCH **ROTARY** 12POS 2.5A 125V	C&K	CKN10200-ND	5.97	3.85429	1	5.97
ULTRASONIC SENSOR DISTANC US-100	Adafruit Industries LLC	1528-2789-ND	6.95	N/A	2	13.9
USB-C **BREAKOUT**	SparkFun Electronics	1568-1958-ND	4.5	N/A	1	4.5
IC GATE **OR** 1CH 2-INP SOT23-5	Texas Instruments	296-1093-1-ND	0.33	0.07107	24	7.92
IC GATE **AND** 1CH 2-INP SC70-5	Texas Instruments	296-8743-1-ND	0.33	0.07107	4	1.32
IC **INVERTER** 1CH 1-INP SC70-5	Texas Instruments	296-1090-1-ND	0.33	0.07107	2	0.66
IC OSC SGL **TIMER** 2.1MHZ 8-SOIC	Texas Instruments	296-1336-1-ND	0.75	0.31971	9	6.75
IC **MCU** 8BIT 32KB FLASH 28DIP	Microchip Technology	ATMEGA328P-PU-ND	2.14	1.78	1	2.14
IC REG **LINEAR 3.3V** 2A 20HTSSOP	Texas Instruments	296-18155-1-ND	10.81	6.13071	1	10.81
Total						55.41

Note that PCB costs and 3D printing are accounted for separately, and we plan to spend budgets on building two prototypes in case one gets damaged accidentally. Parts are still subject to changes after the completion of this document. Resistors and capacitors are not included.

3.1.3 Total Cost for Two Prototypes

As per the table above, two prototypes would cost $110.82, or $9,710.82 with labor taken into consideration. But since most parts can be reused, the testing budget will definitely not exceed that number. For parts in bulk, each button pad could be built for $40.15, a fair reduction from the $55.41 cost per prototype.

3.2 Schedule

Table 3. Schedule with Work Distribution

Week	Adit	Stephanie	Yicong
08/26/19			
09/02/19			
09/09/19			
09/16/19			
09/23/19			
09/30/19	Work on the design document		
10/07/19	Working on GUI and assisting with hardware prototyping	Hardware prototyping; figure out ultrasonic sensor spacing issues	
10/14/19	Refining the GUI and assisting with Eagle CAD	Complete Eagle CAD design for early PCB order	
10/21/19	Choosing and setting up database depending on requirements and constraints of different DB engines	Continue testing hardware components, finalize CAD files for 3D printing, submit button requirements to Machine Shop	Continue testing hardware components and order parts
10/28/19	IPR	3D print physical design and modifications; IPR	Test PCB with components; IPR
11/04/19	Link GUI to database and start creating functionality of software peripherals	Complete third version of Eagle CAD design	
11/11/19	Prepare for mock demo		
11/18/19	Write USB serial code for connection between USB port and microcontroller	Solder components to PCB and test	
11/25/19 (Break)			
12/02/19	Final stages of software and hardware integration testing		
12/09/19	Prepare for final presentation and complete final report		

4 Ethics and Safety

Abiding to IEEE ethics, we are responsible for implementing and maintaining professional and ethical standards by which we follow throughout our design process and afterwards. As a team, we will ensure that all data and communications are truthful, following #3 of the IEEE Code of Ethics [6]. This includes communications between each other, professors, TAs, classmates, and future users. To create a safe working environment for our team and future users, we are aware of our individual technical skillset and will not take on technical tasks that we are not qualified to complete. Assistance will be required in these cases to avoid any unsafe practices that would violate #6 of the IEEE Code of Ethics [6].

Our team values our compliance with the IEEE Code of Ethics through honesty, safety, and our technical work output. We will uphold #7 of the IEEE Code of Ethics to stay true to each other and our individual accountability and contributions within the project [6]. We are each responsible for our individual work and will face proper consequences for discrediting another's ideas or causing extreme setback for our team's progression.

According to #1 and #2 of the IEEE Code of Ethics, we agree to be transparent with any party affected by our work whether it regards safety, our design, or conflicts of interest [6]. It is with utmost importance that we are honest and straightforward with all affected parties, especially when it comes to safety as that is our highest priority in this process. Clarity and guidance will result if a party misunderstands the makeup of our design, technical and physical, the purpose of our product or its safety concerns.

We are also in agreement with the ACM Code of Ethics and Professional Conduct. While this code similarly reflects that of the IEEE Code of Ethics, it points out leadership responsibilities that we as a team must display. Our top priority is to ensure the safety and well-being of all users as listed in #3.1 of the ACM Code of Ethics and Professional Conduct [7]. As leaders, we take complete ownership of our product and strive toward creating a safe and educational environment.

Our display of commitment to the ethical and professional standards outlined will continue during and after our project design and implementation.

The project proposed contains safety hazards that all users must be aware of. The user interface includes several sensors which should be kept dry at all times. Moisture and other types of liquid have the potential to leak into the interior of the PC Buttonpad and could cause damage to the circuitry inside. Corrosion, mechanical failure, electrical shortages, or the creation of fire may occur in this instance. If any liquid leak is suspected, immediately shut off the PC and unplug the device from the PC. To prevent this from happening, we plan on creating a physical model that will seal any gaps between the interior circuitry and exterior of the device.

The metallic buttons are connected to capacitive touch sensors which can malfunction if pressed with too much pressure or if excess layers of dirt, grease, etc. are accumulated on the surface. Excessive forces

placed upon these buttons can result in damages, disconnections, and loss in overall functionality of this product. To prevent the occurrence of these instances, the buttons are placed on a ledge within the 3D model for additional support. However, this does not completely avoid the damaging of our product through forcible touch - disconnections are still possible. The capacitive touch sensors rely on the detection of capacitance change measured across the metallic buttons. If these buttons accumulated thin layers of dirt on their surfaces, the capacitive difference would decrease and eventually make it hard to tell the difference between a touch and no touch. Standard hygiene measures are recommended for the user, such as washing hands, to maximize the button efficiency. In the case that buttons do get damaged, do not disassemble the device, as one incorrect wire placement could lead to overheating elements.

Overheating can also occur if 3.3 V of power is not provided to the design components. This can also happen if more than 5 V of power is provided through the USB port to the voltage regulator. Each module of our device is designed around a 3.3 V power input from the USB, so any more or less voltage could affect the current flowing through the system. Since the device relies on PC power, there is no worry that the voltage through the USB port will exceed 5 V. It is important for the user to plug the device into the correct USB port at 5V when connecting it to the PC monitor. A voltage conversion from 5 V to 3.3 V will be made to ensure that there is no fluctuation in the voltage.

Our team has successfully completed the Lab Safety Training required for access to any lab. Following campus policy #RB-13, Campus Environmental Health and Safety, we take full responsibility for maintaining a healthy and safe environment for ourselves and the rest of the University of Illinois at Urbana-Champaign community [8].

References

[1] "Keyboard with Macro Keys," *Amazon*, 2019. [Online]. Available: https://www.amazon.com/slp/keyboard-with-macro-keys/gnyyx93vhu2484f. [Accessed Sep. 20, 2019]

[2] "Capacitive sensing," *Wikipedia*, Aug. 16, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Capacitive_sensing. [Accessed Sep. 16, 2019]

[3] P. Badger. "Capacitive Sensing Library," *Arduino*. [Online]. Available: https://playground.arduino.cc/Main/CapacitiveSensor/. [Accessed Oct. 2, 2019]

[4] "How to Make a Capacitive Touch Sensor Switch out of Anything Metal Using an Arduino," *Z-HUT*. [Online]. Available: http://thezhut.com/?page_id=1081. [Accessed Oct. 2, 2019]

[5] "Use Analog Techniques To Measure Capacitance In Capacitive Sensors," *ElectronicDesign*. [Online]. Available: https://www.electronicdesign.com/analog/use-analog-techniques-measure-capacitance-capacitive-sensors. [Accessed Oct. 2, 2019]

[6] "7.8 IEEE Code of Ethics," *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed Sep. 15, 2019]

[7] "ACM Code of Ethics and Professional Conduct," *Code of Ethics*. [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed Sep. 15, 2019]

[8] "Campus Environmental Health and Safety," *Campus Administrative Manual.* [Online]. Available: https://cam.illinois.edu/policies/rp-13/. [Accessed Sep. 19, 2019]