Dynamic Keyboard

ECE 445 Design Review

Team 20:

Nigel Haran
Jeevitesh Juneja

TA: Xinrui Zhu

Contents

1. Introduction

1.1 Objective

Programmable keyboards today suffer from the limitation of being programmed through a software and having a lack of communication with the user regarding its functionality. This limitation forces individuals to only be able to utilize their programmable keys through the computer that programmed their keyboard. It also limits the user to using the one keyboard with programmable keys. This can be an issue based on the fact that some keyboards are programmed to provide efficiency, but cannot provide that efficiency with every computer it interacts with.

We hope to resolve this issue by providing a programmable keyboard that is programmed through the hardware of the keyboard itself, instead of the standard software program. What we mean by this is most programmable keyboards use a computer software program to program the keys to a program. Our product will be able to read the key interrupts and record them to a programmable key. This allows one to have the keyboard layout that they desire and have a single key represent a combination of keys. This is done by having an add-on product that reads the keys being pressed through the USB output of the keyboard, recording it, and programming it to a key on the keyboard while providing feedback to the user through an LCD display. By having the keyboard use a hardware based programmable format, the macros that are programmed can be applied to any computer. This is useful for many computer programs and jobs that require the use of multiple key inputs that can become tedious over time. The design we hope to apply is an add on that will record the combination of keys pressed and apply that to a programmable key. However, this would potentially cause a lack of feedback for the user to know whether the programmable key is reading all the functions. We will resolve this by providing an LCD based attachment that will read the function that is being processed and provide a display of the functions being executed at the time to show that the programmable keys are performing what must be done.

1.2 Background

We feel this issue is important because the common software programmable keyboard suffer the limitation of what computers and keyboards can utilize it. Currently on the market there are programmable keyboards that are only programmable through a software process. Lately, they have been designed to provide cosmetics such as color coding for the programmable keys or extra buttons that provide shortcuts. Examples of these programmable keyboards are the Logitech G810 [4], which provides the programmable keyboard with high end switches, LED backlight, hardware built for gaming, etc. There is also the LogicKeyboard Astra [5], which provides backlighting and custom imaging for the keycaps. These two examples are some of the more popular choices when searching for a programmable keyboard. These examples show that most programmable keyboards differ based on the cosmetics and parts used, while still

relying on software for the programming. Our design is trying to provide a keyboard that is programmable through the hardware of the keyboard itself.

By having a keyboard be programmable through hardware it makes the programming aspect be more accessible and no longer have limitation on which keyboard or computer it can be used on. This also allows the keyboard to be independent of the program that is being used on the computer. This is an issue with keyboards that are programmed to be used under a specific computer and program. An example as to how this will be useful is that there are jobs that revolve around using programs like photoshop where an individual has to press a series of keys over and over again. By having a keyboard that can be programmable to perform the series of keys pressed now held by a single key, they can perform their job more efficiently. By having a keyboard that is programmable through the hardware, this individual can perform their job with efficiency at any computer, instead of relying on the computer they used to program the keyboard. By having the LCD display provide feedback it helps the user know what keys are being registered and make sure that the programmable keys have been programmed correctly and debug if there is any issue.

1.3 High-level requirements

- Provide user interface feedback of the keys being pressed through an LED display
- Write and rewrite programmable keys.
- Performs on any keyboard and computer and keeps programmed macros saved for use between different keyboards and computers

2. Design

2.1 Block Diagram

We will have the USB 2.0 provide power to a voltage regulator that will provide 3.3V to the microcontroller and maintain 5V for the LED and switch. The controller will provide information through an interrupt for the LCD to display. This interrupt will also go to the device and program to a key if the switch is activated.

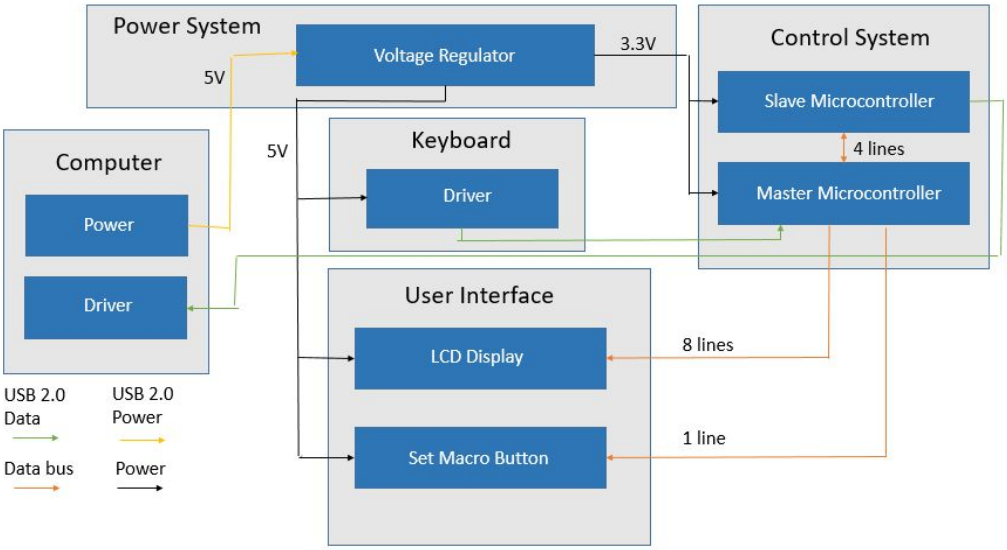

Figure 1: Block Diagram

2.2 Physical Design

The physical design will be created through a 3D printer. Its design is standard, acting as a box that holds all the components in place. There while be two holes to present the switch and LCD display while the PCB will be located within the the empty box casing. There will also be the two USB ports where one will obtain an input from the keyboard and the other will act as an output for the computer.

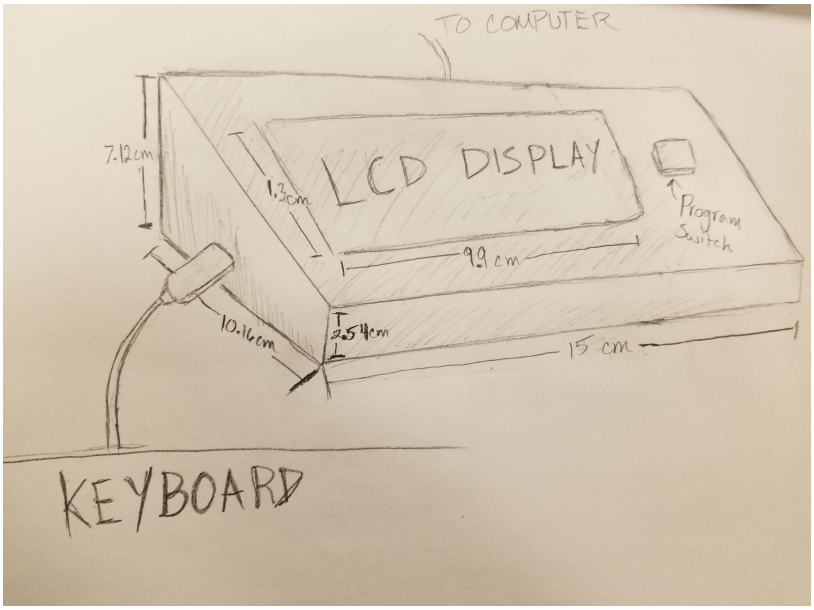

Figure 2: Physical Design and Dimensions of product

2.3 Hardware Design

2.3.1 Power System

A power supply is needed in order to provide the power for all the components in the circuit. We decided to use the USB 2.0 power supply from the keyboard, which can benefit the user by having the power supply built in with the keyboard and not consistently purchase batteries. We will also utilize voltage regulators in order to satisfy the power requirements for the components that require power and provide safety for both the user and the components of the circuit.

2.3.1.1 USB 2.0 Power Supply

A standard USB 2.0 will provide enough power for the entire device. The USB 2.0 is capable of providing a regulated standard of 5VDC to the keyboard [6] in which 3.3V is needed in order to power the microcontroller, driver, and flash memory and 5V are needed in order to provide power for the LCD display user interface.

2.3.1.2 Voltage Regulator

Input: 5V power from the USB 2.0

Output: 3.3V power for the microcontroller, driver, and flash memory. 5V power for the LCD display user interface.

The circuit is made of an LM317 adjustable voltage regulator. The purpose of this voltage regulator is to take in the voltage input and produce a limited voltage output for the rest of the circuit components such as the microcontroller and LCD display. The regulator has a range of input voltage and a maximum output current of 1.5A. The two resistors are for adjusting the output voltage. This is done by having a value for one resistor and adjust the value for the other resistor in order to control the output voltage. The capacitor filters out AC noise, while the other capacitor improves ripple rejection. The schematic and printed circuit of the voltage regulator is provided below. [8]

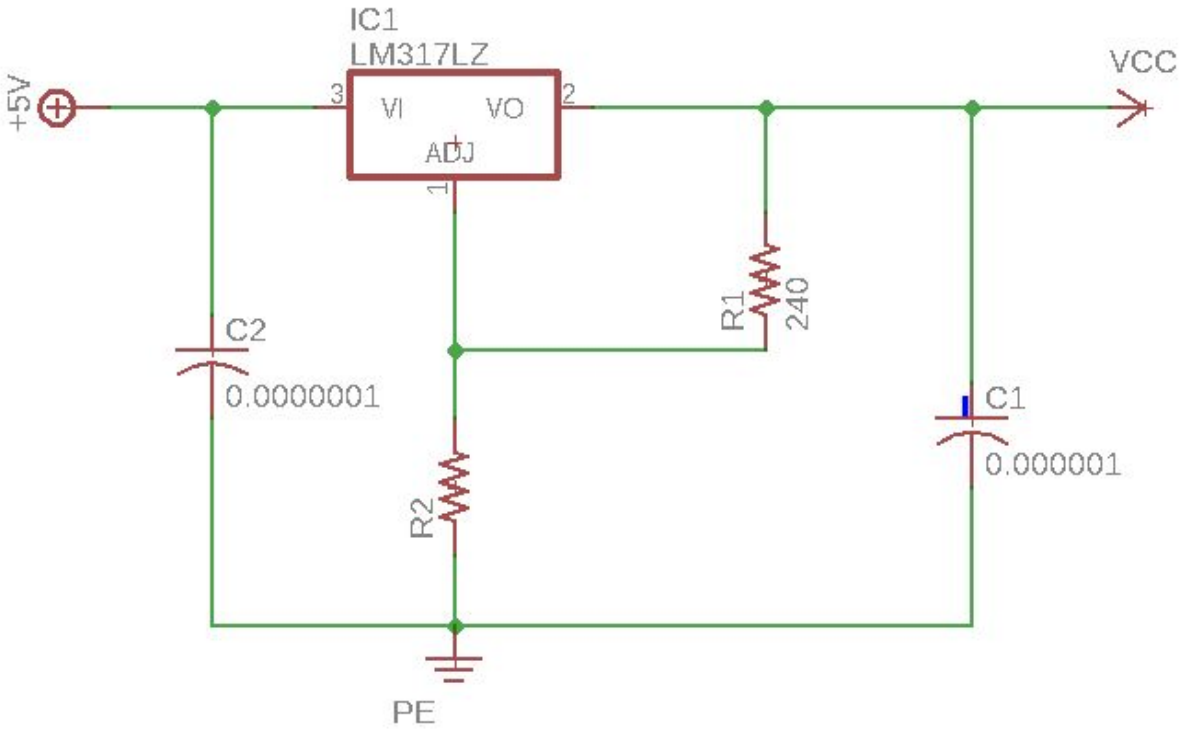

Figure 3: The schematic of the voltage regulator where R2 is 390Ω for a 3.3V regulator and 720Ω for a 5V regulator.

2.3.1.3 Voltage Divider

The voltage divider is two resistors connected in series, this produces a fraction of the voltage supplied by the USB 2.0 which would be too much for the microcontroller and flash memory. We also need it to provide a reference for the LCD applied at pin 3. This part of the LCD needs a voltage of 0.5V in which we create a voltage divider using 9.1 kOhms and 1kOhms.

2.3.2 Microcontroller

Input: Keypad signal, 3.3V power
Output: Keystrokes pattern for programmable keys

We plan to use the PIC18F2550 microchip as they have Full Speed USB 2.0 (12Mbit/s) interface that is required to communicate with the keyboard and computer. As each chip only contain one USB interface we have decided to use 2 PIC18F2550 in a Master/Slave configuration to communicate between our 2 devices. The 256 bytes of programmable storage space on each chip was also another factor in our decision as the we need to be able to store

the macros in non-volatile memory. Figure 4 below shows the the circuit diagram for the 2 chips and includes their connection to other components such as the LCD display, Set MAcro button and USB ports.

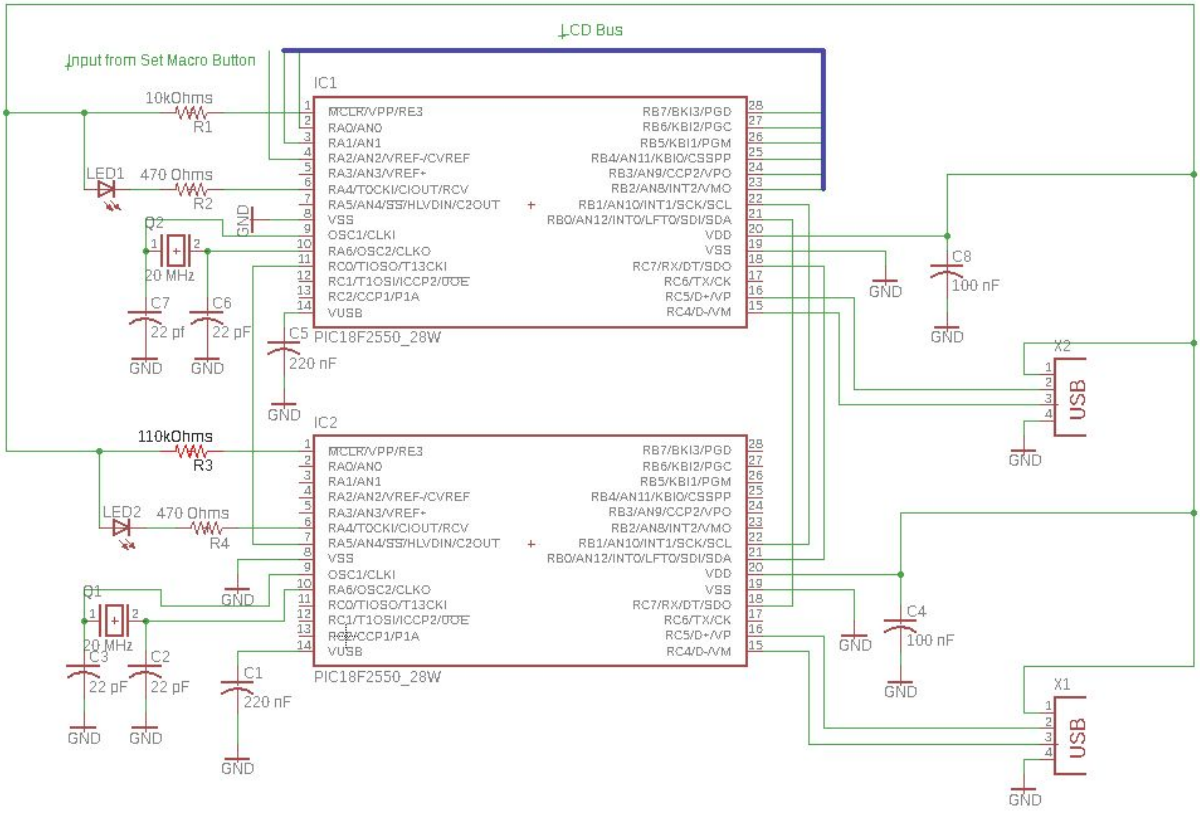

Figure 4: Microcontroller Circuit Schematic

2.3.3 User Interface

2.3.3.1 LCD Display (NHD-0116DZ-FL-YBW)

Input: Key interrupts from the microcontroller and 5V from the power supply
Output: Key interrupt pattern on the display

We wish to use this LCD in order to provide all the characters of the keyboard when we utilize each key. An example would be if we were to program a single key to act as control-alt-delete then the LCD will display this combination when the key is pressed. It will allow the user to know the add on is functioning and that the programmable key is being correctly implemented. We are implementing this through a 1 line 16 character LCD display. [7]

2.3.3.2 Program Switch

Input: Key interrupt from microcontroller and 5V power supply
Output: Key interrupt pattern to provide to the LCD and programmable key

This switch will send the signal to our microcontroller to program a key. The switch is pressed once so the combination of keys pressed will be recorded and then it will be pressed twice to select the key that will keep this macros, and finally pressed again to end the programming.

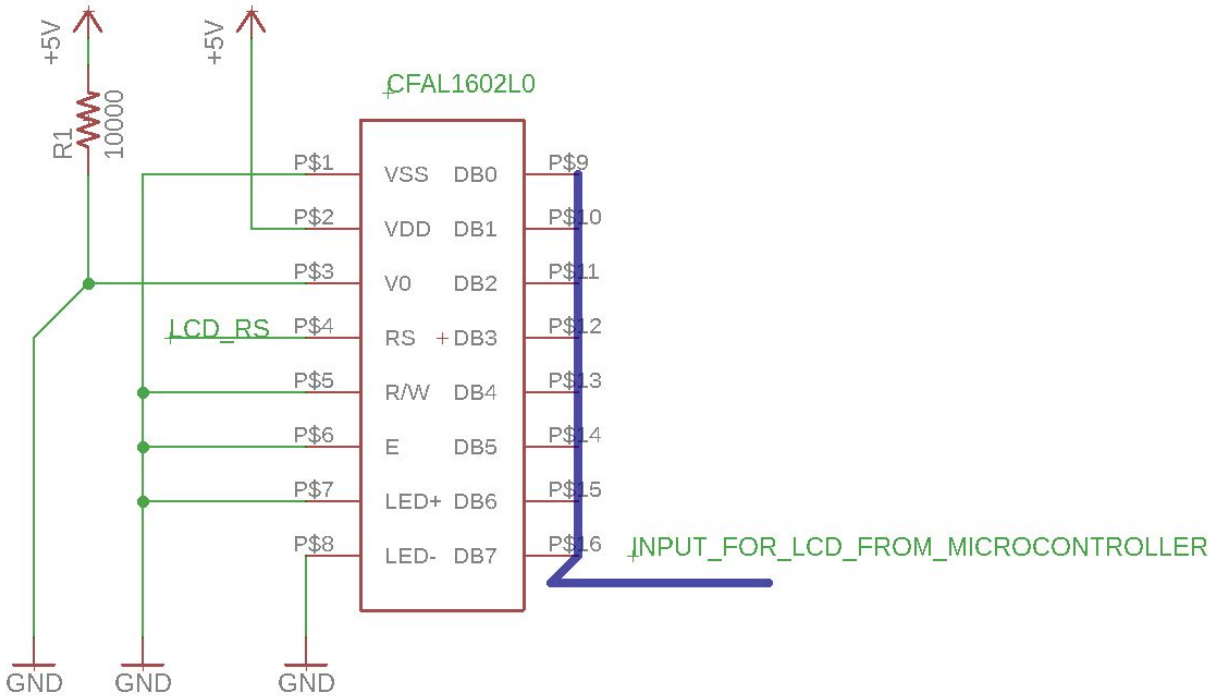

Figure 5: LCD Display Schematic

3 Calculations & Plots

3.1: Voltage Regulator LM317

The output voltage of LM317 is measured as [8]

$$Vout = Vref\,(1 + \tfrac{R2}{R1}) + IadjR2 \qquad \text{Eq.1}$$

In this case Vout is the voltage output once the voltage has gone through the regulator and has been properly divided to provide the desired voltage. Vin is the input voltage from the USB 2.0 power supply. Vref is the voltage that is between the output and adjustments with a value of 1.25V. Iadj is meant to stay minimal in order for the line and load to stay constant. However, because the input current will be small enough, it will not effect the results. We assigned a value of 240 ohms for R1 and use the desired output voltage in order to determine the value for R2. The equation below allows us to figure out what value we want for R2.

In one instance we want an output voltage of 3.3V for the microcontroller and flash memory, this leads to the following value for R2 needed for the voltage regulator.

$$Vout = 1.25(1 + \frac{R2}{240})$$
$$Vout = 1.25 + \frac{1.25R2}{240}$$
$$Vout = 1.25 + \frac{R2}{192}$$
$$Vout - 1.25 = \frac{R2}{192}$$
$$R2 = 192Vout - 240 \quad \text{Eq.2}$$
$$R2 = 192(3.3) - 240$$
$$R2 = 633.6 - 240 = 396.6\Omega$$

We use this same set up in order to find the resistance of the 5V output.

$$R2 = 192(5) - 240$$
$$R2 = 960 - 240 = 720\Omega$$

3.2: Voltage Divider

When using the voltage regulator we need to apply a voltage divider in order to limit the voltage input and produce the necessary voltage output.

$$\frac{Vout}{Vin} = \frac{Rdown}{Rup+Rdown} \quad \text{Eq.3}$$

$$\frac{Rup+Rdown}{Rdown} = \frac{Vin}{Vout}$$

$$\frac{Rup}{Rdown} = \frac{Vin-Vout}{Vout} \quad \text{Eq.4}$$

Vin is the critical voltage level and Vout is half the Vout we want which is 1.65.

$$\frac{Rup}{Rdown} = \frac{7-1.65}{1.65} = 3.24$$

3.3 Plots

Below is a plot of the 3.3 regulator where the regulator will make the output voltage reach 3.3 when it is at a 5V input. If it exceeds a 5V input it will still maintain a 3.3V output.

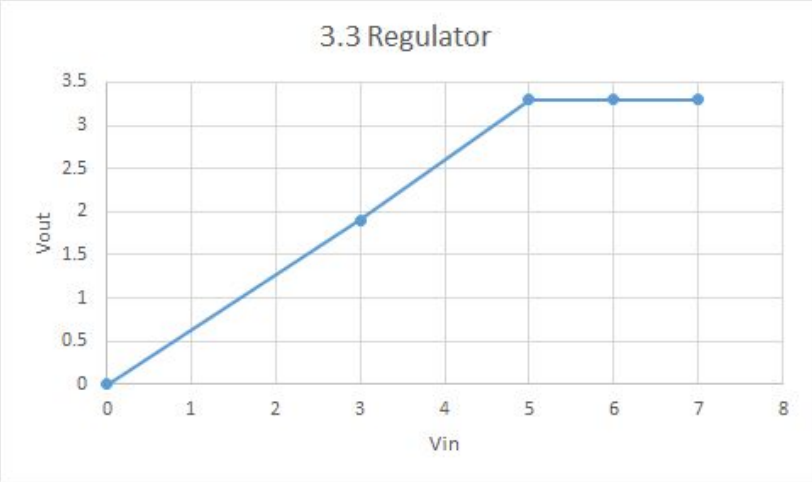

Figure 6: Plot of the 3.3 voltage regulator activity when it surpassed 5V it should level out to 3.3V

Below is a plot of the 5V regulator where the regulator will make the output voltage reach 5V when it is at a 5V input. If it exceeds a 5V input it will still maintain a 5V output.

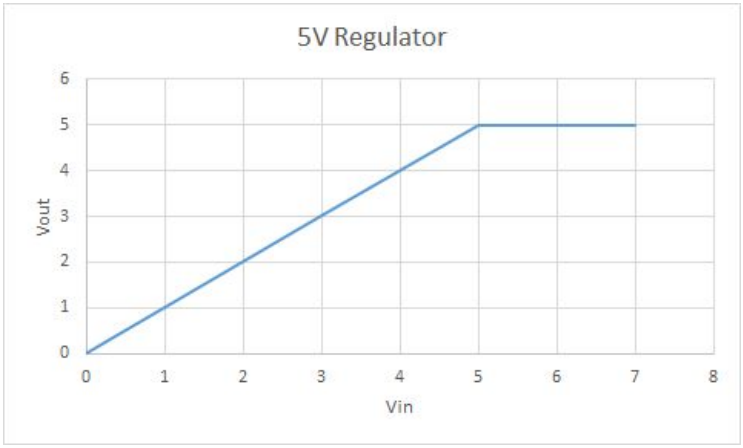

Figure 7: Plot of the 5V voltage regulator showing that when the input surpassed 5V it should maintain 5V output.

3.4 Software Design

The Figure 8 below illustrates the basic control logic the user will interface with when programming their own hardware macros. The detailed software control logic of the the microcontrollers are shown in the figures below.

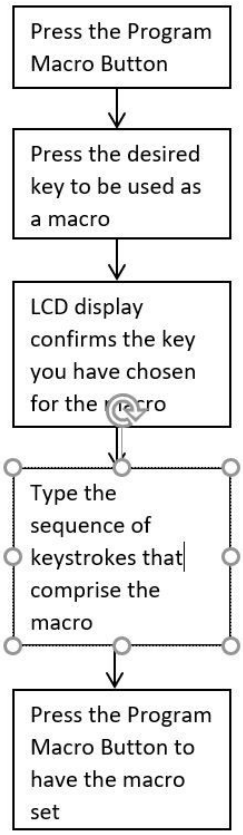

Figure 8: User Control Flow of Setting a Macro

The Master Microcontroller takes on the role of storing and recognizing the appropriate scan-code associated with the the macro keys and providing the data that produces the characters on the LCD. It is also responsible for forwarding any relevant Keyboard output to the Slave Microcontroller. Figure 9 illustrates the control logic of the Master Microprocessor during normal operation, while Figures 10, 11 and 12 describe the control flow when an interrupt is generated from the Keyboard, Slave Microprocessor or Set Macro Button respectively. These interrupts are expected to be asynchronous in initiation, they execute atomically with respect the the control flow during normal operation.

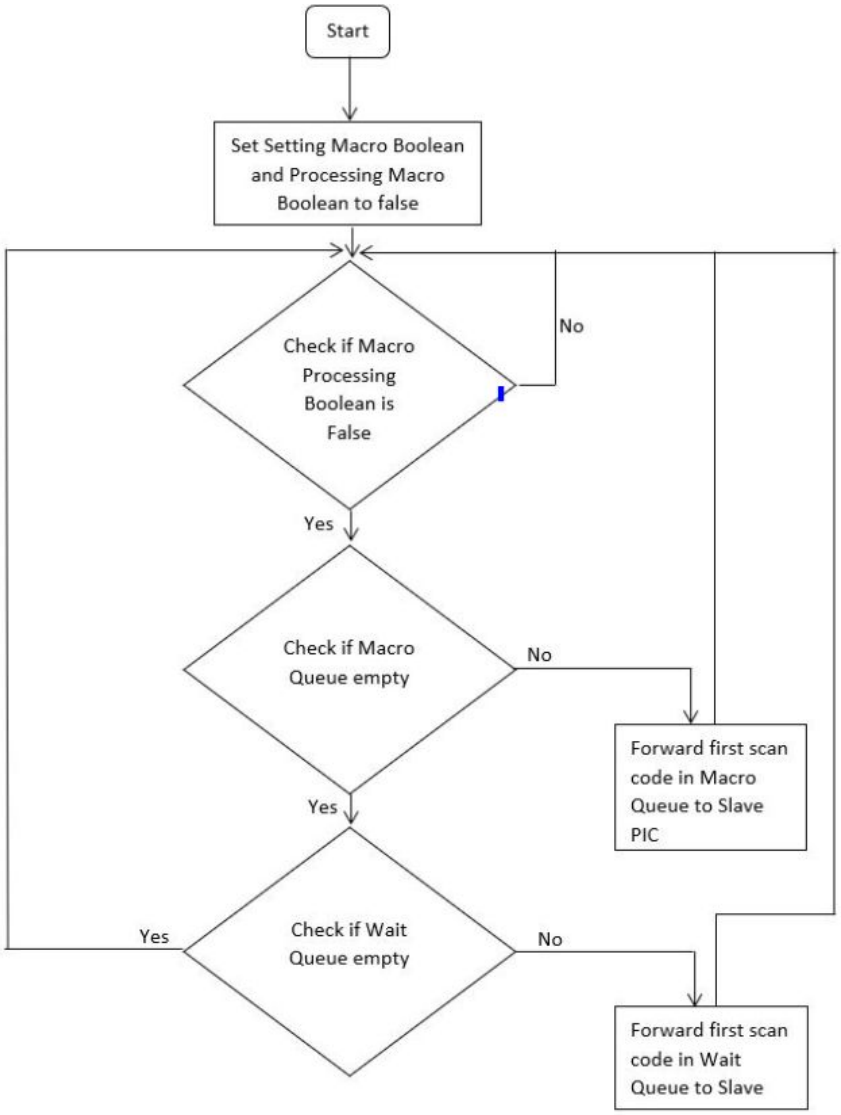

Figure 9: Control Flow of Master Microprocessor during normal operation

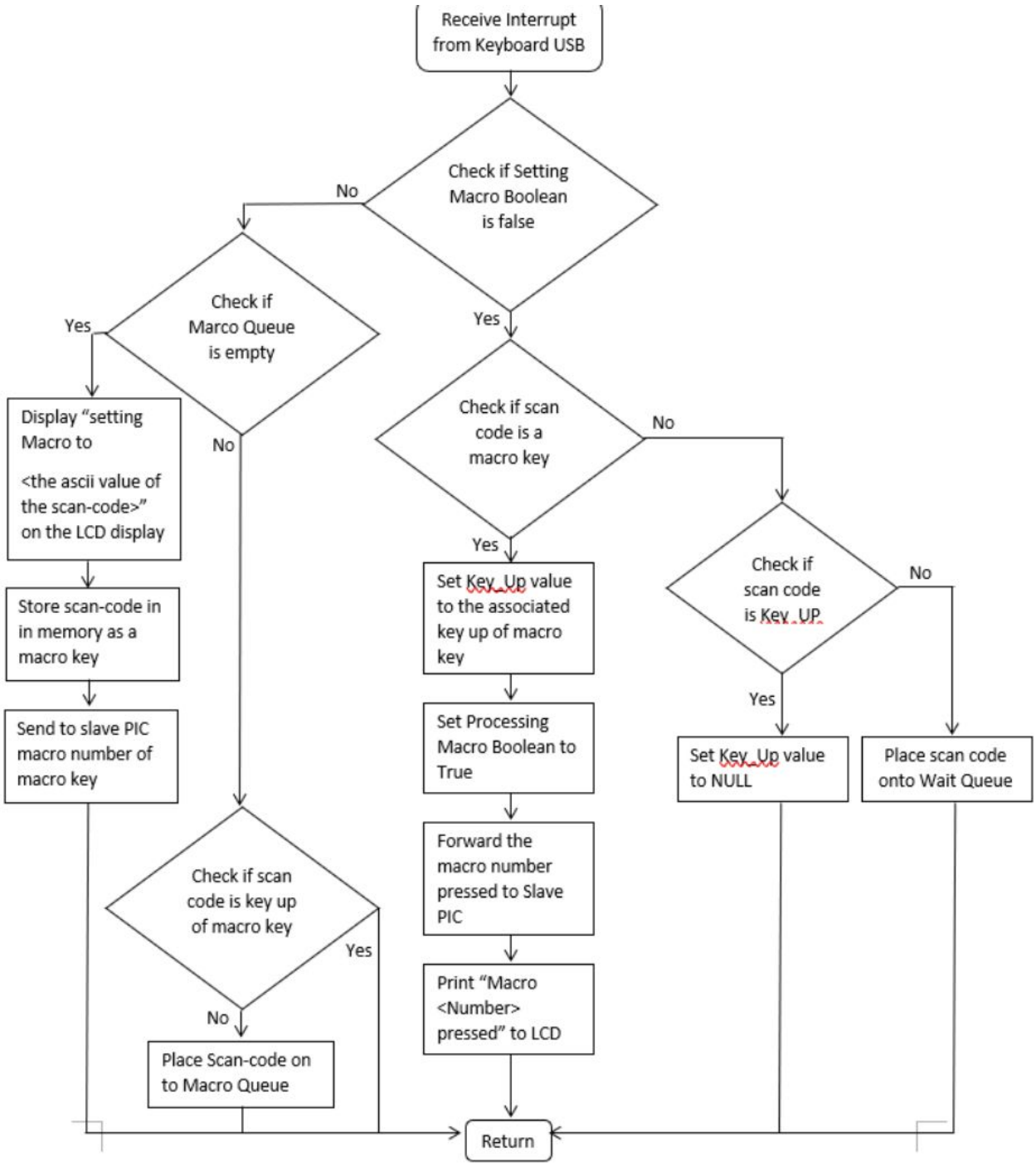

Figure 10: Control Flow of Master Microprocessor when interrupt received from Keyboard USB port

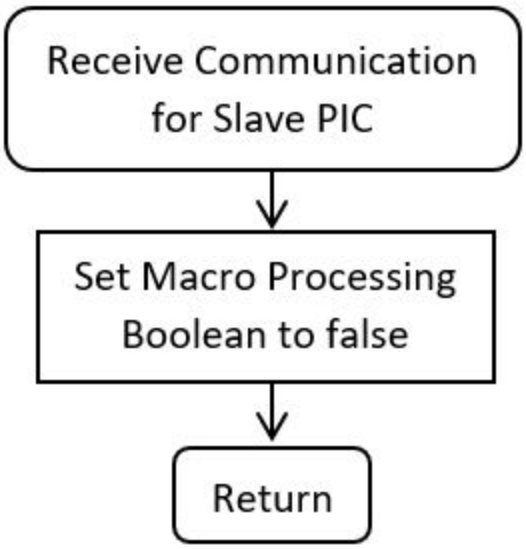

Figure 11: Control Flow of Master Microprocessor when communication received from Slave Microprocessor

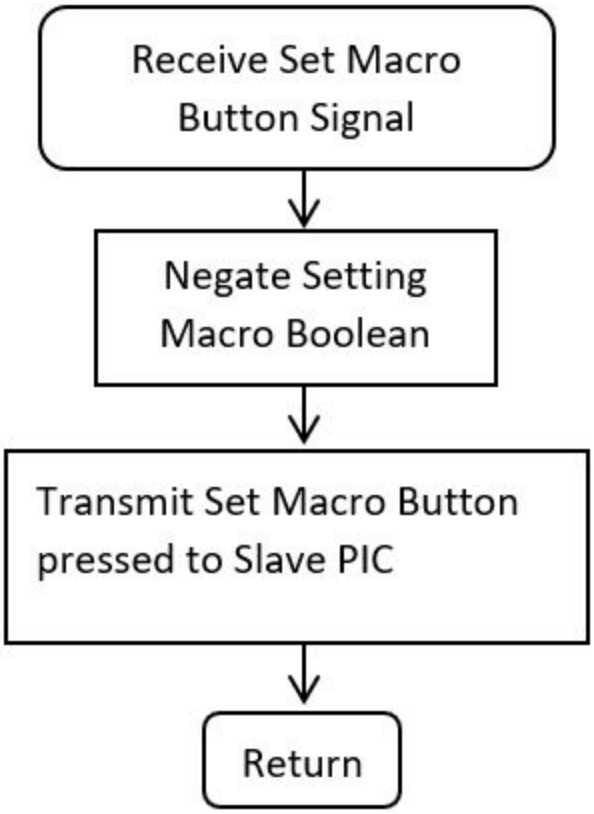

Figure 12: Control Flow of Master Microprocessor when Set Macro Button pressed

The Slave Microcontroller takes on the roles of storing and outputting the array of scan-codes associated with each of the macro keys to the computer. It is also responsible for managing some control booleans that effect the Master Microcontroller's operation. Figure 12 illustrates the control logic of the Slave Microprocessor during normal operation, while Figure 13 describes the control flow communication received from the Master Microcontroller.

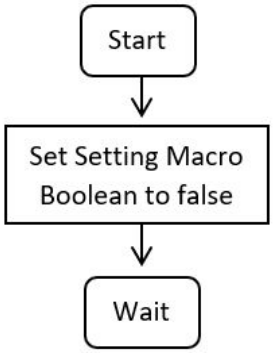

Figure 13: Control Flow of Master Microprocessor during normal operation

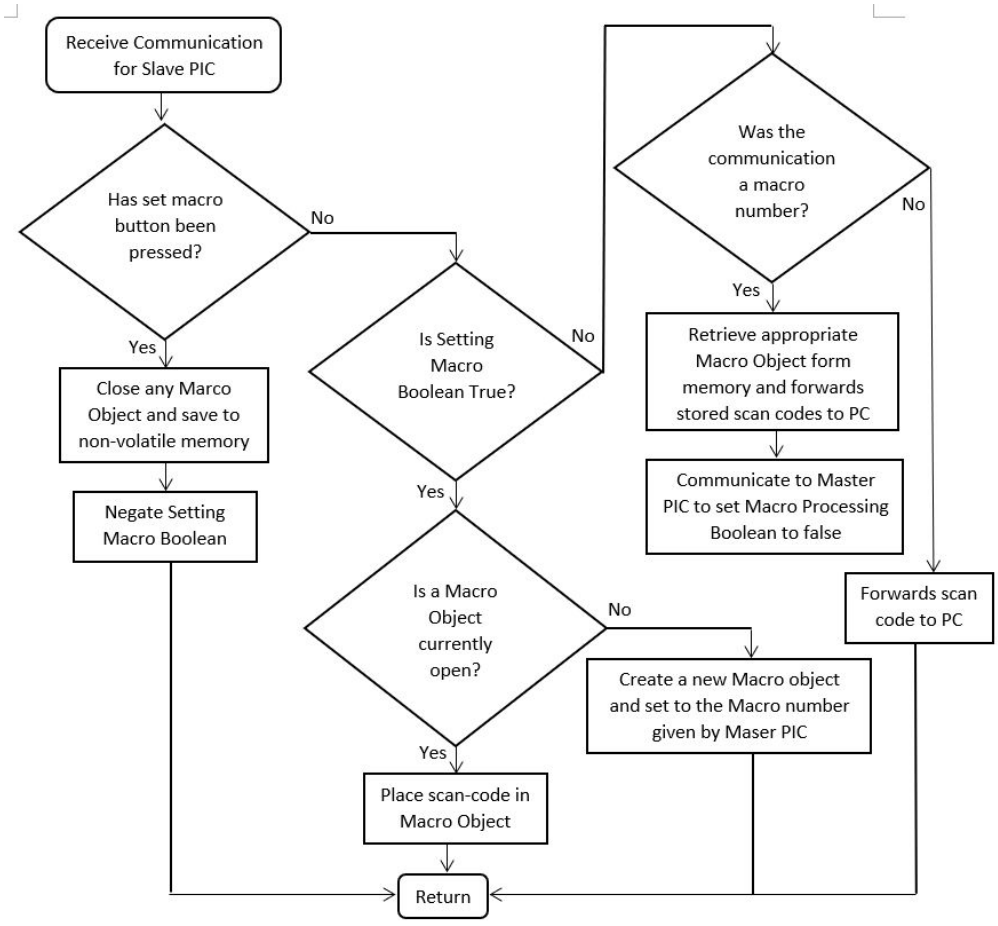

Figure 14: Control Flow of Master Microprocessor when communication received from Slave Microprocessor

3.5 Tolerance Analysis

The core component of this project that will pose limitations to this project are the data storage capabilities of the the microprocessor that we work with and the latency caused by the data manipulation that our device performs to the keyboard output that enables the usage of the hardware programmable keys. Each of these limitations have arose from our choice of hardware components. Specifically the limitations of the PIC18f2550 microprocessor that implements the core control logic of our device. In the following sections we will further discuss these limitations and the tolerance ranges that we have chosen to ensure that our device will be able to successfully function .

Data Storage

Due to the limiting nature of the microprocessors we are using, we have a limit in the space available for us to use. Looking at the Data Sheet we see that the PIC18F2550 microprocessor boasts 2,048 Bytes of RAM. However, we cannot guarantee a secure supply of power throughout the devices operation; as our device is not powered by an internal powers source, but rather relies on the power drawn by the USB port. Therefore, since this is volatile memory, it is not an effective measure of the space we have available to store our macros. Further, in order to have the user's set macros remain even after the device is unplugged, non-volatile memory will need to be employed. Another consideration to note, is that while the PIC18F2550 microprocessor also contains a large amount of non-volatile memory, the majority of it is allocated towards Program Memory and will not be usable for us to store scan-codes. Upon further reading of the Datasheet informs us that the PIC18F2550 microprocessor has 256 Bytes allocated for data storage.

Our next step is to calculate the estimated amount of storage space required for the average macro. As each macro can be of different length this is a variable amount. It is also important to understand how the space is being used between the two microprocessors that comprise our control system. Analysing the flow charts we discover that while the Master Microprocessor stores the scan code for the macro key, it is the Slave Microprocessor that stores all of the scan codes that are to be outputted to the PC when the macro is initiated. As macros are generally composed of multiple key presses, it is the data storage space available in the Slave Microprocessor that becomes the major bottleneck of our project.

Scan-codes are 1 to 3 byte codes that are outputted by the keyboard to the PC. However, the USB HID simplifies this to uniform 1 byte scan codes. It is also important to consider that pressing a key down and releasing it generates two separate scan codes, and while this is possible to get the intended effect of your macro by simply outputting the the key down scan codes, it can cause unintended consequences if the PC thinks you are holding down the keys in your macro. Further, we estimate the average macro created by our design will contain 3 distinct key presses, which totals in 6 scancodes per macro.

$$256\ \text{Bytes} / (1\text{Byte}*6) = 42.67 \qquad \text{Eq. 5}$$

As shown in the above calculation we should then on average be able to store 42 different macros on our device. This model however does not take into account any of the overheads associated with the Macro objects we are storing in memory. We would need to store pointers to the beginning of each of these objects. Furthermore, there needs to be a way for the program to tell where the list of scan codes for each of these macro ends. The easiest way of which would be to null terminate the Scan-Code Objects. This would add an additional 2 bytes per macro as a bare minimum storage requirement.

$$256\ \text{Bytes} / (1\text{Byte}*6+2\text{Bytes}) = 32 \qquad \text{Eq. 6}$$

Therefore, our final analysis results in us having a maximum of 32 programmable macros in our device. It should be noted however that the real number would be smaller in practice. There is also a concern of buffer overflow if the user creates macros much larger than our expected bounds. Therefore we have decided to limit the macro count to 10 and the number of keystrokes in the macro to 3 to ensure that in no circumstance do we extent beyond our available space

Latency

The key to understanding the latency restriction is that we must compare the operating clock cycles of our hardware with the fastest output load possible from the keyboard. The key factor to consider here is the polling rate of the keyboard. That is the number of times in a second that the computer checks for a new input from the keyboard. This value can vary greatly, while standard keyboards tend to have a polling rate of 125Hz, faster gaming keyboards can have up to 1000Hz polling rate.

The next key element is that we need to compare this polling rate to the clock cycle of the microprocessors. The clock input of our microcontrollers are 20 MHz. However, since we are setting the microcontroller to the HS PLL mode in order to be USB compliant, the actual clock cycle that the microcontrollers run at is 24MHz.

$$24\ \text{Mhz} / 1000\text{Hz} = 24000\ \text{cycles} \qquad \text{Eq. 7}$$

This means that 24000 instructions can be run on the microcontroller between the fastest possible 2 keystrokes. While this number seems a lot, it is important to consider that even the most basic assignment line of embedded C code can require multiple instructions and clock cycles to be serviced. This means it is integral that we test the runtime of our code to ensure that it does not take more than 2400 instructions. However, it is important to note that this is an extremely conservative restriction we are placing on ourselves due to some keyboards may have a 1000 Hz poll rate that is far too high of a typing speed to be seen in practical use.

4. Requirement and Verification

Table: The requirement and verification for the Power Supply

Requirement	Verification	Point
Voltage Regulator		6
The voltage across the output and ground for LM317-1 must be 3.3V with a max error of 5%.	1a. Connect USB 2.0 to the input of the regulator to provide a voltage of roughly 5V. 1b. Connect a Digit Multimeter to the output of the circuit and ground in order to measure the voltage. 1c. Verify that output is within +/- 5% error.	2
The voltage must remain stable during the power supply from the USB 2.0.	2a. Have power supply connected to provide power. 2b. Use Digit Multimeter to measure output as the power supply output is altered above 5V. 2c. Verify that output is within +/- 5% error.	2
The voltage across the output and ground for LM317-2 must be 5V with max error of 5%	1a. Connect USB 2.0 to the input of the regulator to provide a voltage of roughly 5V. 1b. Connect a Digit Multimeter to the output of the circuit and ground in order to measure the voltage 1c. Verify that output is within +/- 5% error.	2

Requirement	Verification	Point
USB 2.0 Power Supply		2
The voltage across the output and ground must be 5V with max error of 5%	1a. Connect output of the USB 2.0 power supply to a Digit Multimeter and measure the voltage.	2

	1b. Verify that output is within +/- 5% error.	

Requirement	Verification	Point
LED Power		2
The voltage provided to pin 15, LED+ must read 5V while pin 16, LED- reads 0V providing the correct power and ground for the LED display with max error of 5%.	1a. Connect the input to pin 15 and ground to pin 16 of the LED chip to a Digit Multimeter and measure the voltage. 2a. Verify that output is withint +/- 5% error.	2

Requirement	Verification	Point
LED Display		10
Displays a coded string of characters correctly	1a. Utilize the keyboard to send a bit string to the LCD through the microcontroller. 1b. The LCD has the correct string of characters on the display board. 1c. Verify that all bits of the string are able to produce a character	10

Requirement	Verification	Point
Microprocessor		28
Master microcontroller correctly receive scancodes and recognises when macro key has been pressed	1a. By outputting to the LCD display when a macro key is pressed on the keyboard we can effectively test whether the Master microcontroller is recognising the scan codes correctly.	6

Master microcontroller correctly outputs to LCD display	1a. Run tests to have the Master microcontroller output a predetermined output to the LCD display and examine if the expected outputs are correctly displayed. This verifies the code for communication between the microcontroller and the LCD display to be correct	5
Have correctly implemented Inter-integrated Circuit (I2C) communication protocol between master and slave microcontrollers	1a. Have an oscilloscope analyse the waveform to see if the communication protocol is being correctly followed	5
Have the master and slave microcontrollers correctly store the EEPROM from the scan codes for the macro key and the macro.	1a. Connect the microprocessor via USB to a computer and run a test code that outputs the data stored in the EEPROM of the chips	6
Have the Slave microcontroller only send valid scan-codes to the computer	1a. Have a computer side code that prints out the scancode being received by the computer and iterate through each of the key presses on the keyboard to ensure that the correct scan code is being outputted as pet USB specification.	6

Requirement	Verification	Point
Set Macro Button		2
Have the Set Macro Button correctly register button presses, with less than 2% of false positives or unwanted double click being registered	1a. Connect Button output to to multimeter to see if the presses are registered correctly, collect the frequency data on the false positives to ensure that the	2

	requirements are being met	

5. Cost Analysis and Schedule

5.1 Cost Analysis

5.1.1 Parts

Part Name	Part Number	Unit Cost	Quantity	Total
Microcontroller	MSP-EXP430F5529LP	$13.49	2	$26.98
Voltage Regulator	LM317	$0.98	4	$3.92
Keyboard	Logitech K360	$24.95	1	$24.95
Toggle Switch		$2.90	1	$2.90
LED Display	NHD-0116DZ-FL_YBW	$13.90	2	$27.80
3D Printer Usage	Outside provider	$10	1	$10
				$96.55

5.1.2 Labor

The national average salary is roughly around $67,899 for the year. [9] By performing the calculation

$$67{,}899 \ \frac{dollars}{year} \ x \ \frac{12 \ months}{year} \ x \ \frac{4 \ weeks}{month} \ x \ \frac{40 \ hours}{week} \qquad \text{Eq. 8}$$

we can gather that the average weekly wage is around 35 dollars for the average work week. Using this information we can gather the labor cost for each partner estimating roughly 20 hours per week worth of work for the 13 weeks since partners have been assigned.

$$20 \ \frac{hours}{week} \ x \ 13 \ weeks \ x \ 35 \ \frac{dollars}{hour} \ x \ 2.5 = \$22750 \ per \ person \qquad \text{Eq.9}$$

5.1.3 Grand Total of Labor

Name	Hours Invested	Hourly Rate	Total Cost
Nigel H.	260	$35	$22750
Jeevitesh J.	260	$35	$22750
3D Printer Provider	2	$8.25	$16.50
Total	340		$45,516.5

5.2 Schedule

Week	Nigel Haran	Jeevitesh Juneja
2/5	Finalize Project Proposal	Review Project Proposal
2/12	Research and select voltage regulator Research and select keyboard Research and select LCD display	Research and select microcontroller
2/19	Finalize Design Document Apply extra hardware to project	Review Design Document Prepare for design review
2/25	Study Datasheets Purchase all hardware and parts	Begin programming microcontroller
3/4	Design PCB Build prototype on breadboard	Test microcontroller
3/11	Put in the order for the PCB Debug prototype	Interface microcontroller with keyboard Interface microcontroller with LCD
3/18	Work on Final Report Request 3D print of casing	Run tests

3/25	Set up PCB	Run tests on the project
4/1	Test and debug power measurements	Test and Debug microcontroller
4/8	Prepare for mock demo	Optimize all parts
4/15	Prepare Presentation	Prepare Demo
4/22	Finalize Paper	Prepare Presentation
4/29	Turn in Final Paper Lab Checkout	Proofread Final Paper

6. Ethics And Safety:

In general there is very little health and safety risks. Standard safety issues include the risk of developing arthritis and discomfort during the use of the keyboard. [3] However, in order to prevent any safety risks during the development of the keyboard, we must make sure to follow the safety guidelines that were addressed during the lab safety tests. We must also respect the designs of previous keyboards and give credit where it is due. We must also prevent any misuse of our product and condone the use of our product to commit unethical actions. Another possible ethical issue to address is to accept when an area of knowledge is outside our area of expertise and to obtain help through research and other outside resources. [1]

In order to comply with the first established policy of the IEEE Code of Ethics we shall make sure to establish materials like a voltage regulator in order to limit the amount of power flowing through the device and not put the user at risk. We will also make sure to implement the standard safety principles for the safety of the consumer and workers such as no exposed wires, materials that are not toxic, and insulators that will prevent shocks. To comply with the second law of the IEEE code of ethics we will establish all interests and address when there is a conflict between them. If there is a conflict of interests, we will try to resolve the conflict on our own. If the conflict can not be resolved, we will bring an outside party to give their input. The third code of ethics can be addressed by researching concepts that are confusing until and understanding is developed. [1]

The fourth code will be addressed by not performing any acts of bribery or to accept any forms of bribery. It shall also be addressed by reporting any case of bribery that occurs during the development process. [1] The sixth code of ethics will be addressed by performing what we can do with our knowledge. If we have an issue during the development, we will seek help from the TA and course staff when facing an issue that cannot be solved by us alone. The seventh code of ethics will be addressed by taking all criticism from the assigned TA, course staff, and fellow peers. We will work together in order to resolve any issue that is addressed by outside criticism. The ninth code of ethics will be resolved by following the safety principles that were addressed during our safety test and to follow all the principles established during the safety training portion of the course. The tenth code of ethics will be addressed by offering assistance and guidance to fellow students and partners in order to provide improvement in both the product and the individual behind it. [1]

We will honor the 1.5 ACM code of ethics by looking into the patent of any design aspect we are considering to use and make sure not to wrongfully use something that will break copyrights. We will honor the 1.6 ACM code of ethics by giving credit to all design aspects that have already existed such as the general design of keyboards and the concept of a programmable keyboard. We will honor the 2.1 ACM code by not using materials that cut in costs or quality of the product.

We will make sure to use materials that are for the best of our design and not place peers or consumers at risk. [2] Finally we will address the risk of privacy by clearly placing the uses of the product so that it cannot be used as a keylogger. We will address how this is a breach of the terms & agreements of the product securing the safety of the user and providing proper punishment for the unethical uses of the product.

Citations

[1]Ieee.org, "IEEE IEEE Code of Ethics", 2018. [Online] Available: http://www.ieee.org/about/corporate/governance/p7-8.html. Accessed: 6 Feb 2018.

[2]acm.org, “ACM Code of Ethics and Professional Conduct,” 2018. Available: https://www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct#sect2. Accessed 6 Feb 2018

[3]workplaceohs.com, “Computers,” 2018. [Online] Available: http://workplaceohs.com.au/hazards/office-safety/computers . Accessed: 6 Feb 2018.

[4]logitechg.com, “G810 Orion Spectrum,” [Online] Available: https://www.logitechg.com/en-us/product/g810-orion-spectrum-rgb-gaming-keyboard . Accessed 10 Feb 2018.

[5]logickeyboard.com, “Avid Media Composer PC Backlit Astra Keyboard,” [Online] Available: http://logickeyboard.com/shop/avid-media-composer-astra-backlit-pc-keyboard-3417p.html . Accessed: 10 Feb 2018.

[6]superuser.com, “What is the power output of a USB port,” [Online] Available: https://superuser.com/questions/690074/what-is-the-power-output-of-a-usb-port . Accessed: 6 Feb 2018.

[7]mouser.com, “NHD-0116DZ-FL-YBW,” [Online] Available: https://www.mouser.com/ds/2/291/NHD-0116DZ-FL-YBW-34847.pdf . Accessed: 18 Feb 2018

[8]ti.com, “LM317 - 3 Terminal Adjustable Regulator,” [Online] Available: http://www.ti.com/lit/ds/symlink/lm317.pdf . Accessed: 13 Feb 2018

[9] ece.illinois.edu, “Salary Averages,” [Online], Available: https://ece.illinois.edu/admissions/why-ece/salary-averages.asp . Accessed: 23 Feb 2018

[10] microchip.com, “PIC18F2550,” [Online] Available: https://www.microchip.com/wwwproducts/en/PIC18F2550. Accessed: 11 Feb 2018

[11] microchip.com, “USB MCUs & dsPIC,” [Online] Available: http://www.microchip.com/design-centers/usb/usb-pic-reg-mcus-dspic-reg-dscs . Accessed: 11 Feb 2018

[12] hades.mech.northwestern.edu, “SPI Communication between PICs,” [Online] Available: http://hades.mech.northwestern.edu/index.php/SPI_communication_between_PICs Accessed: 12 Feb 2018

[13] microchip.com, “PIC18F2455/2550/4455/4550 Data Sheet,” [Online] Available/: http://ww1.microchip.com/downloads/en/DeviceDoc/39632e.pdf Accessed: 11 Feb 2018

[14] picturial.net, “Learn Pic Microcontroller Programming,” [Online] Available: http://www.pictutorial.net/2016/09/usb-hid-bootloder-for-pic18f-what-is.html Accessed: 16 Feb 2018