Dynamic Keyboard

By

Jeevitesh Juneja

Nigel Haran

Final Report for ECE 445, Senior Design, Spring 2018

TA: Xinrui Zhu

2 May 2018

Project No. 20

Abstract

Our design is creating a keyboard add on that allows any keyboard to act as a dynamic keyboard storing up to ten macros that are programmed through the hardware itself. Our design was initially through two microcontrollers acting as a master and slave microcontroller. After discovering its inability to act as a host we updated our design to contain three different parts, the leonardo arduino clone, USB host shield, and LCD. Our resulting design ended with a keyboard that can store up to ten macros. Each of these macros can store a combination of three keys while providing feedback through the LCD.

Contents

1 Introduction..... 3
1.1 Objective..... 3
1.2 High Level Requirements..... 3
2 Design..... 3
2.1 Block Diagram..... 3
2.2 Hardware Design..... 4
2.2.1 Power System..... 4
2.2.1.1 Voltage Regulator..... 4
2.2.1.2 Voltage Divider..... 5
2.2.1.3 Zener Diode..... 5
2.3 Software Design..... 6
3 Design Verification..... 8
3.1 Power System..... 8
3.1.1 Output Voltage of Voltage Regulator..... 8
3.1.2 Output Voltage of Voltage Divider..... 9
3.1.3 Output Voltage of Zener Diode 9
3.2 LCD Display 10
3.2.1 Display Different Interactions..... 10
3.3 USB Communication..... 11
3.3.1 USB Host Shield..... 12
3.3.2 USB HID Device..... 12
3.3.3 EEPROM..... 12
4 Costs 13
4.1 Parts 13
4.2 Labor 13
4.3 Grand Total 13
5 Conclusion 14
5.1 Accomplishments 14
5.2 Uncertainties 14
5.3 Ethical Considerations 14
5.4 Future Work..... 14
References 14
Appendix A Requirements and Verifications 16
Appendix B Schematics and Product 20

1 Introduction

1.1 Objective

What we are trying to address with our design is an innovation on a concept that has not had much innovation in the past few years. There are designs of dynamic keyboards like the Logitech G810 [1] and LogicKeyboard Astra [2] that have only made innovations in the design layout and cosmetics of the keyboard itself. Our design is innovative because it will be able to turn any keyboard into a dynamic keyboard while functioning on any computer. Our goal of this design, is to create an add on device that is capable of making any keyboard programmable with any computer. The product itself will be inexpensive allowing any person to be able to purchase and use this product. Our hope for the future is to have a device that can allow either macros of massive size or enough macros to reconstruct the layout of an entire keyboard.

1.2 High Level Requirements

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

2 Design

2.1 Block Diagram

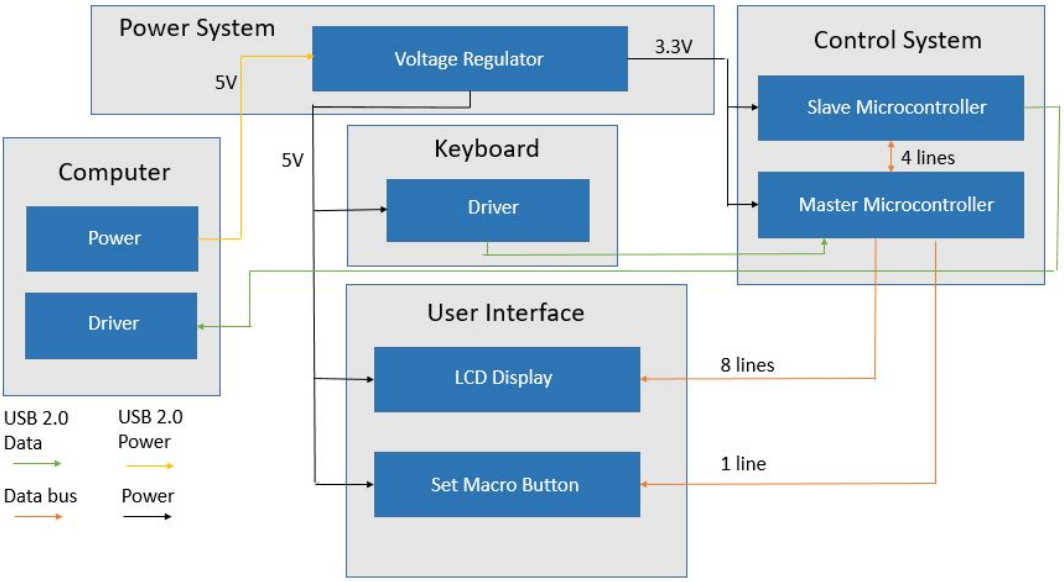

Figure 1: High Level Block Diagram

Power System: The computer itself will be used to power the circuit. The voltage is regulated by a voltage regulator to 3.3 V and is fed into the USB host shield. The voltage is also regulated by a zener diode to 4.2V and is fed into the keyboard driver. The remaining components of the circuit are able to run on the 5V that the USB power provides.

Control System: The Arduino Leonardo Clone is responsible for feeding the scan codes obtained from the keyboard to the computer and LCD screen. It also feeds the programmable macros back to the keyboard through the USB Host Shield. The USB Host Shield serves to assist the Arduino to act as a USB host and assists in the transfer of information to and from the keyboard device.

User Interface: The LCD display is presented by a 16x1 bit screen that provides 16 characters. The purpose behind the LCD is to provide real time feedback to the user to help organize macros and know when to execute the actions to program a macro. The set macro button will be used in order to execute the programming of the macro, the button will be pressed once to set the macro to a key or pressed twice to reset all macros.

Keyboard: The keyboard will provide the scan codes and receive the macros that we program to it.

Computer: This will interface with the keyboard and our add on in order to execute the keys and macros that are pressed. It will also serve to provide the 5V power needed for our add on to function.

2.2 Hardware Design

2.2.1 Power System

2.2.1.1 Voltage Regulator

Input: 5V from the USB 2.0

Output: 3.3 +/- 5% power for the USB host shield

The circuit for the USB host shield contains a LM317T adjustable voltage regulator that is meant to reduce the input voltage and produce stable outputs for other parts of the circuit. We use two resistors (R1,R2) in order to adjust the output voltage. We also have a decoupling capacitor C1 in order to filter out unwanted AC noise and C2 to improve ripple removal.

The output voltage of LM317T is expressed as

$$Vout = Vref(1 + \frac{R2}{R1}) + Iadj * R2 \qquad \text{eq.1}$$

In this case V_{ref} is the voltage between the output and adjustment terminals which has a fixed value of 1.25V. The I_{adj} is meant to minimize and maintain constant values with changes to the load. However, because the current is small the value I_{adj} can be ignored. The datasheet for the LM317T suggests that the resistor for R1 should be 240 Ω and we adjust the value of R2 in order to control what the output voltage should be using the equation provided.

$$Vout = 1.25(1 + \frac{R2}{240}) \quad \text{eq.2}$$

We can rearrange the equation to find the value of R2 when we know we want 3.3V

$$R2 = 240(\frac{3.3}{1.25} - 1) = 393.6 \quad \text{eq.3}$$

From this we can see that we need a resistor of roughly 393.6 Ω, which we were able to successfully execute with a 390Ω resistor. [3]

2.2.1.2 Voltage Divider

We need the voltage divider in order to provide a reference voltage for the LCD. This is so the display can show the characters on the screen appropriately. The voltage divider involves two resistors in series where the value between the two resistors is the voltage applied to the reference point of the LCD.

The ratio of the voltages are meant to satisfy the voltage division law

$$\frac{Vout}{Vin} = \frac{Rdown}{Rup+Rdown} \quad \text{eq.4}$$

We can adjust this relationship so that we can establish a ratio between the resistors and the voltage we have.

$$\frac{Rup}{Rdown} = \frac{Vin - Vout}{Vout} \quad \text{eq.5}$$

The input voltage in this case is the 5V provided from the USB 2.0, while the output that we want is 0.8V to act as the reference for the LCD. The datasheet for the LCD says that we want the reference voltage to be around 0.5V, however when applying the voltage the best results was when we had 0.8V as the reference voltage. We apply these values to the relationship established above to obtain the resistance ratio we want.

$$\frac{Rup}{Rdown} = \frac{5-0.8}{0.8} = 5.25 \quad \text{eq.6}$$

From this ratio we originally planned for the R_{up} to be 9.1kΩ where the ratio relationship would have R_{down} be 1.73kΩ. [4]

2.2.1.3 Zener Diode

We need the zener diode in order to reduce the input voltage of 5V to 4.2V for the keyboard. This is due to the keyboard having a failsafe that shuts the keyboard down if the input voltage exceeds 5V. The design behind the zener diode involves the diode to have a resistor in series and in parallel. This reduces the output voltage from the input voltage.

The first thing we have to establish is the resistance that we would have running in series with the zener diode. This is determined by the equation

$$Rs = \frac{Vs - Vz}{Iz} = \frac{5-4.2}{0.1} = 8\Omega \quad \text{eq.7}$$

In this case V_S is the input voltage of 5V, V_Z is the desired voltage of 4.2V and I_Z is the desired output current of 100mA. We then proceeded to obtain the resistor we would want in parallel with our zener diode which is determined by the equation

$$Rl = \frac{Vz}{Il} = \frac{4.2}{0.1} = 42\Omega \quad \text{eq.8}$$

In this case V_Z is the desired voltage output and I_L is the current in parallel with the zener diode. From these equations we determined that we needed to use a 4.2 Zener Diode with an 8Ω resistor in series with the diode and 42Ω resistor in parallel so that can have a stable 4.2V power supply for the keyboard. [5]

2.3 Software Design

The software portion of the device implements the the control logic utilized by the microprocessors to provide the intended functionality of our project. The flow charts below illustrate how the user would interact with the device to set new macros and clear existing ones.

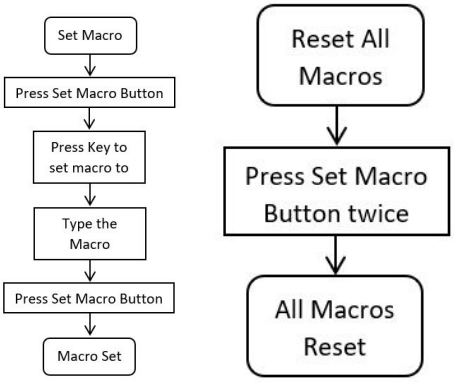

Figure 2: User Interaction Flowchart

All of this functionality is implemented by the code stored by the Leonardo microprocessor and is responsible for receiving the keyboard data from the USB Host Shield, outputting to the Computer via USB HID protocol, controlling the LCD display, storing data to EEPROM and implementing the control logic to create an output based on the defined macros. The figure below illustrates the startup protocols as well as the main control loop the code implements.

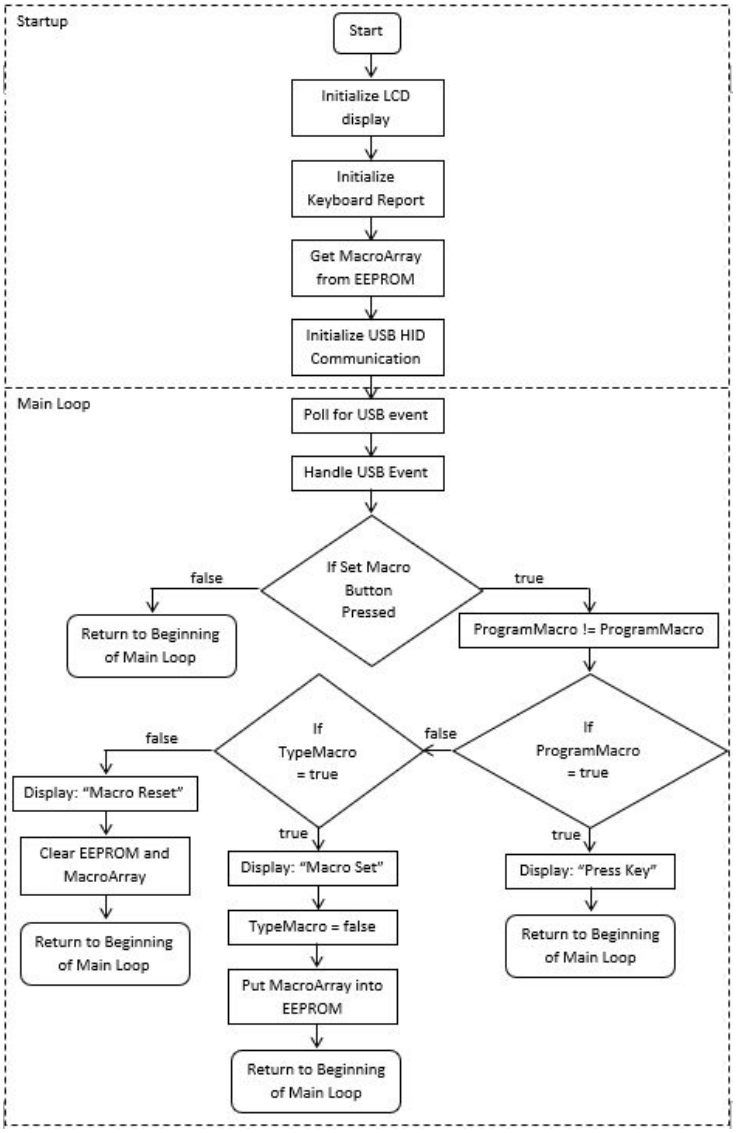

Figure 3: Control Logic of Microprocessor

The figure above abstracts the logic employed to "Handle USB Events". There are three keyboard HID events that our device can handle: Key Up, Key Down, and Modifier Change. The control logic of the USB event handlers detailed in the figure below is responsible for both outputting the computer's keystrokes and macros pressed by the user while updating the MacroArray structure. This is responsible for storing the user defined macros. These events need to be handled differently based on whether the device is in standard operation or the user is setting the macro. This is accomplished through the the flags ProgramMacro and TypeMacro that are set in the Main Loop above.

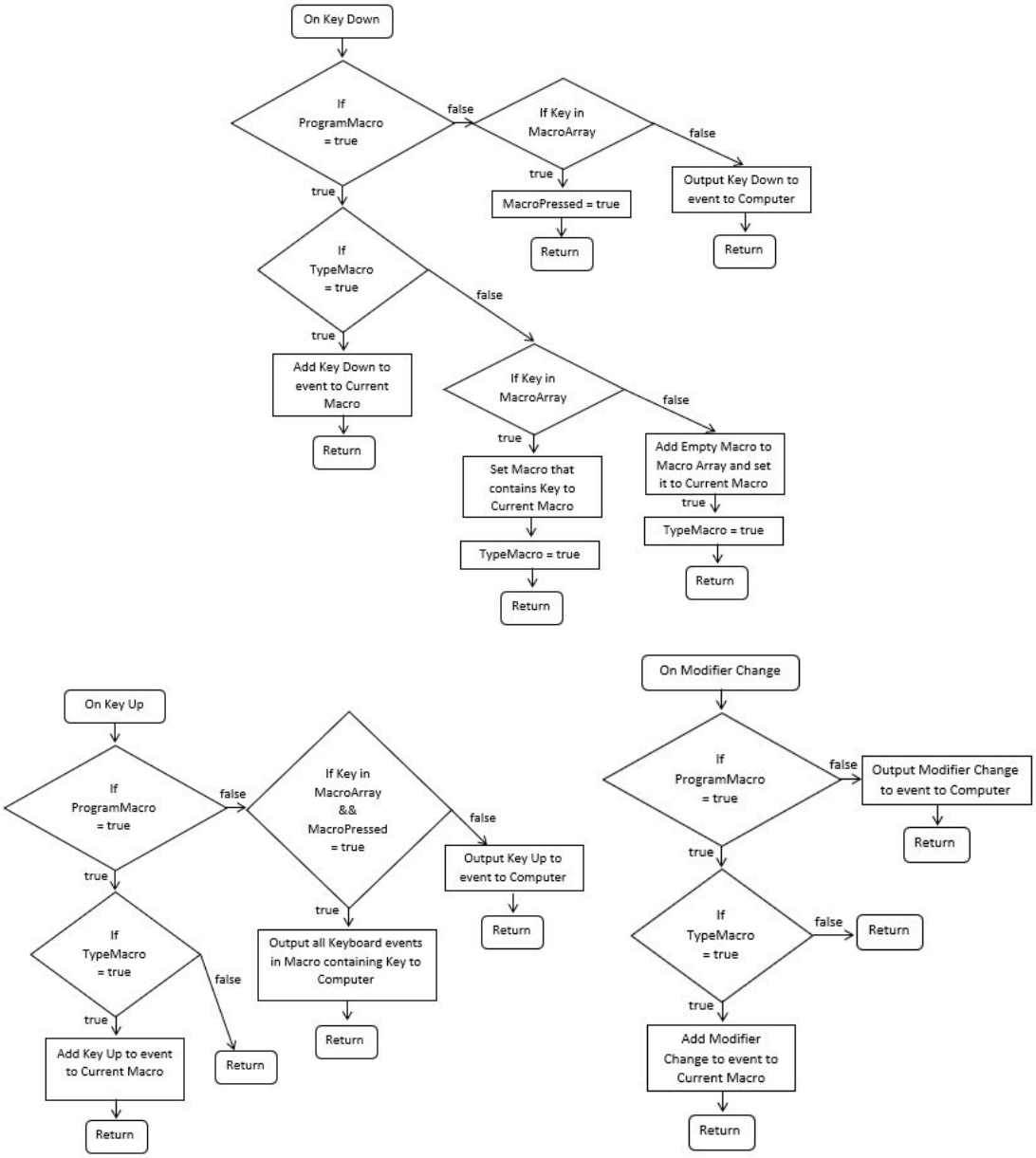

Figure 4: Control Logic of Key Down, Key Up, and Modifier Change keyboard events

3 Design Verification

3.1 Power System

3.1.1 Output voltage of the 3.3V regulator

To verify the voltage regulator output, we connected the input to a DC voltage generator and the output of the regulator to a multimeter. We then made the range of the input voltage between 0

and 7V and read how it affected the output of the 3.3V regulator. The result is show in the chart below.

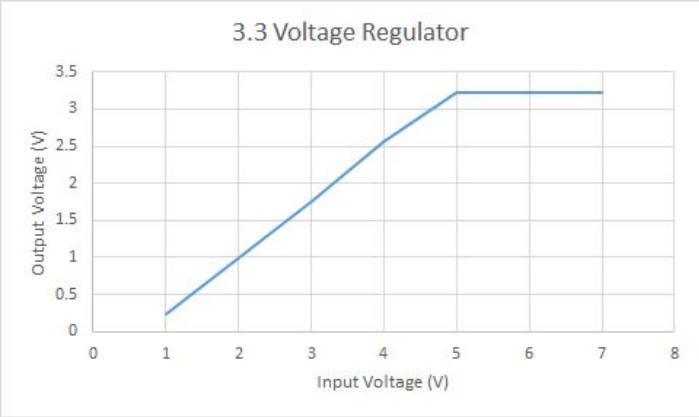

Figure 5: The measured output voltage for the 3.3V regulator

From this figure, we can see that the voltage regulator consistently produces around 3.3V when the input voltage is greater than 5V. The specific measured voltage was 3.23V. We can measure the error through the equation [3]

$$\frac{V\,theoretical - V\,measured}{V\,theoretical} = \frac{3.3-3.23}{3.3} = 0.0212 * 100 = 2.12\% \quad \text{eq.9}$$

3.1.2 Output Voltage of Voltage Divider

To verify the voltage divider output, we connected the input to the 5V USB source and the output of the divider to a multimeter. We then proceeded to measure the output voltage of the divider and noticed that the divider had an output of 0.803V. We can measure the error through the equation [4]

$$\frac{V\,theoretical - V\,measured}{V\,theoretical} = \frac{0.8-0.803}{0.8} = 0.00375 * 100 = 0.375\% \quad \text{eq.10}$$

3.1.3 Output Voltage of Zener Diode

To verify the Zener diode output, we connected the input to the 5V USB source and the output of the diode to a multimeter. We then proceeded to measure the output voltage of the diode and noticed that the diode had an output of 4.05V. We can measure the error through the equation [5]

$$\frac{V\,theoretical - V\,measured}{V\,theoretical} = \frac{4.2-4.05}{4.2} = 0.0357 * 100 = 3.57\% \quad \text{eq.11}$$

3.2 LCD Display

3.2.1 Display Different Interactions

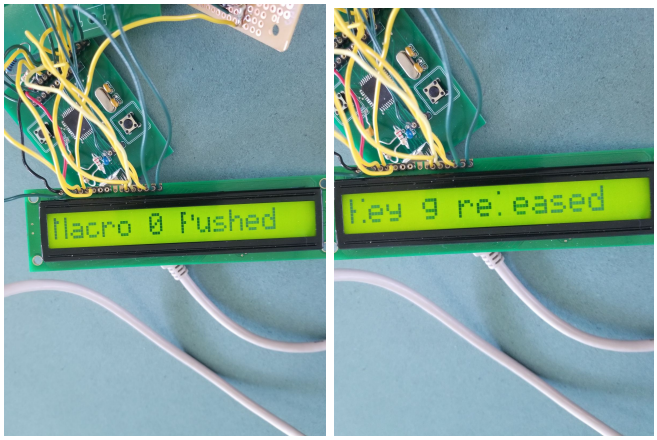

Figure 6: LCD display showing execution of key releases and macros releases

We tested the LCD when our entire product was set up and functioning. The way we would test the basic display of our LCD was by having a macro set to one of the keys. We proceeded to type on the keyboard and check to see if the LCD was displaying the keyboard keys that were released and the macros that were released. We can see from the images above that the LCD display was showing what we wanted. [6]

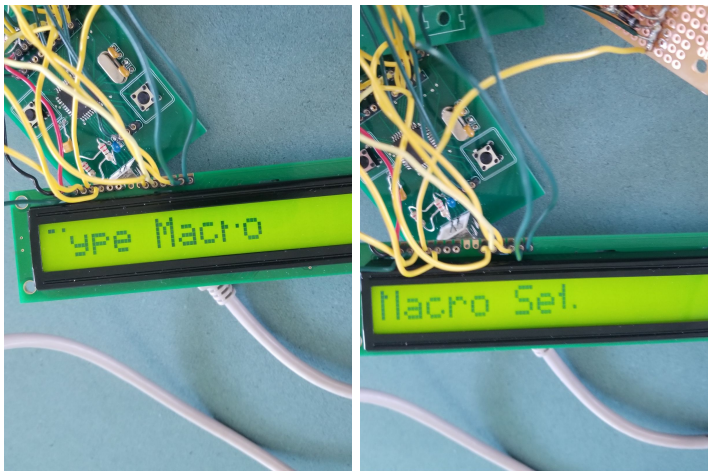

Figure 7: Instruction set of programming a macro

We then proceeded to investigate whether or not the LCD display would display the instructions to set a macro. We would go through the process of pressing the button to set a macro in which it would display “type macro,” where you would press a key on the keyboard and then press the combination of keys. We would then press the macro button again and it should display “macro

set.” The images above verify that the LCD display is capable of displaying the instructions to set the macro. [6]

Figure 8: Display of macro reset

We then proceeded to investigate if the LCD display could display that the macros had been reset. We would proceed to do this by pressing the set macro button twice in order to initiate the reset macro protocol. We can confirm that our display was fully working if it displayed “macros reset,” in which it did. This is shown through the image above and further verifies the functionality of our LCD display. [6]

3.3 USB Communication

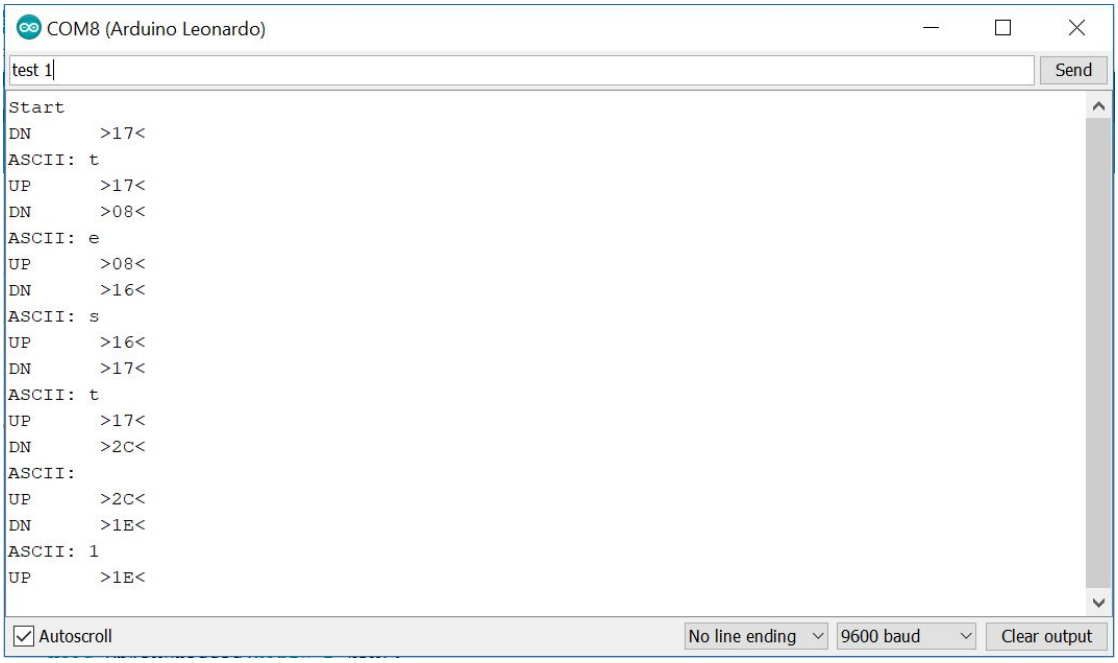

Figure 9: Serial Port output of USB communication test

3.3.1 USB Host Shield

We were able to test host shield communication by displaying the scan codes received from the Keyboard through the serial port. This is shown in the figure above when the input “test 1” was typed on the keyboard plugged into the device. Our test code then printed the scan codes to the serial port as well as whether it was a key down, key up, or modifier change. The Ascii value, if available, of the scan code during key down events are also displayed for ease of testing purposes. We tested each of the 104 keys on our keyboard and confirmed that the correct scan codes were displayed. This ensured that our device correctly functioned as a USB host and read inputs from our keyboard. [7]

3.3.2 USB HID Device

We were also able to use the above test code to verify that our device functioned as a USB device and outputted correctly to our device. As shown in the figure above, when the input “test 1” was typed on the keyboard that was plugged into our device, the text was also typed into the empty text box of the Serial Port window. We further tested this by opening a word file and typing various letters, holding down keys, and inputting complex combinations of keystrokes such as Ctrl+Alt+Delete through the keyboard plugged into our device. We then checked to see if the expected result was outputted to the computer. Lastly, we tested whether the Caps Lock, Num Lock and Scroll Lock LEDs functioned correctly when the keyboard was plugged into our device as these LEDs are controlled by the computer in USB keyboards. All of this confirmed that device also functioned correctly as a USB HID device. [8]

3.4 EEPROM

We tested the proper storage and retrieval of macros to non-volatile memory by creating macros on our device using our fully implemented code. We then unplugged our device from the computer and waited several minutes. Due to the device is completely powered by the 5V Vcc output from the computer’s USB port, all data stored in volatile memory would be erased. We then plugged our device into a seperate computer and tested whether the macros we had previously created still functioned. Finally, we initiated macro reset by pressing the Set Macro Button twice and checked whether the macros were properly erased. This test confirmed that we correctly stored and loaded data to and from the EEPROM. [8]

3.5 Control Logic

Finally, we validated the functionality of the control logic by running the completed code on the integrated device. We confirmed that we were able to store and output ten macros, each a maximum of three keystrokes in length. We used two different USB Keyboards and three different computers, of which two ran on Windows OS and one ran on Mac OS. We also ran a Linux virtual machine on one of Windows computers and tested that the device with Linux OS as well. This not only verified the correct implementation of our control logic but also integration of each of the components listed previously.

4 Costs

4.1 Parts

Table 1: Component Costs

Part Name	Part Number	Unit Cost	Quantity	Total
Leonardo	AtMega34u4	$6.95	1	$6.95
USB Host Shield	MAX3421e	$10.25	1	$10.25
Voltage Regulator	LM317T	$1.59	1	$1.59
Zener Diode		$0.19	1	$0.19
Keyboard	k120	$14.66	1	$14.66
PCB		$26	4	$104
LCD	NHD-0116-DZ-FL-YBW	$13.90	1	$13.90
Op Amp		$0.62	6	$3.72
Total				$155.26

4.2 Labor

Table 2: Labor Costs

Name	Hours Invested	Hourly Rate [9]	Total Cost = Rate*Hours*2.5
Nigel Haran	170	$33	$14,025
Jeevitesh Juneja	170	$33	$14,025
Total	340		$28,050

4.3 Grand Total

Table 3: Total Costs (Components Cost + Labor Cost)

Parts	Labor	Grand Total
155.26	28,050	$28,205.26

5. Conclusion

5.1 Accomplishment

Our team was able to successfully develop a hardware implemented dynamic keyboard. The LCD screen was able to successfully display the keys that were being pressed and which macros are being pressed. The LCD was also able to display the instructions when setting a macro and when the macros were being reset. The USB host shield and Leonardo clone were able to successful communicate with each other. Due to these working we were able to successfully create an add on that allows us to have a key on the keyboard represent a combination of keys. We were able to have up to 10 programmable macros that can be rewritten and reset. Thus, we were able to execute all the requirements we had for our design.

5.2 Uncertainties

The major issue we had faced was in our initial design where we had two pic microcontrollers. However, they would not be able to act as USB hosts which caused us to scrap our entire design. We resolved this with our USB host shield and Leonardo clone. The issue that remained was our voltage divider that was used as a reference for the LCD. The voltage divider would typically have trouble staying stabled and caused some parts of the LCD screen to burn out. The LCD was also using soldered wires that had some issue coming undone or breaking off, which we would hope to resolve by having a firmer connection between the data pins of the Leonardo clone and the LCD pins.

5.3 Ethical Considerations

Throughout our design and research, we made sure to respect the codes of ethics established by the IEEE. We made sure to maintain honesty when collecting data during the verification tests and guaranteeing that successful tests were not a case of an accident. We made sure to respect any criticism that was brought to our attention and put effort into using these criticisms to improve our design. This criticism would come from our TAs, instructors, and fellow students. They assisted us by making our design achievable and making sure that we could make a product that could be functioning by the time we had to demo. We also made sure to give credit to any outside sources we used to create our product. [14] [15]

5.4 Future Work

When we made a change in our microcontrollers we noticed that there was now extra memory that could be used. Due to this extra memory, we could create enough macros to fill the entire keyboard with a combination of three keys or have limited macros that can store a combination of more than 26 keys. We could also work on having our entire design on one single PCB so

that it could be placed in one casing. This would make our product have the potential of being sold. We would also like to use an LCD that could be better soldered to our leonardo clone to guarantee secure connections. Overall, our design is fairly finished with some minor touch ups that would make it a successful product.

References

[1]logitechg.com, “G810 Orion Spectrum,” [Online] Available: https://www.logitechg.com/en-us/product/g810-orion-spectrum-rgb-gaming-keyboard . Accessed 10 Feb 2018.
[2]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.
[3]ti.com, “LM317 - 3 Terminal Adjustable Regulator,” [Online] Available: http://www.ti.com/lit/ds/symlink/lm317.pdf . Accessed: 13 Feb 2018.
[4]allaboutcircuits.com, “Voltage Divider Calculator,” [Online] Available: https://www.allaboutcircuits.com/tools/voltage-divider-calculator/ . Accessed: 10 Mar 2018.
[5]electronics-tutorials.ws, “The Zener Diode,” [Online] Available: https://www.electronics-tutorials.ws/diode/diode_7.html . Accessed: 11 Mar 2018.
[6]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
[7]MAXIM, “USB Peripheral/Host Controller with SPI Interface, MAX3421E,” [Online] Available: https://www.sparkfun.com/datasheets/DevTools/Arduino/MAX3421E.pdf . Accessed: 29 Mar 2018.
[8] Atmel, “8 bit AVR Microcontrollers, ATmega32u4,” [Online] Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Summary.pdf . Accessed: 29 Mar 2018.
[9] ece.illinois.edu, “Salary Averages,” [Online], Available: https://ece.illinois.edu/admissions/why-ece/salary-averages.asp . Accessed: 23 Feb 2018
[10] arduinolibraries.info, “USB Host Shield Library 2.0” [Online], Available: https://www.arduinolibraries.info/libraries/usb-host-shield-library-2-0 . Accessed: 04 April 2018
[11] arduino.cc, “USB Keyboard Reference”, [Online], Available: https://www.arduino.cc/reference/en/language/functions/usb/keyboard/ . Accessed: 04 April 2018
[12] www.arduino.cc, “EEPROM Library”, [Online], Available: https://www.arduino.cc/en/Reference/EEPROM . Accessed: 13 April 2018
[13] www.arduino.cc, “LiquidCrystal Library”, [Online], Available: https://www.arduino.cc/en/Reference/LiquidCrystal . Accessed: 14 April 2018
[14]Ieee.org, "IEEE IEEE Code of Ethics", 2018. [Online] Available: http://www.ieee.org/about/corporate/governance/p7-8.html. Accessed: 6 Feb 2018.

[15]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

Appendix A Requirement and Verification

Table 4: The requirement and verification table of the power system

Requirement	Verification	Verification Status
Power Supply		
The voltage across the output and ground for the LM317-T 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 5V. 1b. Connect a Digit Multimeter to the output of the circuit and the ground to measure the voltage. 1c. Verify that the output is within +/- 5% error.	Yes
The voltage must remain stable during the power supply from the USB 2.0.	2a. Have the power supply connected to provide power. 2b. Use Digit Multimeter to measure the output as the power supply is altered above 5V. 2c. Verify that the output is within +/- 5% error.	Yes
The voltage provided to pin 15, LCD+ must read 5V while pin 16, LCD- reads 0V providing the correct power and ground for the LCD display with max error of 5%.	3a. Connect the input to pin 15 and ground to pin 16 of the LCD to a Digit Multimeter and measure the voltage. 3b. Verify that the output is within +/- 5% error.	Yes

The voltage divider connected to pin 3 of the LCD reads 0.8V as a reference voltage with a max error of 10%.	4a. Connect the input to pin 3 and ground to a Digit Multimeter and measure the voltage. 4b. Verify that the output is within +/- 5% error.	Yes
The voltage across the output of the Zener Diode must read 4.2V with a max error of +/- 5%.	5a. Connect the output of the Zener Diode power supply to a Digit Multimeter and measure the voltage. 5b. Verify that the output is within +/- 5% error.	Yes

Table 5: The requirement and verification table of the LCD

Requirement	Verification	Verification Status
LCD Display		
Displays a coded string of characters correctly.	1a. Utilize the keyboard to send a bit string to the LCD through the Leonardo clone. 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.	Yes

Table 6: The requirement and verification table of the microcontrollers

Requirement	Verification	Verification Status
Leonardo Clone and Host Shield		

Leonardo clone stores up to 10 macros and is able to overwrite previous macros with new ones.	1a. Run up to 10 macros for the keyboard. 1b. Confirm that all 10 of the macros are being correctly displayed on the computer. 1c. Create an 11th macros and confirm that it overwrites the first macros.	Yes
Keyboard correctly running scan codes through USB host shield and Leonardo to the computer.	2a. Have the host shield connected to the keyboard. 2b. Connect the Leonardo to the host shield. 2c. Press keys of the keyboard and verify that those keys are being displayed on a word document. 2d. Press the keys that have macros set to them and verify that they are being displayed on the word document.	Yes
Correctly implemented communication between the Leonardo clone and USB host shield.	3a. Run test code through the Leonardo. 3b. Confirm that the keyboard connected to the USB host shield validly outputs to the serial monitor.	Yes
Have the Leonardo correctly store the EEPROM from the scan codes for the macro key.	4a. Connect the USB host shield to the keyboard. 4b. Connect the Leonardo to the computer and run a test code that outputs the data stored in the memory of the chips.	Yes

Output scan codes through the serial port.	5a. Connect USB Host shield to keyboard. 5b. Connect Leonardo to computer. 5c. Have stored scan codes output to the serial port.	Yes

Table 7: The requirement and verification table of the macro button

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

Appendix B

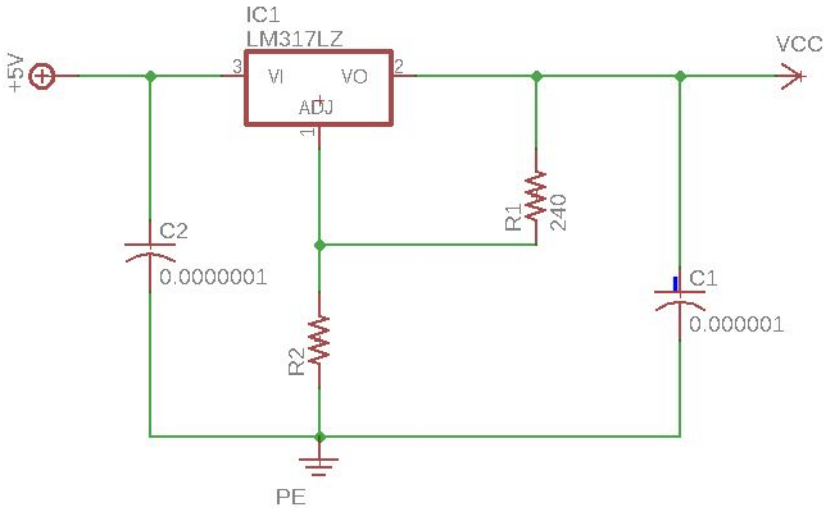

Figure 10: Schematic for 3.3V regulator

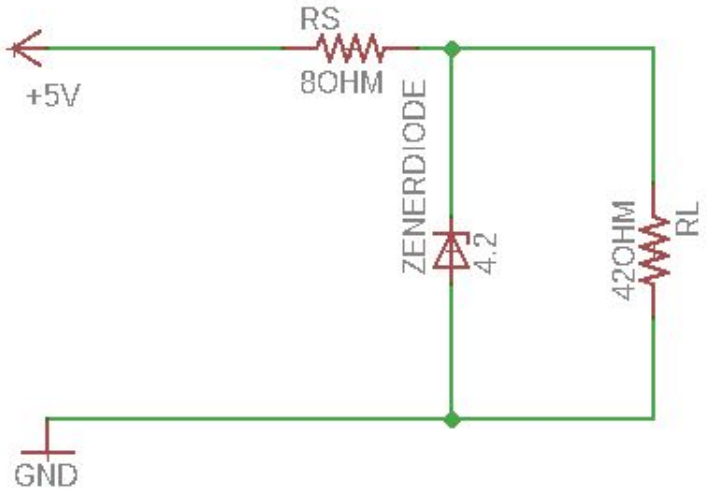

Figure 11: Schematic for Zener Diode

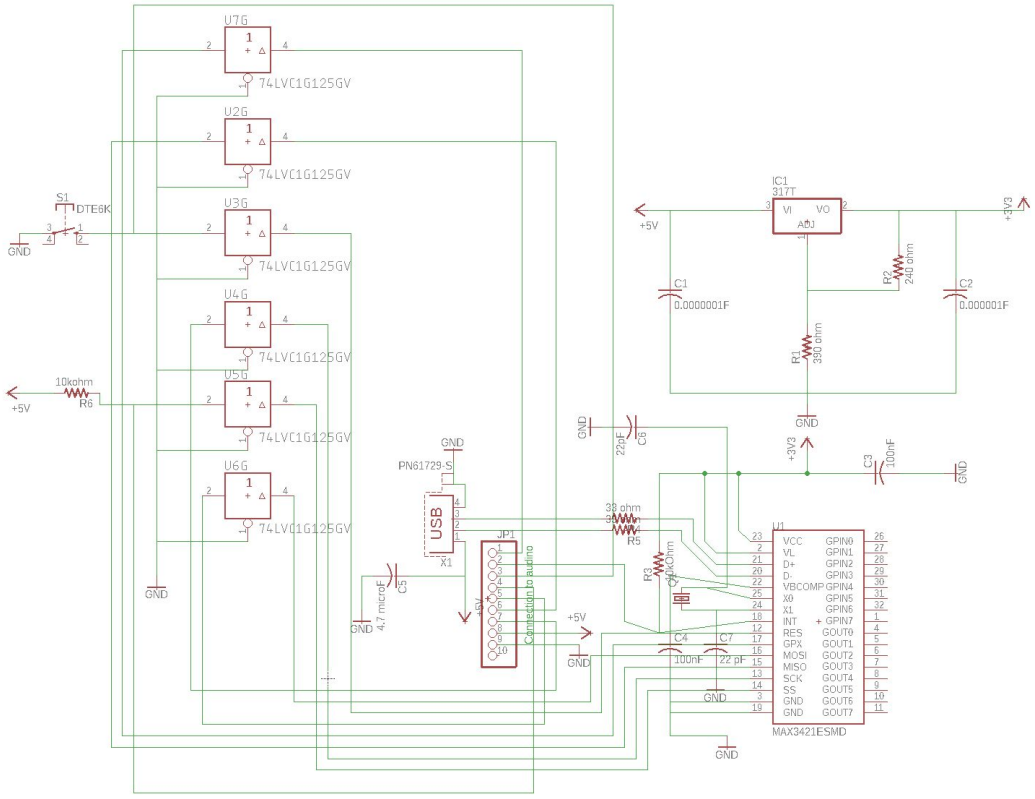

Figure 12: Schematic of USB Host Shield Clone

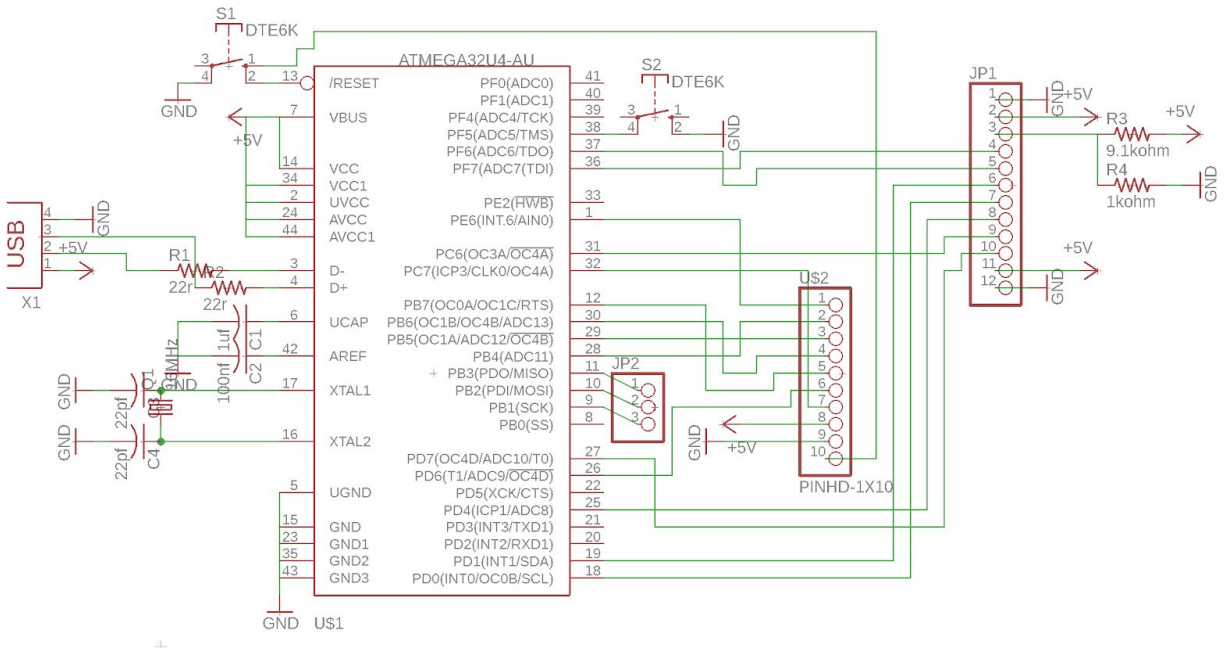

Figure 13: Schematic of Leonardo Clone

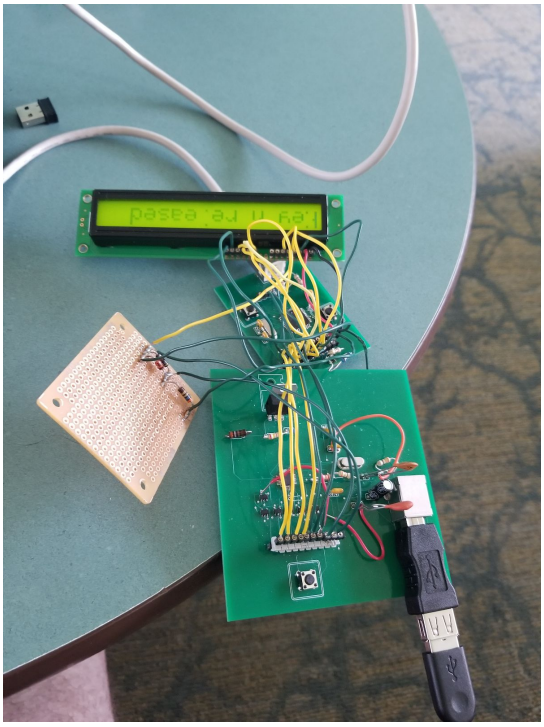

Figure 14: Complete implementation of the Design