LANGPAD: A Computer Peripheral That Allows for Easy Typing of Special Characters Used in Latin-based Languages.

A Technical Report submitted to the Department of Electrical & Computer Engineering

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Christopher Hamilton

Spring, 2022

Technical Project Team Members

Rohan Chandra

Emory Ducote

Rawan Osman

Pedro Rodriguez

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Harry C. Powell, Department of Electrical & Computer Engineering

Statement of work:

Rohan Chandra

The primary focus of my work this semester was to develop an audio subsystem that would accomplish two tasks. The first task would be to emit a tone every time the user touches a button on the LCD, indicating that their input was successfully read. The second task would be to allow the user to control the volume of this tone. In order to accomplish these tasks, I first began by developing a general structure of the system. After conducting some research, I decided on using a piezo buzzer for the audio output and a rotary encoder for the volume control, with our microcontroller serving as the interface between them. Once I had selected these components, I investigated whether we would need additional amplification circuitry in order to produce an audible tone. I consulted the datasheet of our microcontroller to see its maximum voltage and current output, as well as the datasheets for the rotary encoder and buzzer to see how much voltage and current they would need to run reliably. Once I had finalized the hardware of the audio subsystem, I began developing the software. Through online research and help from Chris, I was able to develop code that would read the states of the rotary encoder and set the volume to 1 of 4 presets, represented by a duty cycle that the PWM signals sent to the buzzer would run at. I initially tested the software and hardware on a VirtualBench, ensuring that I could accurately read the signals from the rotary encoder as well as send data to the buzzer. Once I had confirmed that, the audio subsystem was ready to be integrated into the LCD, which was done by working with Emory.

My secondary task for the project was developing the 3D printed enclosure that would house all of our hardware. In order to design this enclosure, I began by sketching out potential designs with Rawan. In these sketches, we determined where the wires would come out from, where the logo would go, and where spots would be placed for the rotary encoder shaft. I then consulted the datasheets for the microcontroller, LCD, and rotary encoder in order to get their measurements and design accordingly. Finally, I asked Chris to print each iteration of the design for me so that I could evaluate them and determine what needed to be fixed for the next iteration.

Emory Ducote

My primary task for the duration of the semester was to program anything that was sent to or received from the LCD display. The display is responsible for sending touch notifications to the microcontroller, so quick transmission of data between the two devices was required. In order to communicate with the microcontroller, the LCD uses SPI communication. Prior to this project I had no knowledge of how this communication was done. Much of the time before receiving any physical parts was spent researching SPI communication as well as combing through the LCD programming guide. Following this effort, I built out initial programs for both the LCD SPI communication as well as display libraries for on-screen graphics. At some point in this researching process, we realized that the initial LCD we chose was going to be difficult to program, so Chris and I worked to pick out a new one with a more thorough programming list.

When we received the LCD and microcontroller, I began testing my SPI and graphics libraries at once. Initial testing did not go super smoothly, and I spent about a week and a half before actually getting the display to do what I thought it would. To do my debugging, I monitored the SPI line with an Analog Discovery 2 so that I could visualize all the data being sent. The issue that took so much of my time was that the startup sequence of SPI commands that the programmers guide specified did not work as intended. However, after reaching out for help on the manufacturer's public forum page, I was able to get the LCD functioning properly, sending and receiving commands over the SPI interface. This result signified that the SPI commands were working correctly and I moved onto developing the LCD user display. I created functions that wrapped the SPI commands so that things could be displayed with higher level code. I created a basic grid of boxes and moved on to trying to decipher how to handle touch inputs. I read in the guide that a certain register had to be read to check for a touch, so I tried reading it and it worked smoothly. I then combined the interface and touch capabilities to finish the LCD work.

In addition to getting the LCD to work, I helped out with the integration of the LCD and USB communication. Chris and I spent time working on a bug with the integration, as there seemed to be a power issue when both the USB communication and LCD display were occurring. I also worked with Rohan and Chris to add a volume bar to the LCD which displays the current level of volume and changes as the user adjusts it. Finally, I also designed a knob that fits over the rotary encoder for easier volume adjustment.

Chris Hamilton

My work focused mainly on the core embedded software of our device. I designed code architecture for our project and my primary responsibility was developing code that enabled our device to type out characters on a computer. I did a lot of research to find out how this could be implemented, and I found that configuring our device according to the HID (Human Interface Device) USB protocol would be a good option. Using the USB libraries made for our MSP430 LaunchPad, I created a program that could configure our microcontroller as an HID device and type out characters. After that, I wrote a program that could type out the special accented characters needed by our device. To do this, I created a Microsoft Windows keyboard layout that had the special keys mapped to combinations of with the right alt modifier. Because our LCD was not working at the time, I added in support for a physical number pad as input and a serial terminal display as output. This allowed me to mimic the input and output functions of the LCD and progress on the implementation of the software architecture of the device. After I implemented the core functionality of the device, I added the feature for rearranging the order of the keys. To make these mappings non-volatile, I learned how to interact with the flash memory on our microcontroller. I then programmed the order of the keys to be stored and loaded from the flash memory to implement this feature.

Beyond my core software responsibilities, I created a GitHub repository and helped Emory and Rohan get their embedded software development environments setup. I helped Emory debug some of the major issues with the LCD and worked with him to integrate the LCD code into the main code. I also helped Rohan out with the printing of the enclosure that he

designed and helped him work through design issues on it. Additionally, I helped Rohan write code to interface with our buzzer.

Rawan Osman

My primary contributions to the team and our project was through the hardware design, particularly the power subsystem, PCB design and manufacturing, and hardware testing. My contributions started by outlining the initial project goals and block diagrams with Chris. After this, I took responsibility for devising our power system. I considered the necessary currents and voltages for our components - the MSP, LCD, buzzer, and rotary encoder - and used a power analysis to make design decisions around using USB power source versus wall power source. We decided on a wall power source because the initial LCD we intended to use required 19 V for backlighting. We changed to an LCD that later on only required 3.3 V and 5 V, however we decided to keep the wall power system for the complexity and challenge it offered to the project. With this decision, my role evolved into creating the block diagram for the system and using that to outline the power and data flow.

I also spent a large amount of time this semester designing on Multisim and Ultiboard. I assisted Pedro in creating the schematics, and I took the lead in designing the 2nd PCB iteration. When our 2nd board came back, I found issues in the component sizes so the board could not be taken to WWW Electronics, Inc. to be assembled. And so I soldered the board in the NI lounge to test what was possible considering the errors. I then worked on the Ultiboard revision of the board with assistance from Pedro to create our 3rd and final PCB design. Between these two designs, the overall board layout remained the same, and only small variations were made in editing and updating component sizes to match what we ordered. I also worked on the Digikey searches and orders to ensure we had necessary components for the board and that the sizes correspond, primarily choosing 1815 and 2512 standardized sized parts. When this board arrived, I completed the hardware testing before and after it was assembled by WWW Electronics, Inc, focusing on the soldering and connectivity tests and power system checks. I additionally took responsibility in assisting Rohan in the case design. And I took the primary role in creating our video.

Pedro Rodriguez

My primary role during the semester was to design and construct the hardware. To start this, first each subsystem was designed in theory, establishing a general idea of what each subsystem is intended to do. This was followed by the selection of components that will be used,

followed by a port mapping the connections of each component with the help of Emory. Once this was done, I used the program Multisim to design the schematics for each subsystem in the device, as well as designing the layout of the components not covered in the database provided with us like the barrel jack, the three-pin connector to the rotary encoder, the buzzer, the Transient-voltage-suppressor, and the 3.3V and 5V regulators that were picked. Once the schematics were completed, I designed the first iteration of the printed circuit board using the program Ultiboard, and then I helped Rawan to design the third and final version of the PCB,

accounting for the components changed during the semester. My secondary role came as a supporting role in different tasks, like helping Rohan test the buzzer on the Virtual Bench and writing the script of our video presentation alongside Rowan.

Table of Contents

This should list the page of each of the major headings and subheadings below. An example is shown below.

Contents

Table of Figures

Table of Tables

Abstract

The Power Ranger's technical capstone project is called the LANGPAD, a linguistic system device that provides easy access to special characters from the Roman alphabet found outside of the English language. The project involves the use of a MSP430 programmed to use the expanded ASCII language, attached to an LCD touch-screen that displays the special characters available of its respective language, with the current options available being Spanish, French, and Greek. A PCB will work as the interface between the two components and the communication to the user's computers will be done through USB connection. The project was successful in achieving its proposed functionality and was even expanded beyond the proposed final product by adding a third language: Greek. This product is expected to improve the experience of writing in foreign languages, as well as other cases that require special characters for technical reports. A great deal of knowledge and experience was attained in the engineering design and construction process.

Background

In addition to leading to the closure of several small businesses, schools, and churches, the COVID-19 pandemic has also inspired an international shift to technology to allow us to stay more connected to one another, with platforms like Zoom, Skype, and Discord being the primary tools. However, this COVID-19 inspired shift to technology has also exposed the digital divide present between English speakers and non-English speakers, with the latter group being more likely to express an unfamiliarity with using digital tools and navigating digital environments [1]. This digital divide is due in part to the development of the QWERTY keyboard, which was done primarily with English speakers in mind [2]. Because of this, those who do not speak English, or who are not fluent, lack an easy way to access the diacritical marks needed to effectively communicate online.

The LANGPAD, a USB-connected touchpad, was conceived to address this issue. When plugged into the wall and connected to their user's computer, the LANGPAD provides the user with easy access to the special characters needed for their language. Additionally, the LANGPAD features an audio subsystem that emits a tone whenever a user selects a character, giving them an easy way to know that their touch input was successfully received. The volume of this tone can also be changed, should the user be typing in a public place or would simply rather not have the tone be present.

Current attempts to provide users with access to diacritical marks and other foreign characters on their laptops include providing character codes for each character and physical keyboards that support their preferred language [3]. While these choices do accomplish the end goal of letting users type in non-English languages, they each have their own set of disadvantages. The codes associated with each character are often complicated, with no pattern or structure to make the process of remembering them easier. Therefore, users consistently have to pause their writing in order to find these codes online to use them.

The keyboards that provide access to the user's language do serve as an easy way to type. However, these keyboards are physical, leading to portability issues should the user want to type somewhere other than their home. Furthermore, because each keyboard only serves for 1 language, the user would need multiple keyboards to support multiple languages.

The LANGPAD solves both of these issues, both by using an LCD Screen that can be programmed to support a multitude of languages in one device, and in its size, which is smaller than the average keyboard, allowing it to be easily transported from one place to another.

In order to develop this device, each member of our team used knowledge gained from their previous classes. Emory, Chris, and Rohan have multiple semesters of experience working with the TI suite of microcontrollers, which were featured extensively in the Embedded Computing and Robotics course sequence, consisting of ECE 3501 and ECE 3502. Therefore, Chris was able to use his experience to write code that could type characters over a USB connection and Emory was able to write software to interface with a touchscreen display using the SPI communication protocol. Emory used strategies he learned in CS 3240 to produce a user-friendly graphical user interface. Rohan and Chris were able to write the software to read the signals from the rotary encoder and send the appropriate audio signals to the buzzer. Rohan also had experience from his high school robotics team in CAD Design, which he used to design the enclosure for the LANGPAD. Chris's 3D printer was used to print the enclosure.

The printed circuit board was designed by Rawan and Pedro, who used their experience from the Fundamentals of Electrical Engineering series, ECE 2630, 2660, and 3750, to design the circuits in Multisim and lay out the boards in Ultiboard. They also ensured that the parts we needed were ordered correctly, and that any necessary documentation that 3W needed to manufacture the boards was provided on time and in the proper format.

Constraints

Design Constraints

ECE Capstone Constraints

Because our team included members studying Electrical Engineering, we were required to design our own custom printed circuit board for the device. Our team also included members studying Computer Engineering, so we were required to incorporate a professional-level microcontroller in our project.

Software Constraints

We were provided a license for NI Multisim and NI Ultiboard to design the PCB for our board design [4] [5]. We used Code Composer Studio as our development environment for embedded code because of its rich support for TI microcontrollers [6]. To create a Microsoft Windows Keyboard layout that supported typing special characters, we used Microsoft Keyboard Layout Creator [7].

Economic and Cost Constraints

Our team has been allotted $500 to create the project. While this budget was large enough to allow us to order the LCD that we wanted, we did not have enough funds to order a spare. Since we had room in the budget, we were able to order multiple microcontroller units to allow for parallel development of embedded software. We were also able to source some of our parts from the NI Lab, saving money for the rest of the budget.

Environmental Impact

The plastic used to print the LANGPAD's 3D-printed enclosure, PLA, is a bioplastic manufactured from fermented plant starch, usually from corn [8]. Also, PLA does not emit toxic fumes when incinerated. These properties have positioned the plastic as a viable replacement for traditional petroleum-based plastics. Additionally, our PCB and LCD are fully recyclable so long as they are deposited at a proper electronics waste handling facility as opposed to the typical recycling bin for paper and plastics [9][10]. However, it is worth noting that the manufacturing of such devices can have negative environmental impacts, such as wastewater contaminated with heavy metals and organic matter. Unfortunately, handling these environmental effects are beyond the scope of both our team and this project.

Sustainability

Our design is sustainable for a number of reasons, the first being that it does not make use of any rechargeable batteries, instead relying on wall power and laptop power. Because of this, we reduce the amount of waste that would otherwise be generated from the use of these batteries. Additionally, the LANGPAD can work with any language the user wishes without needing any additional hardware or power supplements. Finally, the enclosure for the LANGPAD can be built using any material, including ones more sustainable that the PLA plastic enclosure we currently have for it.

Health and Safety

Since we are relying on wall power to power the multi-language keyboard, we need to handle that power responsibly to avoid electrocution of users. To mitigate this risk, we secured all of our components inside a plastic enclosure that we designed. Since the enclosure is made out of a non-conductive material, this should reduce the likelihood of a short circuit and reduce fire hazards.

External Standards

To provide enough energy for the LCD display, our PCB design will contain a barrel jack that will allow us to connect our PCB with an outlet. The power plug used will work for both the NEMA 1-15 and the NEMA 5-15 standards popularized in the United States, that allows for a 120 Volt/15 Amp with a 60 Hz connection. However, only 5 Volts (peak) is going to be used. Since this is considered low voltage, and will plug to the typical outlet, no external utility standard will be needed to be addressed.

When the PCB will be designed, the standards designed by the Institute for Printed Circuits are going to be adhered to. More specifically, the IPC-2221A standard will determine the spacing in the parts of the board and tracing.

USB standards will define communications between the computer and the development board. USB communication was used to allow for the connection for the keyboard to write in the computer [11].

Communication between the LCD and the microcontroller will use a standard 4-line SPI interface. This interface is not a default standard for LCD communication, but it is the de facto standard for these types of configurations [12].

When it comes to encoding the characters for the multi-language touchpad, the keyboard extension will follow the Unicode Transformation Format UTF-8, which allows for backward compatibility with the American Standard Code for Information Interchange (ASCII), while expanding the range of vocabulary that we can provide. [13] [14]

Tools Employed

A variety of tools were used for the software and hardware elements of this project. They are separated by category below.

Software Tools

Code Composer Studio 10.4.0 was used as the Internal Development Environment (IDE) of choice for developing our software because it was developed by Texas Instruments, the same company that manufactured the microcontroller we used for the project [6]. As such, the IDE allowed us to quickly build, run, and debug our software as needed. GitHub was used to store all of our software in one place for easy access, and GitKraken was used as a task scheduler to keep us on track and update the professors and TAs on our progress throughout the semester [15] [16].

The C programming language was used to code all of the software needed for the project [17]. We also made extensive use of the USB Developer's package from TI to develop the USB interface between the LANGPAD and the user's laptop [18]. To program the LCD, we made use of the documentation provided by Newhaven Display International, the manufacturer of the LCD, as well as a library specifically coded for the MSP430 [19].

Hardware Tools

National Instruments' Multisim and Ultiboard were used to design, test, and layout the general configuration for our custom PCB [4] [5]. AutoDesk Inventor was used to design the 3D printed parts, while an Ender 3 printer was used to 3D print the parts [20] [21]. Testing of the PCB and LCD was done using the NI VirtualBench and the Analog Discovery 2. [22][23]

Ethical, Social, and Economic Concerns

Health and Safety

Since we are relying on wall power to power the multi-language keyboard, we need to handle that power responsibly to avoid electrocution of users. To mitigate this risk, we secured all of our components inside a plastic enclosure that we designed. Since the enclosure is made out of a non-conductive material, this should reduce the likelihood of a short circuit and reduce fire hazards.

Ethical Issues

From an ethical standpoint, it is important that our device does not raise any ethical concerns within the user when in use. There are two main ethical concerns that may arise from use of the LANGPAD. The first concern is related to the source of the parts on the device. The user should not have to question whether the device was produced with ethically produced parts and labor [24]. Secondly, an ethical concern can arise to the additional power used by the device. The LANGPAD should try to conserve energy, so as to not be a particularly power-hungry burden on the environment.

From a social standpoint, it is important that the LANGPAD be accessible and usable to people of diverse backgrounds. There should be no high barrier that prevents particular social or economic groups from taking advantage of the LANGPADs features [25]. Thus, the user interface of the device should take into consideration these factors to make it as accessible as possible.

From an economic standpoint, the total initial cost of the device should not exceed a reasonable amount. Following the prototype stage, significant efforts should be made to reduce the overall cost of the device while keeping in mind the various ethical and social concerns. Particularly costly parts, such as the LCD should be bought in bulk to reduce the overall cost [26]. Efforts should also be made to combine the microcontroller and PCB to reduce costs as well. The casing should be designed in a way that it can be injection-molded, so that the cost per unit is decreased significantly [26]. These various changes would help drive the overall cost of the device down to reach a larger user base [26].

Intellectual Property Issues

This project has a high potential for being patented because no similar invention could be found that worked as a keyboard extension that focuses on expanding the characters available for typing. The patent “Virtual Keyboard Input for International Languages” [27] was the closest patent found, but there are many key differences. As stated on its description:

“In some international contexts, the keys of the graphical keyboard may be associated with characters in an alphabet of a first language (e.g., English). The user may select one or more keys of the graphical keyboard to enter a string of characters that represents a word and/or character included in a second language (e.g., Chinese, Korean, Japanese, etc.). Using word prediction, auto-correction, and/or suggestion techniques may speed up text entry and reduce spelling mistakes”

Looking at the description, this patent for starters is not an actual physical device, but instead a completely digital device. Furthermore, what this device is trying to accomplish is to help translate letters to other languages, then to provide the characters of those languages for writing.

[28] "Multilingual key input apparatus and method thereof" also has the objective of providing users for an easier experience when inputting special characters. However, this patent differs from ours since it focuses on mobile devices keyboards, instead of computers keyboards, and its main focus is to improve the experience of inputting characters on the mechanical keyboard, with the special characters as a bonus, instead of being the main focus. A last difference would be that this patent works on modifying the keyboard, instead of being an expansion of it.

Detailed Technical Description of Project

LANGPAD is a standalone touchpad device capable of typing special characters commonly used on foreign languages. It connects to a computer using a USB cable and acts as a keyboard input device. The touchscreen display consists of pages of buttons labeled with specific special characters. Tapping on one of these buttons will type the corresponding character on the connected computer. There are tabs along the top of the display to allow the user to quickly toggle between languages.

Figure 1: LANGPAD User Interface Mockup

Our device will trigger key presses on Microsoft Windows computers via a USB connection. A microcontroller will be programmed to register inputs from the touch screen and determine which character a user has pressed. The microcontroller will then communicate this information to the computer using the standardized USB HID protocol, causing the computer to type the corresponding character [11].

Hardware

The hardware consists of one central Printed Circuit Board, that works as an intermediate between an LCD screen and an MSP430 microcontroller. The device is powered through a barrel jack, and it communicates with the computer through USB communication. Furthermore, an audio sub-system was developed to provide feedback for the user while using the product. The development of the subsystems will be described in the following sections.

Schematics

The goal of the power system was to provide just the enough power required of the MSP430 and the LCD screen. The team took much time evaluating whether to use an LCD that could be powered through USB alone, or one that should be powered through wall connection. Since the LANGPAD is an add-on for the keyboard, the design needed to be kept as accessible as possible. However, given the NHD-5.0-800480FT-CTXL-CTP, LCD touchscreen advantages when programming, the team decided to compromise and pick this LCD touchscreen despite the amount of power needed [19]. Once the decision was taken, two voltage regulators, of 5 volts

and 3.3 volts, to provide just the enough power required to feed both the MSP430 and the LCD screen.

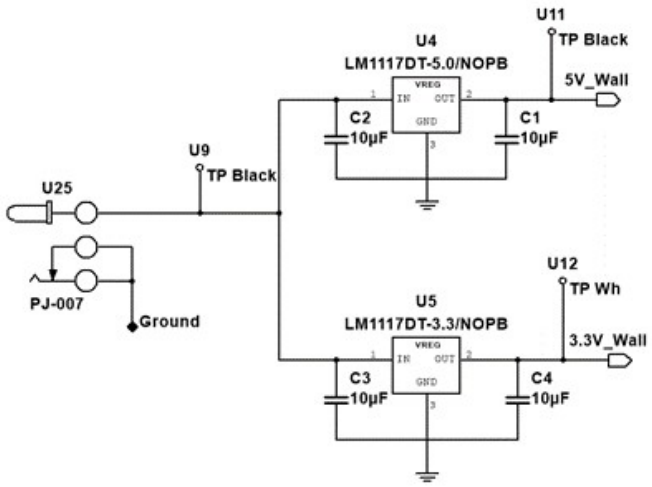


Figure 2: Circuit Power Supply Schematic

An audio subsystem as agreed upon to provide an intuitive user experience. The idea is to add a piezo buzzer to the PCB that will provide audio feedback to the user whenever he/she presses any of the keys on the touchscreen. The buzzer is accompanied with a digital 2-bit rotary encoder that will adjust the audio feedback depending on the frequency set up to. From left to right, the pins of the rotary encoder are output A, ground, and output B. When a voltage is applied to the output pins and the shaft of the device is turned, the rotary encoder generates two square waves that are 90 degrees out of phase. By looking to see which output, A or B, is ahead, we can determine the direction the device is turning, as well as read each of the four possible states: 00, 01, 10, and 11.

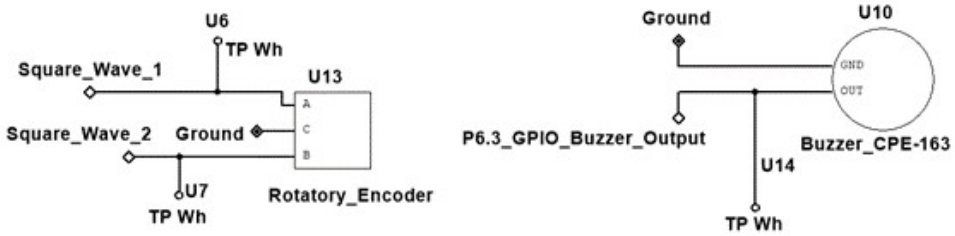


Figure 3: Audio Subsystem Schematic

For the processor, instead of just ordering an MSP430 CPU, the development kit of the MSP-EXP430F5529LP was ordered [29]. This decision came mostly as a way to save time, and if the project were to be produced for high quantities, just the CPU would be ordered. The MSP430 works as the main way to communicate between both the LCD and the audio Subsystem, and it is also powered through the 3.3 voltage regulator.

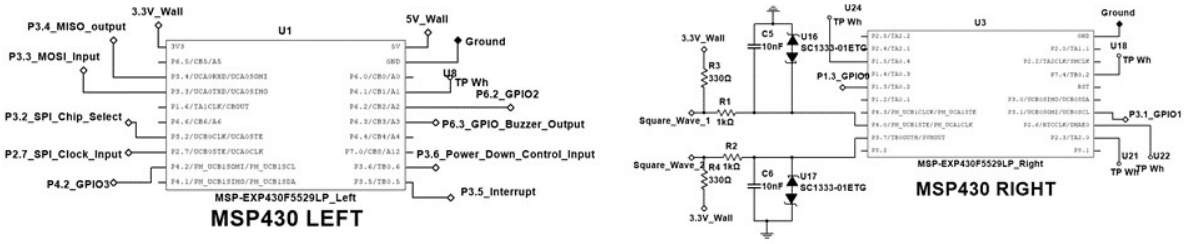


Figure 4: MSP Schematic

The PCB itself is going to have a male 20 pin header, and the LCD touchscreen is connected through a ribbon cable. This decision was made so that we could test all the other components while having the LCD connected, otherwise, the LCD would have overshadowed all the components in the PCB.

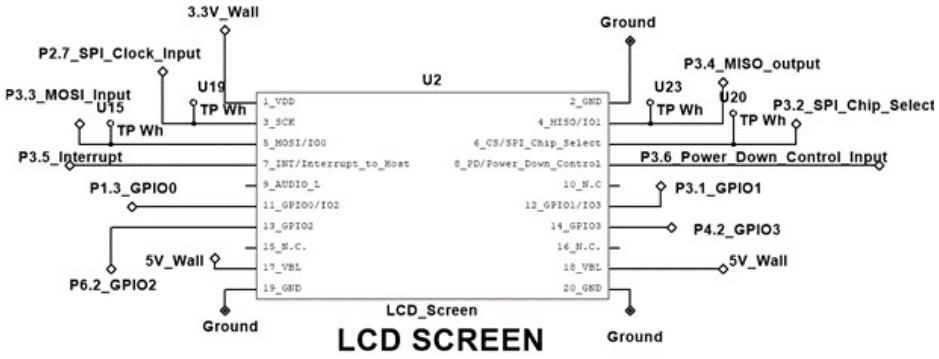


Figure 5: LCD Schematic

The PCB was designed to attach to the MSP430 Launchpad header pins from the bottom through two 20-pin female pin headers, while connecting the LCD touchscreen at the top through one 20-pin male header and a 20-pin ribbon cable. Figure 6 shows the complete layout of the and trace routing. The board was designed using only two layers (top and bottom) in order to simplify manufacturing and decrease production costs.

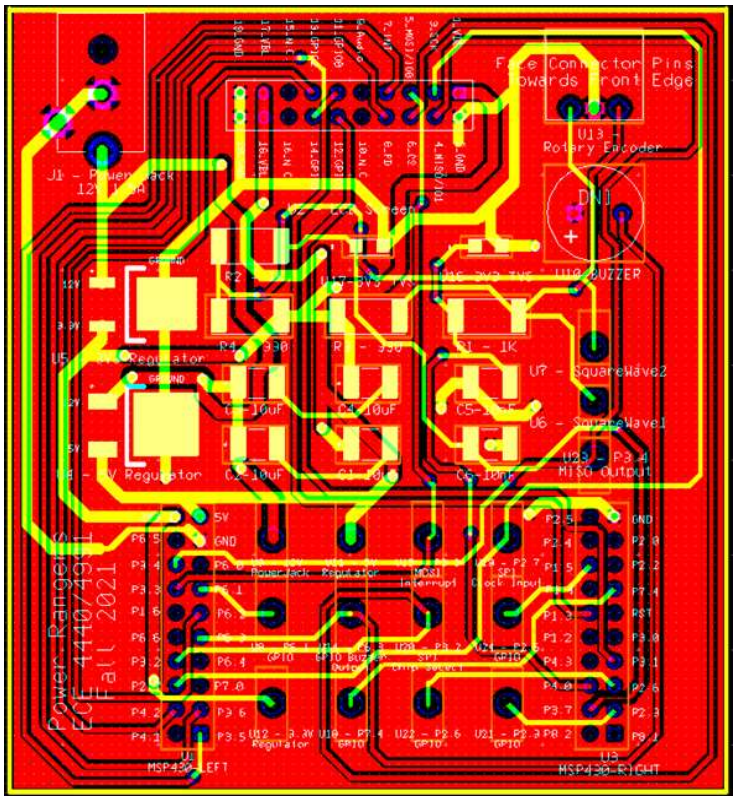


Figure 6: PCB Layout

Components Used

To visualize the system, the project was divided into hardware components used and software used. The hardware components are the following:

1. For the power we have our barrel jack [30] that will input a DC voltage of 20 Volts. To avoid burning the main components connecting to the PCB, two voltage regulators, one 5Volts [31] and one 3.3Volts regulators [32] parallel to each other, were used. With these regulators, a total of four 10nF Capacitors [33] (two for each regulator), were placed as a safe mechanism.

2. We have the MSP430 [29] itself that is connected through the PCB via two 20 pin female header connectors [34]. Accompanying the MSP, there are two 3.3Volts Transient-Voltage-Suppressors (TVS) [35] that will protect the connections between the MSP and our rotary encoder [36] to avoid any unwanted voltage pike coming into the MSP. Along with the TVS, there are a total of 2: 330 ohm and 1k ohm resistors [37][38] and another 2 10nF Capacitors (one of each TVS) are placed to help the TVS prevent the voltage pike.

3. The rotary encoder is connected through a 3 head-pin header [39]. The rotary encoder interacts with a buzzer [40] to provide the audio feedback when pressing a key on the LANGPAD.

4. To connect the LCD [19], and avoid any intrusion while connecting and testing the PCB, the LCD is connected through a 20-pin male header [41] and a 20-pin ribbon cable [42].

5. Last but not least, a total of 17 test points [43] were placed to test each one of the main components of the LANPAD.

Software

A high-level diagram of the software running on our device is shown in the figure below. Our software is bare-metal and written in the C programming language [44]. The software architecture consists of two main stages, a Setup stage and a Normal Operation stage. Our program begins in the Setup stage, which is responsible for setting up all of the necessary configurations for our device. This includes establishing a USB connection with the connected computer, setting up the LCD, and configuring necessary pins for our audio system.

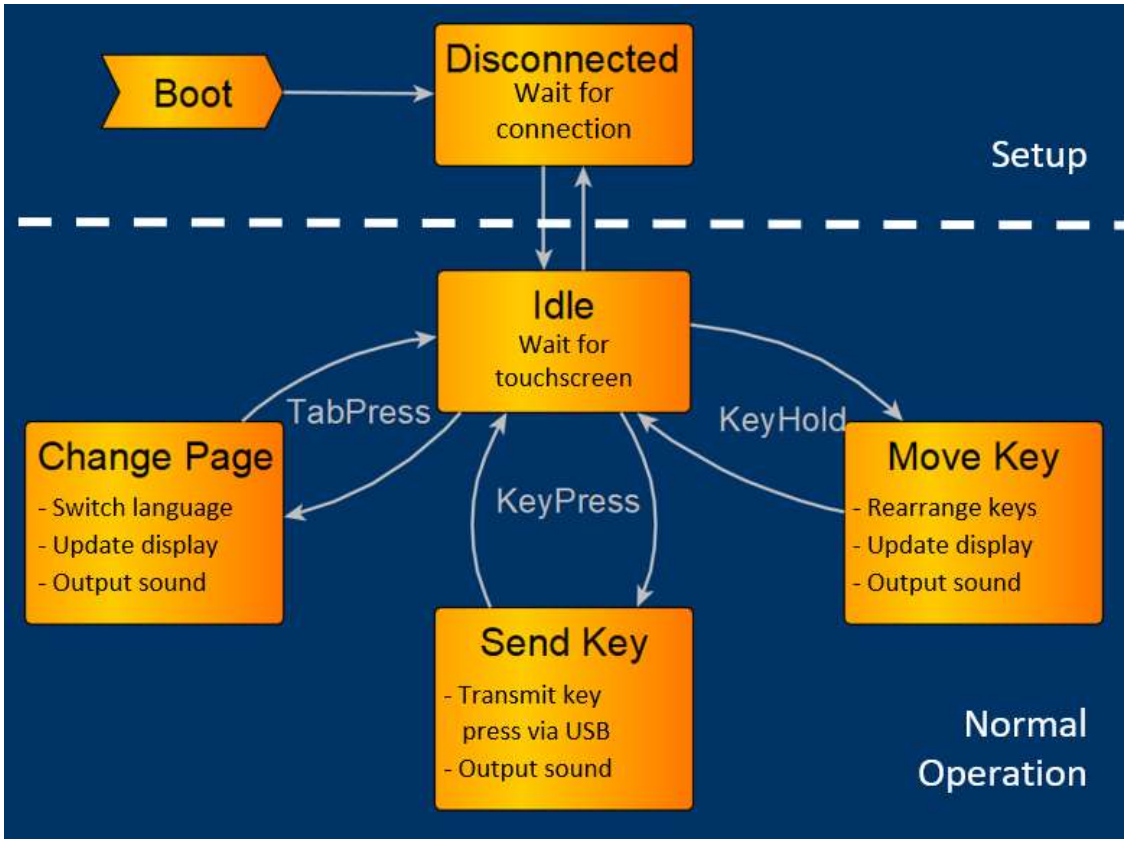


Figure 7: LCD Software Block Diagram

After the Setup has been executed, the software will then transition into the Normal Operation stage, where it should remain as long as there are no errors in the USB connection such as a physical disconnection. The program remains idle until inputs are detected from the touchscreen. When input is detected from the touchscreen, the software will move to the corresponding state, execute the necessary code to handle the touch input, and then return to the idle state. While the code is not in the idle state, it will not be able to handle new inputs from the user until execution returns to idle.

USB Communication

In order for the device to trigger key presses on a connected computer, we had to program our device to communicate via USB. Because we chose the MSP4305529 Launchpad, we were able to use the MSP430 Developer's Package [29] [18]. We used these libraries to configure our device to operate as an HID device. HID stands for Human Interface Device, and it is a standard USB protocol used by common peripherals such as keyboards [11]. HID consists of transmitting structured packets over the USB bus to the target computer [45]. After the USB connection is established in our setup phase, our device can activate key presses on the target computer by sending a properly formatted HID packet.

The way that key presses are implemented in the HID protocol is that the input device transmits 7-bit character codes over the USB connection [45]. The operating system of the target device is then responsible for translating these 7-bit codes into a corresponding character and typing that character. The way that Microsoft Windows translates the character codes into characters is through a keyboard layout [46]. Each key position is associated with a corresponding character code, and the operating system will translate an incoming character code according to the mapping that is associated with a given character code. The standard United States keyboard layout is shown below in Figure 8 [7].

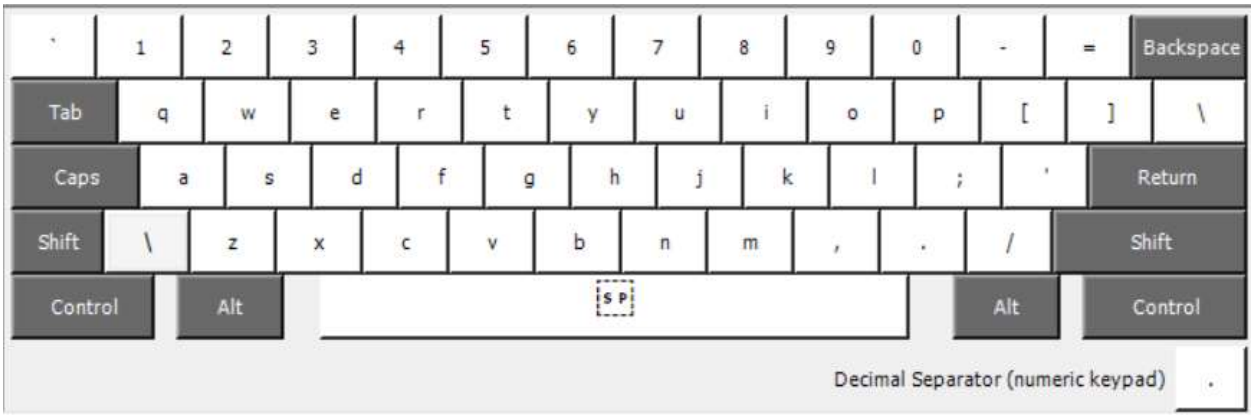


Figure 8: Standard Windows Keyboard Layout

While the standard HID code that we implemented allows us to use standard keys that are available on the US standard keyboard layout, it is not capable of sending the special characters that we need for LANGPAD's core functionality. This is because there are no mappings associated with these special characters in the default United States layout. To address this issue, we created our own keyboard layout based on the standard United States layout that included mappings for our required special characters using Microsoft Keyboard Layout Creator [7]. To allow for these extra mappings, the layout uses the right "Alt" key as a modifier to access these keys. The "Alt" mappings we have made are shown below in Figure 9 [7].

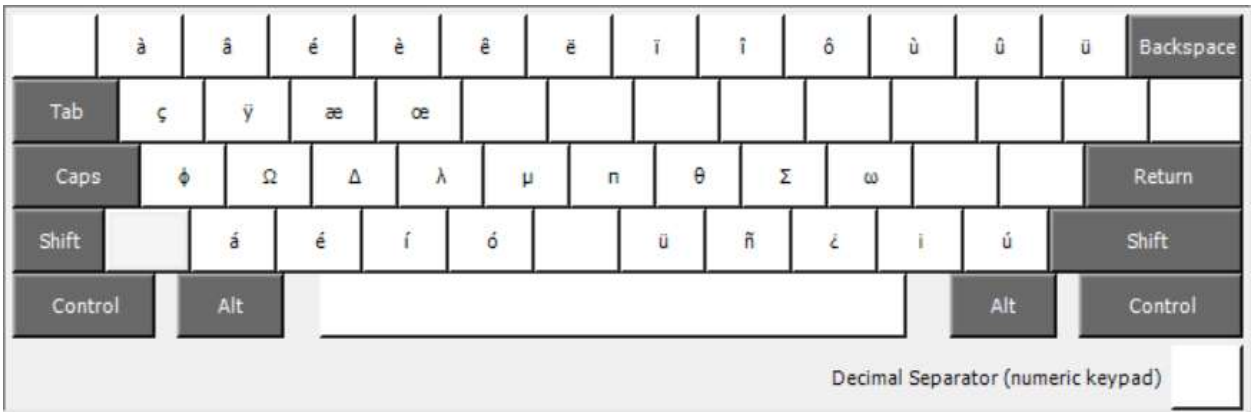


Figure 9: Special Character Keyboard Mapping

In order for our device to type these special characters using our keyboard layout, LANGPAD triggers 4 HID instructions for a single key press. The sequence we use is a press of the right "Alt" key, a press of the character code associated with the special character that should be typed, a release of the character code of the special character, and a release of the right "Alt" key. Since the transmission of HID packets takes a very small amount of time, this executes quickly enough to not result in noticeable latency from a user.

The mappings for the character codes are stored in globally accessible arrays. Whenever a button is pressed, the software checks the mappings at the index of the key that was pressed on the touchscreen and uses that information to handle the key press accordingly. In order to implement the rearrangement of keys on the display, mapping arrays are modified each time the user moves a key on the screen. Every time that these mappings are modified, they are stored in a flash memory bank. Storage of the mappings in flash memory is used to retain the ordering when the device is unplugged. In order to use the stored memory, it is read in the startup sequence, validated, and then used to generate the mapping array. If the memory data that is read is invalid, a default mapping is loaded instead of the one stored in memory.

Touchscreen

To provide for a modular and seamless experience for the user, we opted to use an LCD touchscreen for our input device. This device was chosen because it allows for a similar experience to a typical keyboard while still allowing the user to choose different languages. This system allows for different languages, as the screen can be customized to support different characters. There are several different layers to how the touchscreen works and they are described in the next section.

The lowest level of the touchscreen is the communication over SPI between the screen and the microcontroller. The two communicate over standard SPI, consisting of: a MISO, a MOSI, a select line, and a clock [12]. This system is how we send and receive commands to the LCD. It is important that we also be able to read from the LCD over SPI, as we will need to constantly be pinging the LCD to check for a screen press. The system is able to send over a byte at a time, which we can combine with the select pin to build out detailed commands. This system allows us to start up the LCD using the given list of startup commands, as well as build a shell on top of this system containing all the drawing commands we need.

The second level of the touchscreen is the interface to send commands. Using the base SPI communication of a singular byte being sent, we broke out the commands into a few different structures. We used commands for reading and writing one byte, two bytes and four bytes all in one function. These commands were created using the fact that the system expects the select bit to be toggled based on how long the system is reading or writing.

Finally, the highest level of the touch screen programming is the graphics commands set. The programmers guide provides a number of built-in graphics routines that are accessible by sending commands. We used a pre-made library that had many of these graphics functions already written out [47]. Using some of these routines, such as draw line, draw rectangle, and draw text, we compiled even higher-level functions such as: draw rectangle with text and draw rectangle with touch enabled. By combining many of these functions we were able to build out a modular graphics display for all of the character pages.

One special functionality of the LCD is the ability to register when the user touches the screen. In order to achieve this functionality, every "block" on the screen gets assigned an ID tag. For example, if the character "a" were in the first position on the display, it would get assigned an ID tag of 1. In order to check for when and where a touch happens, we constantly read a register that reports if there has been a press on the screen. Once this press is confirmed, we check the register on the LCD that contains the ID tag of the pressed block. Using this information, we can look up which character is in that location and send the appropriate character to the computer.

The LCD by default has a custom font layout loaded in that contains all the basic alpha-numeric characters. This font does not contain any special characters, however. In order to display special characters on the screen, we had to import a custom font into the ROM of the LCD. This process involved selecting characters that we needed, and loading them into the bitmap form usable by the LCD. Once we imported the custom font, we were able to access the special characters using their index in the ROM of the LCD and display them correctly on the grid of characters.

There were a few considerations made when choosing how to layout the user interface. Firstly, there was an intention to mimic the look of a standard keyboard to ensure the user would be familiar with the device. Secondly, we wanted to allow for three main languages, so a tab system as we have implemented makes the most sense. Additionally, we wanted to allow the user to control and view the current volume in case they are working in a quiet area, so we display that information at all times.

Audio

In order to provide the user a way to change the volume of each button press, we needed to be able to read the input signals from a 2-bit rotary encoder. Because this is a digital rotary encoder, the various states of the device are represented as DC Signals, with a 0 being represented as 0 volts and a 1 being represented as 3.3V [48]. Because this is a 2-bit encoder, we knew that the device would be in one of 4 discrete states at a time. To determine the order in which the states appeared, we used a VirtualBench to read the signals as we turned the shaft. We found the order of the states to be 11, 01, 00, then 10. An example of how the 00 state looks on the VirtualBench is shown below [22].

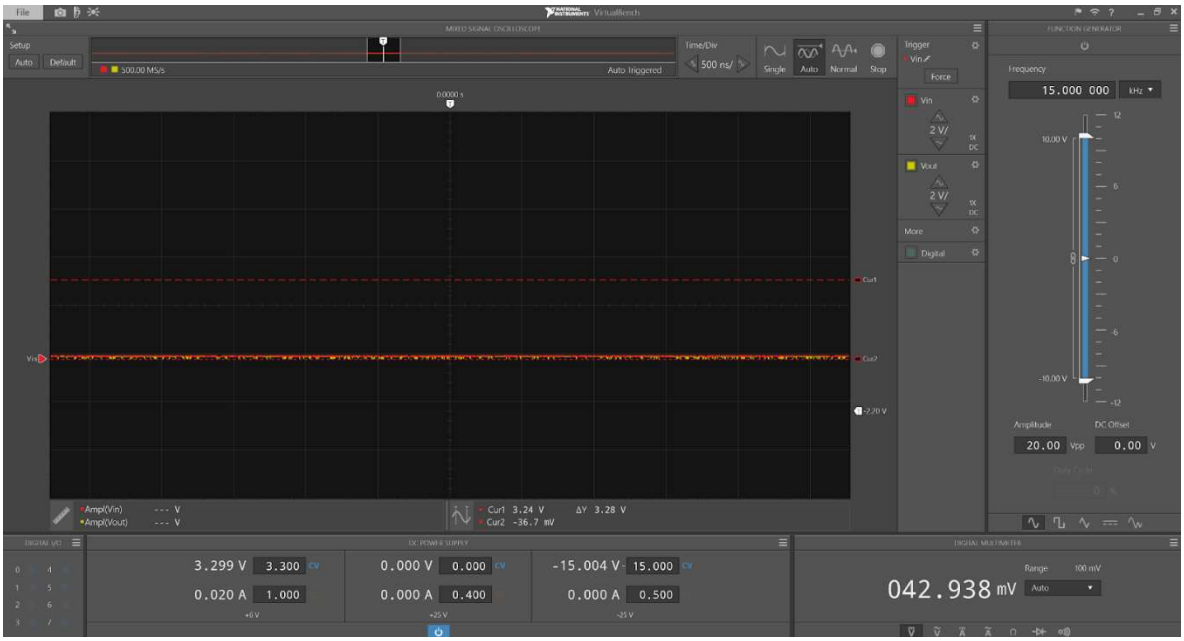


Figure 10: 00 Reading from the Rotary Encoder

As one can see, both states are represented by DC signals at 0V. With this information in hand, we were able to begin writing the software to read the two rotary encoder outputs. Currently, whenever a user presses a button on the LCD, the algorithm will check the current state of the rotary encoder via GPIO pins: 3.7 and 4.0. Using an in-conditional statement, the algorithm checks the state against four possible volume options, and then accordingly sets the appropriate volume.

The original plan to vary the volume of the tones from the buzzer was to set the amplitude of a PWM signal in response to a state of the rotary encoder. However, when we discovered that the MSP only emitted PWM signals at a set amplitude, we needed to find another way to vary the signal. Through some experimentation and consultation of the MSP430 Driver Library, we found that at very low frequencies, changing the duty cycle of a PWM signal affects how loud or quiet it can be. We configured Pin 2.0 as the output pin of choice because it was connected to Timer A, and using the methods provided by the driver library, we were able to initialize the timer as an SMCLK with a period and duty cycle of our choosing. The SMCLK runs at 8MHz. When divided by 32, this yields the actual rate of the clock, 250kHz. We then set the timer period to be 1000 ticks on the clock, yielding a period of 0.004 seconds or a frequency of 250Hz. We experimented with different duty cycles ranging from 0.1% of the tick period to 1% of the tick period (1 through 10). We found that a good range of volume was represented by using duty cycles of 0.1%, 0.4%, 0.6%, and 0.9% of the tick period.

Finally, to show the user that the volume was changing properly, we implemented a volume bar on the screen itself. Turning the rotary encoder fills the volume bar, and thus gives the user a visual way of ensuring that the volume is changing. We also decided to have the volume roll over every 4 states, rather than give 8 separate volumes for the user.

3D Printed Enclosure

The CAD Model for the 3D printed enclosure was designed in Autodesk Inventor, and can be seen in the figure below [20]. The top piece housing the LCD and buzzer, the walls containing a hole for the cables and rotary encoder, and the bottom plate for the MSP and PCB were all printed separately to avoid warping.

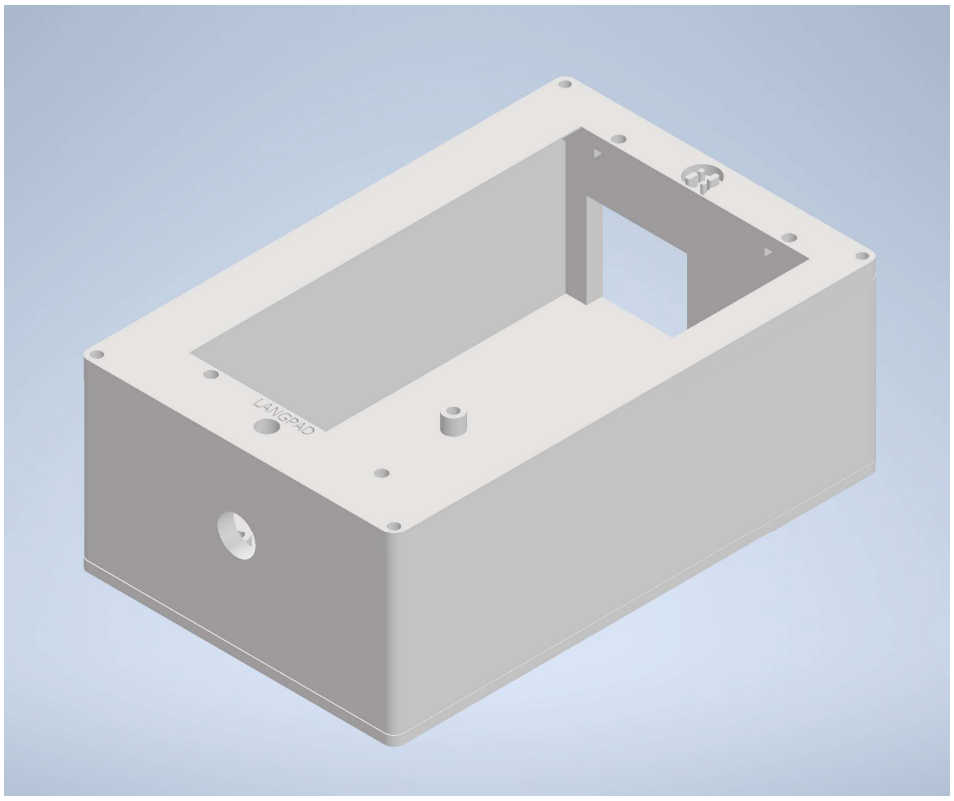


Figure 11: CAD Model of LANGPAD Enclosure

Design Decisions and Tradeoffs

Size

A decision pretty early on the project timeline was made to make this board pretty small. This decision came from the very nature of the project. Since the intent of this project is to create a product that will serve as an add-on/extension, the team believed that reducing the board size will make its transportation and utilization easier. As an outline for the board, given that the MSP and the LCD were going to be both above and below the PCB respectively, we decided to make the maximum possible size of the board equal the size of the LCD (109.00mm W x 65.80mm H), since this was the biggest component interfacing the board [19]. The board's size ended up being 75.5mm x 82.8mm long [29].

USB vs Bluetooth

Another design decision we had to make was between using bluetooth for character entry vs using a usb connection. The decision was made to use the USB connection as it required much less complexity to our device. If we were to design it around bluetooth, we would need an additional module, and the bluetooth was not determined to provide enough of an additional advantage to be worth such complexity. Additionally, Chris had experience with using HID device libraries that the MSP supported, so it was also more familiar to us. Thus, we decided to opt for USB connectivity.

External power source

For starters, the team originally designed a power system that relied on a 20 Volt, 12 Volt, and 5 Volt Voltage regulators, expecting to provide the necessary amount of power into the system. However, the team quickly came to realize that the 12 volts regulator was not going to be necessary, since 20 volts was enough to feed the LCD touchscreen we had at the moment, and feeding the MSP430 either 12 volts would risk to burn the device, and the MSP430 could work properly with less than 5 volts feedback. So, the design was modified to account only for two voltage regulators (20 volts for the LCD, and 3.3 volts for the MSP430). This design was, once again, modified when a different LCD was chosen. With this new LCD, the idea of having the laptop feed all the device was entertained, but because USB connection can only provide 5 volts, there was no way that both the MSP and the LCD were going to function simultaneously, returning to the initial plan of wall connection, now with two voltage regulators (5 volts regulator for the LCD, and the 3.3 volts regulator for the MSP430).

Microcontroller choice

For our microcontroller, we have decided to select the TI MSP430F5529LP [29]. We wanted to work with a TI MSP microcontroller because they are professional-grade devices that we have experience with in the Embedded Computing & Robotics curriculum. To further refine our selection, we looked for devices that had developer support examples of USB HID devices. We found that the MSP430F5529LP had support for this functionality.

Our main reason for choosing this microcontroller over others was that the MSP430 is well-documented and has a mature support library for USB device development. The MSP430 USB Developers Package provides a lot of support tools and documentation for use with the MSP430 variant that we have selected [18].

Another consideration when choosing a microcontroller was the size of the device itself. Our MSP430 selection is 4.4 inches by 3.2 inches. These dimensions were acceptable because they were similar to the size of our LCD and they will not require creating a larger housing for the device [29].

Our major concern about using the MSP430 is that it would not have enough pins to interface with the display. The display we originally selected was interfaced via a 40 pin and a 6-pin connector. The MSP430 only has 36 port pins available, so we planned to connect all 24 color pins on a single GPIO pin to have color options of black (all 1's) and white (all 0's) [29]. We felt that losing the color functionality was a necessary compromise over boards with more available GPIO pins because it is not essential to our device. We ended up changing our LCD

selection to one that used less GPIO pins, so this particular design concern did not end up being an issue.

Processing speed was also a consideration of our microcontroller selection. The MSP430 that we chose has a single core processor that is configurable up to 25MHz [29]. While there are many faster offerings, we did not anticipate that our device would need this much computing power. We ended up operating our device at 8MHz in the final product, our device was able to support our desired functions with plenty of overhead.

Our choice of development board was reasonably inexpensive at a price of around $20. Since the MSP430 is cheaper than other offerings we found, it gave us more room in our budget allowing us to buy multiple development boards to facilitate independent work on embedded programming. We can buy a development board for each team member and we will still be well under our budget.

LCD vs. physical buttons

When initially planning out our device, we were trying to decide the user input method for the device. Our two considerations were a touchscreen interface or a physical keyboard interface. Our reasons for considering an array of physical keys were that it would likely be easier and more affordable to implement. This would also allow us to design a device that runs completely off of USB power because power requirements for physical switches are very low. We ultimately ended up deciding against this because we felt like the extra flexibility offered by the display outweighed the drawbacks. We wanted to design a device that could be expanded to support many different languages and this really wasn't possible with a physical interface. Despite the increased development difficulty and the increased power requirements, we felt like this was the better option for our device's interface.

Audio Subsystem

For the audio subsystem, the team's advisor pointed out that due to the analog nature of the rotary encoder, there was a high risk of voltage spikes breaking the MSP during usage. To fix this, the team's advisor recommended applying a Transient-Voltage-Suppression diode between the pins of the MSP and rotary encoder for safe use. This protection system went through different iterations until calculating the proper resistors to accompany the TVS. Also in regards to the rotary encoder, we chose a rotary encoder with a 90-degree shaft because we wanted the shaft to stick out of the side of the LANGPAD, as opposed to the top, which may have led to an awkward typing experience for the user. The one downside to this was that the rotary encoder chosen used solder lug terminals as opposed to PCB friendly terminals. To fix this, we decided to solder 3 wires to the ends of the rotary encoder. The other ends of the wires would lead into a 3-pin header, which would be soldered to our PCB. This decision turned out to be better for the design, as we had the freedom to place the rotary encoder anywhere we wanted on our enclosure. Finally, the team's advisor recommended using a piezo buzzer because of its simplicity and ease of use. Furthermore, the recommended buzzer had a low enough minimum voltage threshold that additional audio amplification circuitry, such as an inverting operational amplifier, would not be needed.

3D Printed Casing

When designing the 3D printed casing, we wanted something that was small, easy to carry, and sturdy enough to protect all of the components inside of it. After making some preliminary sketches, we drafted our first design, seen below.

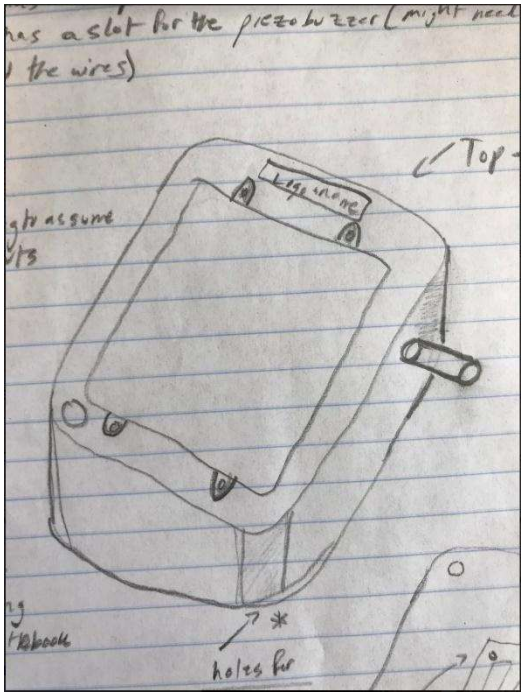


Figure 12: Preliminary LANGPAD Enclosure Sketch

This design features a rectangular body. We have a hole in the side to house the shaft of the rotary encoder and a hole in the bottom left for the buzzer. The MSP and PCB would be mounted via screws on the bottom plate, and the LCD would be attached via screws on the top plate. We also added a logo at the top. Finally, we decided to print the top as one piece, and the walls and the bottom as another piece. However, when we went to print, we ran into some major issues. The design was too big for the devices, leading to a lot of unnecessary space that we knew would affect the device's portability. Additionally, because we printed the bottom and walls as one piece, the print experienced major warping. We also noted that the positioning of the rotary encoder would be awkward for left-handed users. Furthermore, the screw holes were too wide for the screws to thread in, and the LCD was not fitting properly in the top plate. Finally, we forgot to include a cutout for the USB and Barrel Jack cables.

To account for these issues, we instituted a number of changes. First, we reduced the overall size of the print and printed the top plate, walls, and bottom plate as three individual pieces. In doing so, we would reduce the print time of each piece and prevent the warping we saw earlier. Secondly, we moved the rotary encoder to the bottom of the device for easy access and added a cutout for the wires at the top. Finally, we reworked the dimensions of each piece so that the LCD would slide neatly into the top plate and the screw holes would thread into the plastic. For securing the LCD and MSP, we used screws and nuts. Once we instituted these changes, the housing was complete.

Device Platform

We decided to design our product to work with Microsoft Windows computers [46]. Because the default English language configuration for windows does not natively support the use of the special characters that our device types, we had to create our own custom keyboard layout that added support for these characters.

External Power Source

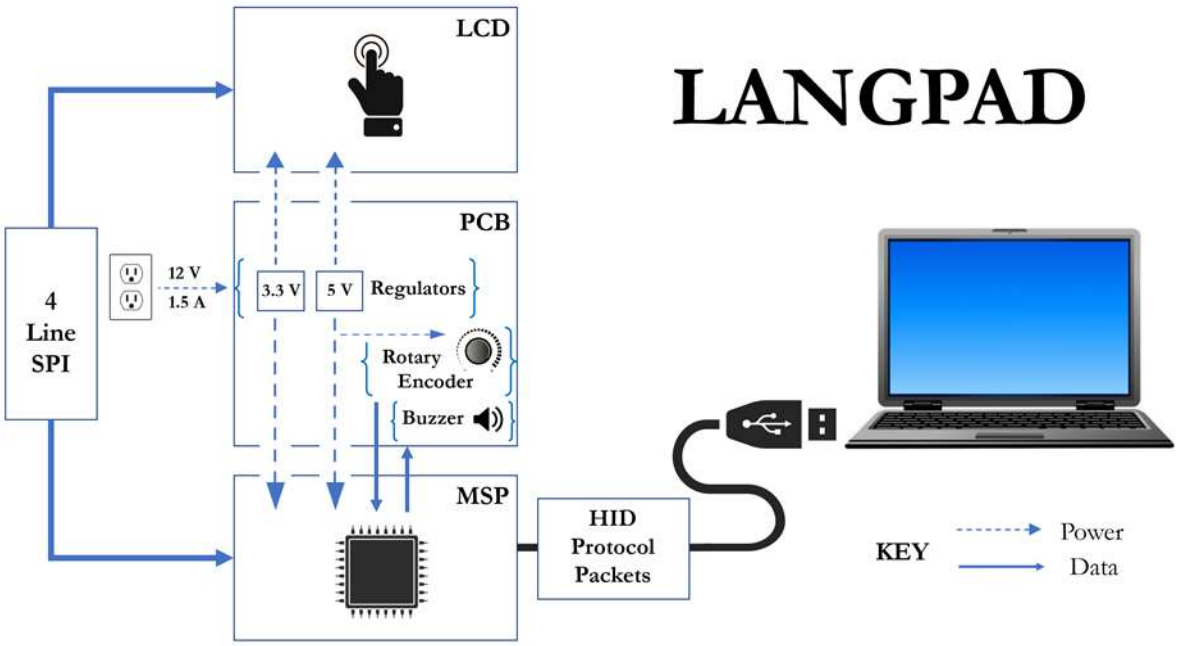


Figure 13: Original Block Diagram

Figure 1 depicts the original block diagram for the LANGPAD. There are three main components: the LCD, the PCB, and the MSP. Through the LCD, users are able to see the language that they are on, and their respective symbols as well. This is the device users' interface with to select characters to be transferred to their computer. The MSP controls and processes the operations like turning on the LCD and the USB communication. The MSP and LCD communicate through a 4-line serial peripheral interface. And the MSP sends data to the computer using human interface device protocol packets. The PCB acts as a connection interface between the MSP and LCD simply through headers. The PCB also houses two main subsystems: power and audio. The audio subsystem provides sound feedback that a key has been successfully processed when typing through the LANGPAD. Power was intended to be provided to the LCD and MSP from a wall power source; a transformer delivers 12 V and 1.5 A to the board, and two voltage regulators step these down to the appropriate voltages of 3.3 V and 5V. The ground from the wall transformer is also used to ground the rotary encoder.

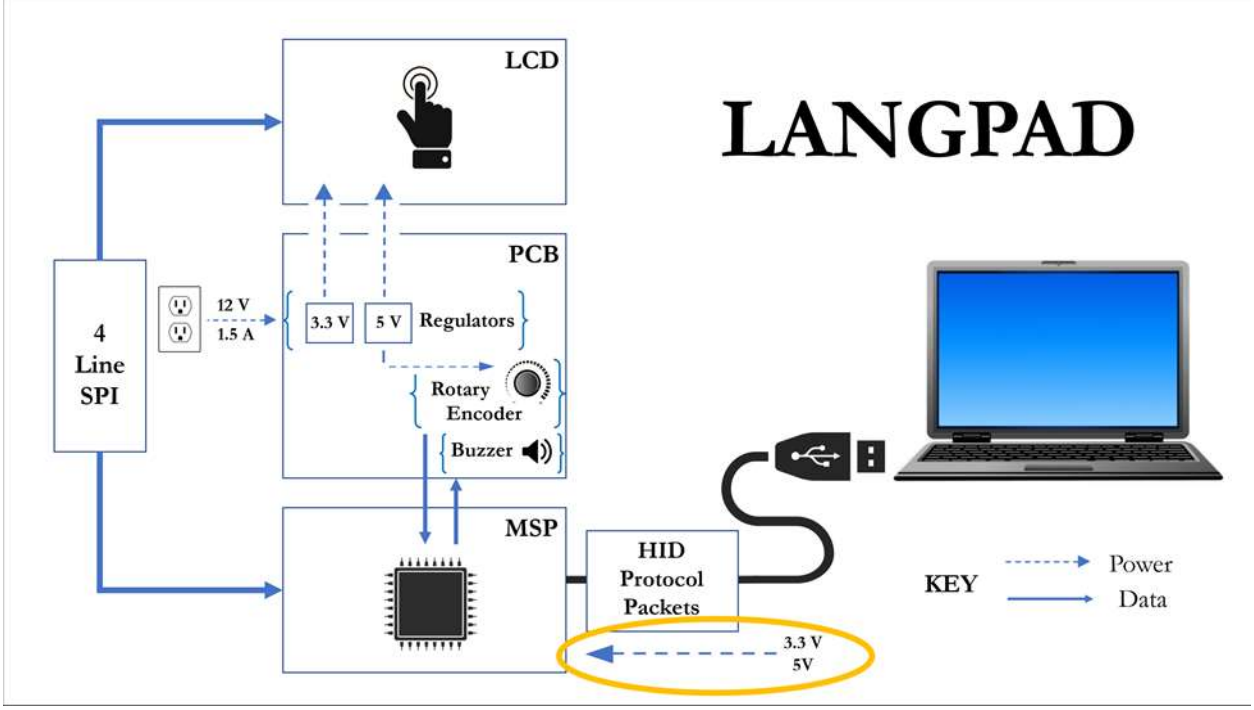


Figure 14: Revised Block Diagram

When testing our board, we came across an issue in which the PCB did not provide the correct power supply to the MSP. This was corrected by changing the MSP configuration so that the MSP launchpad is powered by the laptop. This change is reflected in Figure 2.

Problems and Design Modifications

PCB

The first problem found with our design was on the PCB itself. Once it was finally printed out, we quickly realized that the layout size of most components did not match those of the actual component. Furthermore, while verifying the PCB for the third final iteration, we realize that the voltage regulators were traced incorrectly, with the output traced to ground, and the ground traced to the output. Luckily, we were able to fix this issue without having any component burn.

LCD

When first considering which LCD display, we were going to use, we did not consider deeply enough the usability of the component we were purchasing. Upon further investigation into the user guide for the first LCD we chose, we found a lack of detailed programming information [49]. It appeared that the controller on the LCD was not very powerful, and many of the graphics designs would have to be built completely from scratch [49]. This fact meant that we would likely end up spending a more than significant amount of time working to develop end to end graphics libraries, which is not the main focus of this project. Thus, we decided to rectify this problem by choosing an LCD with a more powerful graphics controller that had more

documentation and instructions [19]. This change was one we considered thoughtfully, as the new LCD chosen increased our total cost by approximately $40 [19]. However, after much consideration we decided to make the switch and we believe we saved much time by purchasing the LCD with the more powerful graphics controller [19].

Ordering Parts

Over the course of the semester there was a pressure to order parts quickly, as the supply chain for electronics currently is not consistent. This fact urged us to quickly decide on parts and order them, causing us to make mistakes occasionally in our haste. This resulted in us ordering some incorrect components, as we chose the correct part values but wrong size. This looming supply problem affected us throughout our part selection and ordering process.

Power Delivery Change

In our original design, the MSP430 and the LCD touchscreen were both powered by our PCB's power system. However, when we tried to integrate our touchscreen interface with the code that typed out USB characters, the USB software no longer functioned properly. While troubleshooting the error, we found that the integrated code worked when the hardware was powered by an external power supply, instead of the one we designed on our circuit board. We believe that power from our PCB was not satisfactory for USB communication with the computer while supplying load to the LCD. In order to get around this issue, we decided to disconnect the power system from our MSP430 and instead power it via the USB connection. This was achieved by clipping off the 5V and 3.3V pins from our microcontroller and reconnecting jumper pins for the USB power system. By making this change, the integrated software was able to operate reliably while connected to our manufactured PCB.

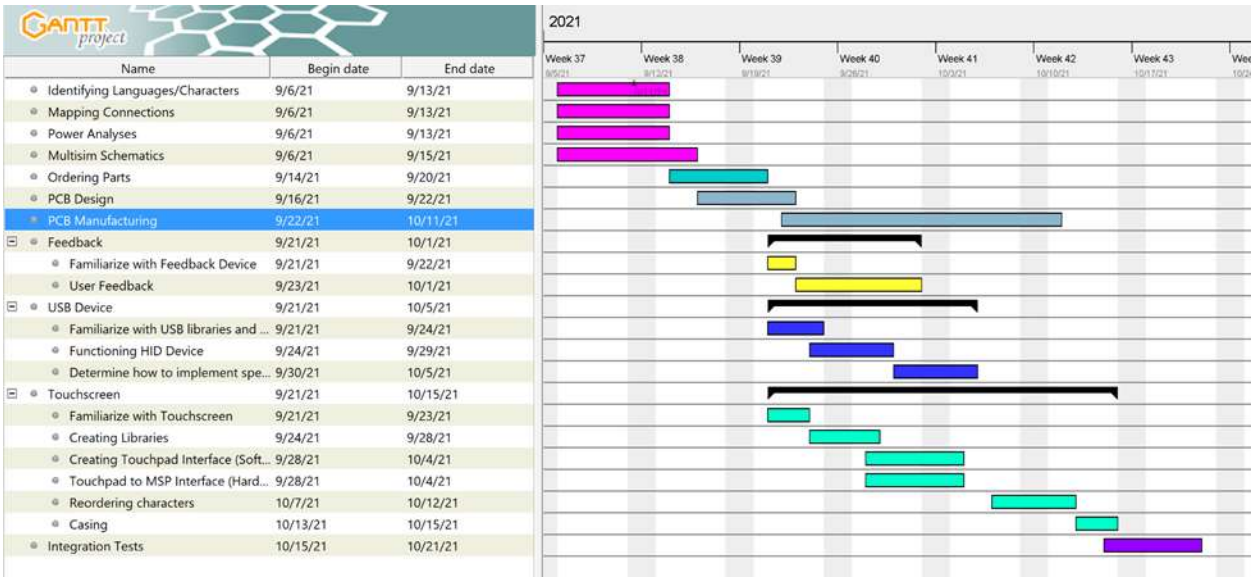
Audio

With the audio subsystem, the team realized while putting the PCB together, that the cables connecting the 3-pin header and the rotary encoder did not match on size. To account for this, female headers and jumpers were used to connect the two pieces. Furthermore, we discovered that the MSP430 was capable of sending PWM signals at only 1 voltage level, meaning that we could not vary the volume of the buzzer by varying the amplitude of our PWM signals, as was previously planned. Therefore, we had to change the software so that the volume was changed by the duty cycle of the signal instead. Finally, we discovered that Pin 6.1, while being a GPIO pin, was not connected to any of the MSP timers, meaning we would be unable to send PWM signals from it. To remedy this, we soldered a wire from Pin 2.0, connected to Timer A, directly to the positive lead of our buzzer. Once this connection was made, we were able to send signals to the buzzer and have it emit the required tones.

Project Timeline

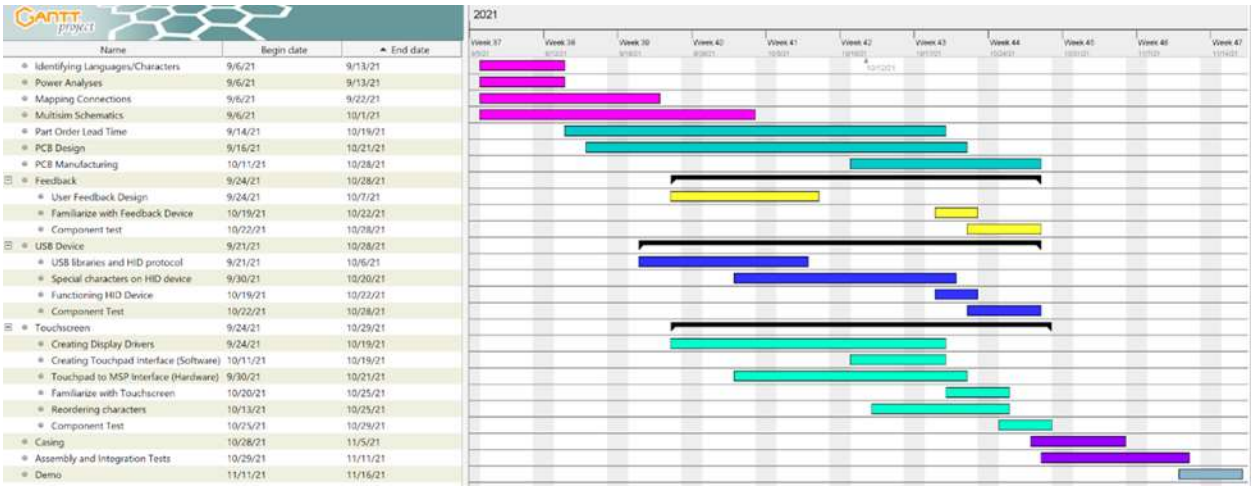
A free Gantt Chart tool was used to create a timeline of the project [50]. The original timeline designed can be seen in Figure 15. In retrospect, this timeline was fairly optimistic. Our first setback came from the components wanted for the project. Mainly, both the audio subsystem and the LCD were too many iterations before a final decision was concise by the

team. This hurt the schematic and the PCB deadlines since they could not be completed until the final decision was made. Finally, the touchscreen section took the biggest hit since a different LCD would require the use of a different code library. Nonetheless, by the Midterm review a new timeline was designed that would account for all the setbacks that the team had until that moment. That timeline can be seen in Figure 16.



Name	Begin date	End date
Identifying Languages/Characters	9/6/21	9/13/21
Mapping Connections	9/6/21	9/13/21
Power Analyses	9/6/21	9/13/21
Multisim Schematics	9/6/21	9/15/21
Ordering Parts	9/14/21	9/20/21
PCB Design	9/16/21	9/22/21
PCB Manufacturing	9/22/21	10/11/21
Feedback	9/21/21	10/1/21
Familiarize with Feedback Device	9/21/21	9/22/21
User Feedback	9/23/21	10/1/21
USB Device	9/21/21	10/5/21
Familiarize with USB libraries and ...	9/21/21	9/24/21
Functioning HID Device	9/24/21	9/29/21
Determine how to implement spe...	9/30/21	10/5/21
Touchscreen	9/21/21	10/15/21
Familiarize with Touchscreen	9/21/21	9/23/21
Creating Libraries	9/24/21	9/28/21
Creating Touchpad Interface (Soft...	9/28/21	10/4/21
Touchpad to MSP Interface (Hard...	9/28/21	10/4/21
Reordering characters	10/7/21	10/12/21
Casing	10/13/21	10/15/21
Integration Tests	10/15/21	10/21/21

Figure 15: Initial Gantt Chart Timeline



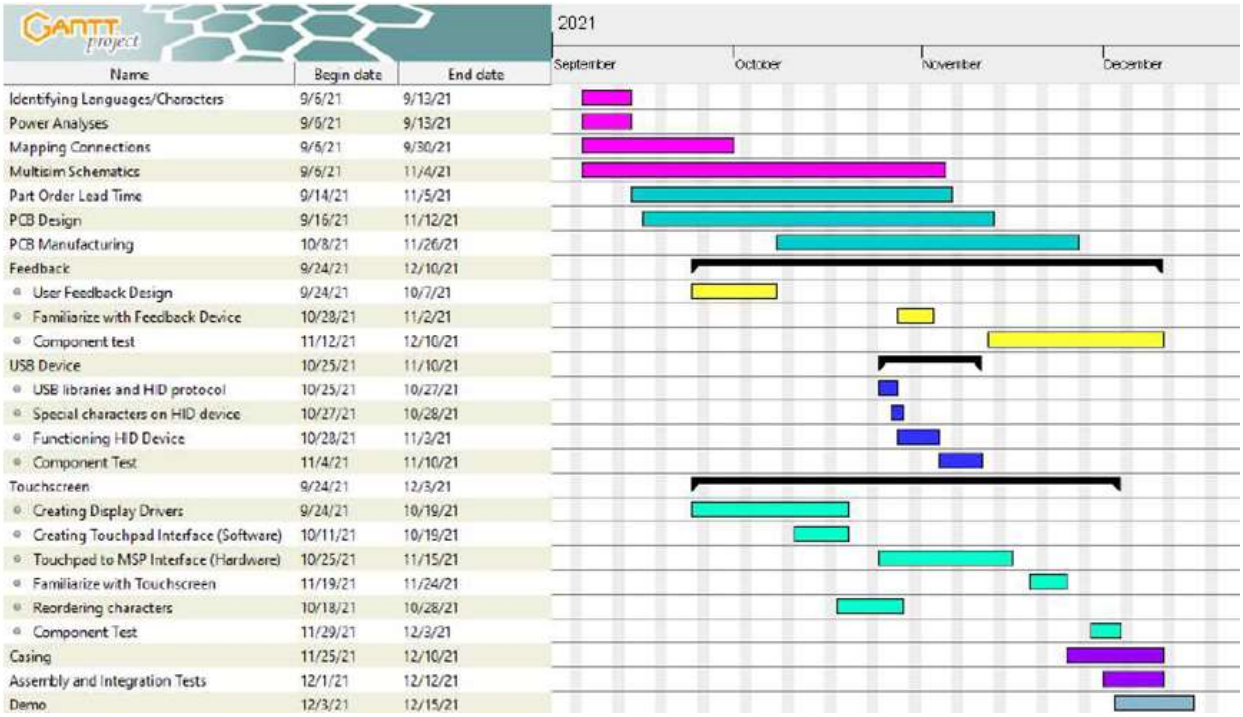
Name	Begin date	End date
Identifying Languages/Characters	9/6/21	9/13/21
Power Analyses	9/6/21	9/13/21
Mapping Connections	9/6/21	9/22/21
Multisim Schematics	9/6/21	10/1/21
Part Order Lead Time	9/14/21	10/19/21
PCB Design	9/16/21	10/21/21
PCB Manufacturing	10/11/21	10/28/21
Feedback	9/24/21	10/28/21
User Feedback Design	9/24/21	10/7/21
Familiarize with Feedback Device	10/19/21	10/22/21
Component test	10/22/21	10/28/21
USB Device	9/21/21	10/28/21
USB libraries and HID protocol	9/21/21	10/6/21
Special characters on HID device	9/30/21	10/20/21
Functioning HID Device	10/19/21	10/22/21
Component Test	10/22/21	10/28/21
Touchscreen	9/24/21	10/29/21
Creating Display Drivers	9/24/21	10/19/21
Creating Touchpad Interface (Software)	10/11/21	10/19/21
Touchpad to MSP Interface (Hardware)	9/30/21	10/21/21
Familiarize with Touchscreen	10/20/21	10/25/21
Reordering characters	10/13/21	10/25/21
Component Test	10/25/21	10/29/21
Casing	10/28/21	11/5/21
Assembly and Integration Tests	10/29/21	11/11/21
Demo	11/11/21	11/16/21

Figure 16: Midterm Review Gantt Chart

However, it did not take one week until the expectation of this timeline became impossible to meet, and most of it came from one unexpected challenge. The MSP430 order did not come until 2 weeks later than all of the other components. To put it into perspective, the Order Set #2 deadlines passed before we got our MSP430. While coding libraries could be investigated and worked on, there was no way for our team to actually test the code itself, which made a long-term setback to the team. Furthermore, while we expected to go through two iterations for the PCB, in the end it took us all three iterations to make a proper PCB, with most

of the problems coming from small issues like component footprints not having the proper size and last-minute decisions for ordering a new component.

Finally, testing was predicted to be time consuming, so all the individual tests where deadlines were pushed, which also led us to push the begin date of the assembly and demo. Nonetheless, all of our deadlines were met, resulting in a functional and mostly error-free system on demo day.



Name	Begin date	End date
Identifying Languages/Characters	9/6/21	9/13/21
Power Analyses	9/6/21	9/13/21
Mapping Connections	9/6/21	9/30/21
Multisim Schematics	9/6/21	11/4/21
Part Order Lead Time	9/14/21	11/5/21
PCB Design	9/16/21	11/12/21
PCB Manufacturing	10/8/21	11/26/21
Feedback	9/24/21	12/10/21
User Feedback Design	9/24/21	10/7/21
Familiarize with Feedback Device	10/28/21	11/2/21
Component test	11/12/21	12/10/21
USB Device	10/25/21	11/10/21
USB libraries and HID protocol	10/25/21	10/27/21
Special characters on HID device	10/27/21	10/28/21
Functioning HID Device	10/28/21	11/3/21
Component Test	11/4/21	11/10/21
Touchscreen	9/24/21	12/3/21
Creating Display Drivers	9/24/21	10/19/21
Creating Touchpad Interface (Software)	10/11/21	10/19/21
Touchped to MSP Interface (Hardware)	10/25/21	11/15/21
Familiarize with Touchscreen	11/19/21	11/24/21
Reordering characters	10/18/21	10/28/21
Component Test	11/29/21	12/3/21
Casing	11/25/21	12/10/21
Assembly and Integration Tests	12/1/21	12/12/21
Demo	12/3/21	12/15/21

Figure 17: Final Gantt Chart

Our project was unique in that it could be paralleled very easily. The designing of the PCB, the testing of the audio subsystem, and the programming of the LCD touchscreen were all able to work on at the same time. Rawan and Pedro worked on the PCB design, Chris and Emory worked on the USB communication between the LCD and the computer and programming the characters into the LCD, and Rohan worked in making the audio subsystem work, while helping debug problems on the LCD. The only part where the project was serialized was at the end, where the all the components were assembled

Test Plan

Hardware Testing

The PCB was tested using the following procedure: connectivity test, solder and part assembly, connectivity test, power supply check, LCD-to-MSP interface check, and finally audio subsystem check. The manufactured board was first tested to ensure the traces were correct and there were no unintended shorts or connections. Once this was complete, the board and

components were taken to WWW Electronics, Inc. A short-circuit and connectivity test was repeated after the fully assembled board was done to ensure no errors occurred in the assembly process.

The power system was tested next because it was critical that there would be no voltages that could be damaging to the MSP or LCD. We included test points for the power jack, 3.3 V regulator, and 5 V regulators. These test points read the correct value when measured using a multimeter. The various headers and nodes across the board connected to these were then double checked. We made the mistake of not including test points for ground. However, ground was verified using the multimeter at each respective pin.

Once the power system was verified to be correct, then the LCD and MSP were plugged into the PCB headers to test the interface. This is where we primarily ran into issues. The LCD connected properly. However, the MSP was not powered correctly by the wall power source and board; this was corrected by changing the MSP configuration so it is powered by the laptop via the USB port, rather than by the power supply on the PCB.

In regards to the audio subsystem, we first tested that each individual component behaved as expected. We started by testing the rotary encoder. First, we recreated the PCB rotary encoder circuit on a breadboard. Next, using a VirtualBench, we ensured that we could read the different states of the rotary encoder when we applied a 3.3V signal [22]. We knew the test would be successful if we could see DC signals at 3.3V (corresponding to a 1) and 0V (corresponding to a 0), and that the different states of the rotary encoder presented themselves in a repeatable sequence, both when we turned the shaft right and when we turned it left. This was a crucial test to perform, as being able to read the states of the encoder was the very basis of our audio subsystem.

Next, we tested the buzzer. Like the rotary encoder test, we recreated the buzzer circuit on our breadboard and used a VirtualBench to provide the necessary PWM signals [22]. After getting this set up ready, we tested the buzzer at 250, 500, 750, and 1000 Hz signals to see if the pitch of the buzzer changed accordingly. Next, we applied a constant 1000 Hz signal and changed the amplitude of the signal to see if the volume of the generated tones changed. Once we had tested these components individually, we connected the components to the PCB.

Using the test points, we first set about testing the buzzer. However, we ran into an issue when we noticed that the pin that buzzer was tied to, Pin 6.3, was not capable of sending PWM signals, meaning we could not turn the buzzer on. To correct this, we jumped a wire from Pin 2.1, which was connected to Timer A on the MSP, to the positive lead of the buzzer. Once we had this in place, the buzzer worked as expected. Next, we tested the rotary encoder on the PCB.

We initially ran into issues with the rotary encoder where we could read values of 2.4 V instead of the expected 3.3 V for certain positions of the encoder. We discovered this issue was due to a faulty connection with the encoder's 3 pin connector. We corrected this by soldering the wires to the connector rather than using ribbon cables. This corrected the issue, which then verified that everything on the PCB was working properly except the MSP power issue resolved separately as described above.

LCD Testing

The first substantial part of the LCD testing plan was to verify that SPI commands were working correctly. In order to verify this functionality, an Analog Discovery 2 was connected to the MOSI, SOMI, select, and clock lines of the MSP [23]. By monitoring these wires, we were able to see exactly what was being sent over SPI. Using this information, we monitored the lines while the MSP was running the startup sequence for the LCD. We were able to verify that the correct commands were being sent in the correct order. Additionally, we were able to verify that whenever a read command is given to the LCD, it is able to send back the correct bits. This test was integral to getting the rest of the LCD functionality, as it is integral to the device.

The second substantial testing that was done was following the connection of the LCD and USB communication components. When trying to bring these two parts together, we did testing to ensure that both functions worked together without interference. To do this, we separately tested all of their functionality and brought them together when we knew they were functional. When we did this combination, we encountered some problems. When the two components were brought together, the HID functionality of the USB started to fail. After much trial-and-error testing, we determined a solution to be powering the MSP with the USB power and keeping the LCD separate, using the power from our PCB. This testing was important because without it, we would not have found the bug that occurs when the two parts are combined.

Audio Testing

Once we had verified that the audio hardware was working, we proceeded to test the audio software. First, we checked to see if we could read the states of the rotary encoder as we could on the VirtualBench [22]. This test initially gave us some trouble, as the states would unexpectedly change, and they would sometimes not respond to changes in the shaft position. We discovered that loose connections in the circuit as well as software initialization issues were causing these troubles. Once we switched to using the MSP430 library as opposed to setting the pins as inputs manually and rechecked the circuit connections, the rotary encoder states were being read properly.

Once we confirmed that, we then proceeded to verify that we could send PWM signals with varying amplitudes to the buzzer in order to change its volume. However, we soon discovered that the MSP430F5529LP can only send PWM signals of a single amplitude, and that the pin the buzzer was routed to, Pin 6.3, could not support PWM signals via a timer. To fix this, we switched the output pin of the buzzer to Pin 2.0, which was connected to Timer A, and used duty cycles to vary the volume. We discovered that at very low frequencies (around 250Hz), varying the duty cycle of the signal also varied how loud it could be. Once we made these changes, we were able to hear tones from the buzzer and change the volume of those tones from the rotary encoder, as was expected.

USB Testing

For the testing of the USB software, we developed many different test programs throughout the semester. We started with a simple program that set up the microcontroller as an HID device and typed out a predefined character when a button was pressed. Once we were able to get this program running, we demonstrated the capability of our device to type characters over a USB connection. The next progression of our testing was a simple program that was capable of

typing all of the accent Spanish characters with a button press. This required us to create a demo keyboard layout as well to enable the typing of these special characters. We built upon the previous program to create this one. Once it was successfully running, we were able to show that our device could be successfully configured to type out special characters. This program was later expanded to simulate typing all of the characters that we would need to use for our device to fully confirm that it was capable of typing everything that we needed. Since we were able to get these programs to work with only simple software debugging techniques, we did not have to resort to testing the readings of the signals with an oscilloscope.

After we did individual testing for the typing capabilities of the special characters, we created a number integration test programs. The first integration program used a physical number pad to simulate the different buttons that would be available on the touchscreen. This program tested the backend software that would be similar to the software necessary to software that would later run the LCD touchscreen. This program tested that we could assign different characters to different pages, switch between languages, and rearrange keys. After this was successfully implemented, we decided to add in a display through the serial terminal to simulate the display of the LCD before it was working. Once we made this integration program, we were able to fully simulate the touchscreen functionality without using and verify that the underlying software could work. Once the LCD was running with a GUI, we integrated the two pieces of code together. Despite a power issue described in the Problems and Design Changes section, these were able to integrate together fairly smoothly, and only basic software debugging was needed for integration. This integration also required that the physical keypad and serial terminal display be removed from the software. Once this integration was complete, the core functionality of our device was completed, and we continually user tested the functionality to verify that it still functioned as we added more features to the device.

Final Results

The final results for the LANGPAD included a functioning USB touchpad device that is capable of typing special characters in Spanish, Greek and French in any text editor the user wishes to type in. The user is able to power the device via a wall transformer and connect the device to their laptop via a USB cable. Additionally, the LCD is responsive to the user's touch inputs, allowing them to select characters and rearrange the characters to their liking. Although we did experience some difficulties in regards to the circuitry, specifically with the audio subsystem, we were able to easily remedy those issues, and as such, the PCB is able to provide the necessary functionality for all of the LANGPAD subsystems. Tapping a button the device emits a tone from the piezo buzzer, and the volume of this tone can be changed via a rotary encoder. Finally, the entire device is enclosed inside of a 3D printed case that provides easy access to the rotary encoder, cable ports, and the LCD screen. The enclosure can also be easily disassembled via a screwdriver, in case we need to make some changes to the device's interior.

Given that the device works as we intended it to, the finalized LANGPAD satisfies the proposed functionality in Table 1, and thus meets the A+ bracket.

Table 1: Grade Rubric

Letter Grade	Grading Criteria
A+	• The device inputs the characters the user chooses onto the user's preferred choice of text editor in the same amount of time it would take the user to type on a normal keyboard (no added latency). • The device allows for switching between the special characters of multiple languages. • The device successfully powers the LCD backlighting and logic when plugged into the wall and is able to power the audio subsystem. • Audio feedback is provided to confirm to the user when a key is pressed. • Professionalism of final product and report, including meeting standards and creating an appealing aesthetic of the design.
A	• Device does not perform one of the tasks required for an "A+"
B+	• Device does not perform two of the tasks required for an "A+".
B	• Device does not perform three of the tasks required for an "A+".
C	• Device does not perform four of the tasks required for an "A+"
D	• Device does not perform five of the tasks required for an "A+"

In addition to the goals specified in the table above, we were able to add some additional functionality to the LANGPAD. For example, when the user changes the volume via the rotary encoder, a volume bar on the LCD itself will progress, giving the user a visual cue as to how loud the system is currently and ensuring them that their volume change request has been properly handled. Additionally, we had originally planned to support Spanish and French, however, we were able to add functionality for Greek characters as well for those writing electrical engineering reports.

The final result of the LANGPAD has demonstrated the importance of multiple core values that constitute any successful team. One such value was constant and open communication. By being prompt and honest with each other about the difficulties we were

facing on our parts of the project, we were able to address those concerns quickly and efficiently. This constant and open communication was part of a culture of learning that we created in our team, encouraging each other to ask questions without fear of being judged by the other team members. Another value we learned was time management and organization. By keeping ourselves organized and on-track with our Gantt chart, we were able to plan ahead and react to any unforeseen stumbles, such as the delay in receiving our microcontrollers, while still being productive. By following these values, we were able to meet our end goals and deliver the finalized LANGPAD.

Costs

The cost of our individual project was significant. Below is a table outlining the costs of the components of the LANGPAD for one unit, 1,000 units, and the number of units used to produce one LANGPAD device:

Table 2: Cost Breakdown for the LANGPAD

PCB parts	Cost for 1 unit	Cost per unit for 1,000 units	Total used in 1 PCB	Total price
Buzzer	$0.97	$0.52	1	$0.97
Rotary Encoder	$3.37	$2.25	1	$3.37
20 male's headers	$0.89	$0.66 (for 320 units)	1	$0.89
20 female headers	$1.26	$0.70	2	$2.52
Barrel Jack	$0.78	$0.40 (for 1,120 units)	1	$0.78
3.3 Voltage Regulator	$0.75	$0.378	1	$0.75
5 Voltage Regulator	$0.87	$0.38	1	$0.87

3.3 TVS	$0.40	$0.12	2	$0.80
10 nF Capacitor	$0.70	$0.21	2	$1.40
10 uF Capacitor	$1.22	$0.44 (for 500 units)	4	$4.88
1k Resistor	$0.30	$0.041	2	$0.60
330 Resistor	$0.83	$0.199	2	$1.66
3 Pin Connector	$1.46	$1.024	1	$1.46
Test Points	$0.34	$0.198	15	$5.1
Ribbon Cable	$1.26	$0.69	1	$1.26
Touchscreen	$147.32	$147.32	1	$147.32
MSP430	$17.28	$17.28	1	$17.28
PCB	$33	$4.75	1	$33
Total	$213	$177.56		$224.91

From Table 2 it can be seen that the components contributing the most to the cost of the LANPAD are the MSP430, and the LCD touchscreen. If this project was to be mass produced, the total cost would decrease by around 21.05%, from $224.91 to $177.56 which does not reduce the price exponentially. It is important to note that this calculation would be a rough estimate since the mass production cost assumes that all components are used only once, while the actual PCB uses some of the components twice or thrice. On the bright side, this only occurs with the smaller priced components, so the difference between the expected reduced value, and the actual reduced value will not be significant.

Furthermore, the LCD touchscreen chosen for this project was selected due to the limited amount of time to work on the project and due to the abundance of coding libraries available

online, simplifying most of the code. Since mass production wouldn't deal with both of those issues (in theory), then a cheaper LCD touchscreen would be used for the LANGPAD. Similarly, the MSP430 development kit would be replaced with only the microprocessor being connected into the PCB reducing the price, although this approach will require small changes to the PCB like adding an USB port

You can see a table looking at all the components bought during this project, including the components not used, on Appendix A.

Future Work

In order to improve the design, there would be a few different areas that would be changed. Firstly, a modification we had to make to the board was soldering the rotary encoder directly to the PCB. This addition was not by design, as there was a designated three pin connector that it was supposed to plug in to. However, the cable we ordered to fit that connector did not work as intended. Thus, we would fix that connection in a future iteration. Next, an oversight occurred and there was no test point made for ground. In a future model we would add a test point for ground to the PCB for easier testing. Additionally, an improvement on the product would be to add support for more languages. The tab and character layout of the device is modular, so adding more languages would be both useful and straightforward. Currently, the device is only supported on windows. Modifications could be made in the future so that it works on both Mac and Linux as well. Additionally, the current casing could be improved upon so that the plugs for the power and computer connection are counter-sunk into the plastic of the case so that there is not a large square hole in the side of the device. Additionally, an on-off switch could be added to the device, as currently the device powers on upon plugging into the computer and wall socket. Also, in order to streamline the device footprint, the microcontroller launchpad could be consolidated in a way that everything resides on the PCB. This modification would include moving both the USB connection and the microcontroller chip to the PCB. This change would allow us to make the device significantly smaller.

References

[1] "The digital language divide." http://labs.theguardian.com/digital-language-divide/ (accessed Oct. 22, 2021).

[2] J. Stamp, "Fact of Fiction? The Legend of the QWERTY Keyboard," Smithsonian Magazine. https://www.smithsonianmag.com/arts-culture/fact-of-fiction-the-legend-of-the-qwerty-keyboard-49863249/ (accessed Sep. 10, 2021).

[3] "How the QWERTY keyboard shapes the way we communicate," The British Academy. https://www.thebritishacademy.ac.uk/blog/summer-showcase-2020-how-qwerty-keyboard-shapes-way-we-communicate/ (accessed Sep. 10, 2021).

[4] “Multisim,” National Instruments. https://www.ni.com/enus/shop/software/products/multisim.html (accessed Dec. 16, 2021).

[5] “Ultiboard,” National Instruments. https://www.ni.com/enus/shop/software/products/ultiboard.html (accessed Dec. 16, 2021).

[6] “CCSTUDIO_10.1.1.00004 | TI.com.” https://www.ti.com/tool/download/CCSTUDIO (accessed Dec. 16, 2021).

[7] *Microsoft Keyboard Layout Creator*. Microsoft. Accessed: Dec. 16, 2021. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=102134

[8] “The Environmental Impact of Corn-Based Plastics,” *Scientific American*. https://www.scientificamerican.com/article/environmental-impact-of-corn-based-plastics/ (accessed Dec. 17, 2021).

[9] A. Guha Roy, “Recycling LCD panels,” Nat Sustain, vol. 2, no. 6, pp. 442–442, Jun. 2019, doi: 10.1038/s41893-019-0320-4.

[10] M. Gerić, G. Gajski, V. Oreščanin, A.-M. Domijan, R. Kollar, and V. Garaj-Vrhovac, “Environmental risk assessment of wastewaters from printed circuit board production: A multibiomarker approach using human cells,” *Chemosphere*, vol. 168, pp. 1075–1081, Feb. 2017, doi: 10.1016/j.chemosphere.2016.10.101.

[11] “USB Human Interface Devices - OSDev Wiki.” https://wiki.osdev.org/USB_Human_Interface_Devices

[12] ”Learn about the Serial Peripheral Interface(SPI),” VectorNav. https://www.vectornav.com/resources/inertial-navigation-primer/hardware/synccomm/ (accessed Dec 17, 2021).

[13] “ASCII Table - ASCII codes,hex,decimal,binary,html.” https://www.rapidtables.com/code/text/ascii-table.html (accessed Sep. 10, 2021).

[14] “List of Unicode characters,” Wikipedia. Aug. 26, 2021. Accessed: Sep. 10, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=List_of_Unicode_characters&oldid=1040787898

[15] “Build software better, together,” GitHub. https://github.com (accessed Dec. 17, 2021).

[16] “GitKraken | Legendary Git Tools.” https://www.gitkraken.com (accessed Dec. 17, 2021).

[17] D. Ritchie, The C Programming Language. Bell Labs.

[18] “MSP430USBDEVPACK Application software & framework | TI.com.” https://www.ti.com/tool/MSP430USBDEVPACK (accessed Sep. 10, 2021).

[19] “NHD-5.0-800480FT-CTXL-CTP.” Newhaven Display, May 08, 2020. Accessed: Dec. 16, 2021. [Online]. Available: https://www.newhavendisplay.com/specs/NHD-5.0-800480FT-CTXL-CTP.pdf

[20] Inventor Software. Autodesk. Accessed: Dec. 17, 2021. [Online]. Available: https://www.autodesk.com/products/inventor/overview

[21] “Ender-3 3D Printer.” https://www.creality.com/goods-detail/ender-3-3d-printer (accessed Dec. 17, 2021).

[22] ”VirtualBench All-in-One Instrument - NI,” NI. https://www.ni.com/en-us/shop/hardware/products/virtualbench-all-in-one-instrument.html (accessed Dec 17, 2021).

[23] ”USB Oscilloscope and Logic Analyzer - Digilent Analog Discovery 2,” Diligent. https://digilent.com/shop/analog-discovery-2-100ms-s-usb-oscilloscope-logic-analyzer-and-variable-power-supply/ (accessed Dec 17, 2021).

[24] “Understanding Ethical Sourcing in the Supply Chain,” Intertek. https://www.intertek.com/blog/2016-11-15-ethical-sourcing/ (accessed Dec. 17, 2021).

[25] Kate Conger “Tech has more of an exclusion problem than a culture problem,” Techcrunch. https://techcrunch.com/2016/05/09/tech-has-more-of-an-exclusion-problem-than-a-culture-problem/ (accessed Dec. 17, 2021).

[26] “From Prototype to Mass Production - Understanding Scaling Costs for Physical Products,”. Predictable Designs. https://predictabledesigns.com/from-prototype-to-mass-manufacturing-understanding-scaling-costs-for-physical-products/ (accessed Dec. 17, 2021).

[27] L. Ou *et al.*, “Virtual keyboard input for international languages,” US10073536B2 Accessed: Dec. 17, 2021. [Online]. Available: https://patents.google.com/patent/US10073536B2/en?q=international+characters+keyboard&oq=international+characters+keyboard

[28] Wu, Sunjo and Wu, Sunjo, “多国語キー入力装置及びその方法,” JP6000385B2 Accessed: Dec. 17, 2021. [Online]. Available: https://patents.google.com/patent/JP6000385B2/en?q=keyboard+extension&q=international+characters&page=1

[29] “MSP430F5529 LaunchPadTM Development Kit (MSP-EXP430F5529LP).” Texas Instruments, Apr. 01, 2017. [Online]. Available: https://www.ti.com/lit/ug/slau533d/slau533d.pdf?ts=1639635887190

[30] “PJ-059A Datasheet - Jacks | Dc Power Connectors | CUI Devices.” CUI Devices, Apr. 04, 2016. Accessed: Dec. 16, 2021. [Online]. Available: https://www.cuidevices.com/product/resource/pj-059a.pdf

[31] “Very low drop voltage regulator with inhibit function.” ST, May 01, 2017. Accessed: Dec. 16, 2021. [Online]. Available: https://www.st.com/content/ccc/resource/technical/document/datasheet/c4/0e/7e/2a/be/bc/4c/bd/CD00000546.pdf/files/CD00000546.pdf/jcr:content/translations/en.CD00000546.pdf

[32] “µA78Mxx Positive-Voltage Regulators datasheet (Rev. T).” Texas Instruments, Jan. 01, 2015. Accessed: Dec. 16, 2021. [Online]. Available: https://www.ti.com/lit/ds/slvs059t/slvs059t.pdf

[33] V. Vitramon, “Surface Mount Multilayer Ceramic Chip Capacitors for Commercial Applications.” VISHAY, Apr. 10, 2019. [Online]. Available: https://www.vishay.com/docs/45199/vjcommercialseries.pdf

[34] “100” [2.54 mm] Contact Centers, Female Headers, Straight/Right Angle/SMT.” Sullins. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/Sullins%20PDFs/Female_Headers.100_DS.pdf

[35] “PTVSxS1UR.” Nexperia USA Inc., Jan. 10, 2021. [Online]. Available: https://rocelec.widen.net/view/pdf/u7nc6qq6tp/PHGLS22186-1.pdf?t.download=true&u=5oefqw

[36] “Series 288 Data Sheet Rotary Encoder.” CTS Electrocomponents, Oct. 11, 2017. [Online]. Available: https://www.ctscorp.com/wp-content/uploads/288.pdf

[37] “SMD POWER RESISTORS AEC-Q200 QUALIFIED.” TE Connectivity . [Online]. Available: https://www.te.com/commerce/DocumentDelivery/DDEController?Action=srchrtrv&DocNm=9-1773463-5&DocType=DS&DocLang=English

[38] “RMCF / RMCP Series General Purpose Thick Film Standard Power and High-Power Chip Resistor.” Stackpole Electronics, Inc., Aug. 09, 2021. Accessed: Dec. 16, 2021. [Online]. Available: https://www.seielect.com/catalog/sei-rmcf_rmcp.pdf

[39] “MC 0,5/ 3-G-2,5.” Phoenix Contact, Mar. 11, 2010. [Online]. Available: https://www.digikey.com/en/products/detail/phoenix-contact/1881451/308247

[40] “CPE-163 Datasheet - Audio Transducers | Buzzers | CUI Devices.” CUI Devices, May 05, 2020. Accessed: Dec. 16, 2021. [Online]. Available: https://www.cuidevices.com/product/resource/cpe-163.pdf

[41] “Box Header & IDC Connector WR-BHD.” Würth Elektronik. Accessed: Dec. 16, 2021. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/Wurth%20Electronics%20PDFs/Box%20Header_IDC%20Conn%20WR-BHD.pdf

[42] “Ribbon Cable Length REV1.” Digi-Key. Accessed: Dec. 16, 2021. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/Digi-Key%20PDFs/Ribbon_Cable_Assembly_Length.pdf

[43] “(THM) THRU HOLE MOUNT TEST POINTS - COLOR KEYED.” Keystone, Jul. 31, 2021. Accessed: Dec. 16, 2021. [Online]. Available: https://www.keyelco.com/userAssets/file/M65p56.pdf

[44] D. Ritchie, The C Programming Language. Bell Labs.

[45] W. Goh and K. Quiring, “MSP430™ USB HID Windows API Programmer’s Guide,” ti.com, Jan. 2011.

[46] *Windows 10*. Microsoft. Accessed: Dec. 16, 2021. [Online]. Available: https://www.microsoft.com/en-us/windows/get-windows-10

[47] ”EVE Projects - Bridgetek,” Bridgetek. https://brtchip.com/softwareexamples-eve/ (accessed Dec 17, 2021).

[48] “How Rotary Encoder Works and How To Use It with Arduino,” HowToMechatronics, Jul. 25, 2016. https://howtomechatronics.com/tutorials/arduino/rotary-encoder-works-use-arduino/ (accessed Dec. 17, 2021).

[49] “NHD-5.0-800480TF-ATXL#-CTP,” newhavendisplay.com, May 11, 2021. https://www.newhavendisplay.com/specs/NHD-5.0-800480TF-ATXL-CTP.pdf

[50] B. S. s.r.o, “GanttProject: free project management tool for Windows, macOS and Linux,” GanttProject. https://www.ganttproject.biz (accessed Sep. 10, 2021).

Appendix A: Project Costs

The project cost is the total cost that took the team to finish the product. This include extra MSP430 used for prototyping, components whose values were altered at later points of the project, the two first iterations of the PCBs, components whose sizes did not matched what their intention were for, components that were accidentally bought twice, and components with extra pieces that ultimately were not used, alongside with the components that were used in the end

Item	Quantity	Individual Price	Total Price
Printed Circuit Board	3	$33	$99
Order 1			
MSP430	3	$17.28	$51.84
Touchscreen	1	$147.32	$147.32
Beeper/Feedback	2	$0.97	$1.94
Rotary Encoder	2	$3.37	$6.74

PCB to LCD cable	2	$1.26	$2.52
Barrel Jack	3	$0.78	$2.34
3.3V Regulator	3	$0.75	$2.25
5V Regulator	3	$0.87	$2.61
TVS Diode 5V	4	$0.70	$2.80
3 Position Connector	2	$3.13	$6.26
20 pin header female	4	$1.26	$5.04
20 pin header male	2	$0.89	$1.78
2 pin header female	6	$0.32	$1.92
Order 2			
MSP430	2	$17.28	$34.56
H3CCH-2006G (Ribbon Cable)	3	$1.26	$3.78
Test Points	25	$0.34	$8.52
Power Jack	3	$0.78	$2.34
Ribbon Cable for Rotary Encoder (IDSS-03-D-18.00)	2	$8.39	$16.78
3-Pin Connector (1881451)	2	$1.46	$2.92
330 ohm resistor	3	$0.83	$2.49
1k ohm resistor	3	$0.30	$0.90

10uF cap	4	$1.22	$4.88
10 nF cap	3	$0.70	$2.10
0.1 uF cap	3	$0.73	$2.19
2.2 uF cap	3	$0.87	$2.61
0.33 uF cap	3	$0.65	$1.95
3.3V TVS	3	$0.40	$1.20
Order 3			
A03XAF03XAF22K254B	3	$1.38	$4.14
A03XAF03XAF22K51A	3	$1.34	$4.02
Total Price	**$428.74**		