Handheld LED Display

Group 21

Team Members:

Peter Guarner, EE
Sean O’Shaughnessy, CpE
Tyndall Darnell, EE
Tyler Stokes, CpE

Sponsor:

Michael Young

12/4/2017

1. Executive Summary

When considering situations where communication of a visual message to a particular person or a crowd of audience at a distance is needed at any given moment, there aren't many reliable products available. The purpose of the project is to design a handle-held LED display that illuminates text that can be seen from a distance, preloaded with messages given by the user.

With USB or optional Bluetooth connectivity and a convenient programming interface, users will be able to load a maximum 4 preset messages to the device memory for quick selection. The display will be capable of displaying up to 2 lines of readable text, along with optional emoticon graphics. Additionally, users will have the ability to adjust the brightness of the matrix and check the battery level of the system using buttons located within proximity of the device handle.

Currently, such devices that are hand-held are not produced. However, there are a few products available that are similar, with a programmable LED display and reasonable size for transporting. The issue with these products is that they are either not hand-held capable, very expensive, or have a small screen size with minimal resolution. Also, programming some of these becomes somewhat complex for users, since configuration of these devices may not have been designed for programming novices.

The goal of this project is to design, build, test, and debug a hand-held LED display that, with production documentation, will be available at a reasonable price. To achieve this goal, only the necessary feature additions were selected, as excessive enhancements to the project would raise the cost and difficulty for design. From this, the decision was made to include improvements such as: lightweight, low cost, compact encasing, removeable handle, wireless interface, optimized battery life, larger message capability, and in-use brightness adjusting. Considering these features, as well as opportunities to mass produce the device, this product should provide something that consumers would purchase if properly marketed.

2. Project Description

The purpose of this project is to produce a rechargeable, cost-effective, handheld LED display. Our goal is to make the display where users will have the ability to see messages from the LEDs clearly at a distance of at least 10 meters away, with optional text and configuration of the characters. To achieve this, over 2,000 LEDs will be used, forming a matrix of 32 rows and 64 columns that all fit in a compact encasing. This way, maximum brightness will be produced, and text will be clearly defined over a span of numerous, miniature LEDs. Messages can also be loaded in a scrolling fashion when the numbers of characters exceeds the lateral space of the display, or text will be automatically centered on the display when messages can fit within the display. Users will have access to a wide variety of characters including letters, numbers, and a certain amount of symbols and punctuation.

Loading messages on the screen will be made easy by a simple GUI, where users can adjust their text and other options, and then simply upload the message to the device. The GUI will be available from a computer disk that will have the ready-to-run software stored. The user only needs to run the CD and follow the on screen prompts to start the program and become connected to the device. Messages can be uploaded to the board by using the provided USB cable and connecting to your PC, or by connecting wirelessly to your mobile Android device through Bluetooth. Connecting through Bluetooth will be made possible through an included Bluetooth Low Energy (BLE) module and an app that can be downloaded on to any Android O.S. Users just pair their device to the display with the Bluetooth pairing button on the rear of the display encasing, and the connection will be established for use any time both devices are active. Both interfaces will be made for easy operation, but will still have full functionality for all of the available features.

One unique feature of the display is the ability of the user to add not only one, but up to four preset messages for which they will be able to switch through during use. Four buttons on the back of the device will allow the user to quickly select any of their four desired programs. Switching programs will automatically occur once the button is pressed for that preset message, and the message will remain on the display until another button is pressed by the user. As the user changes these presets from their computer or mobile device, the new message will be illuminated on the display, so users can see their new changes instantly.

Another feature available on the handheld screen, which is accessible by two buttons, is adjustable LED dimming. With a button provided for either raising or lowering the brightness, users can easily adjust the illumination intensity to make their message clear during the day, or reduce the harsh brightness at night. Dimming on the display will be controlled using pulse width modulation, a method for altering the percentage of time for which the LEDs are ON versus OFF over a continuous cycle of toggling, also called the duty cycle. As the user adjusts the brightness through the input buttons, the duty cycle of the LEDs is changed accordingly. Of course, this toggling will occur as a frequency that the human eye will not be able to detect, making the display seem constant for the LEDs that are ON.

This frequency consideration is also taken into account for another aspect of the screen manipulation, which is the data refreshing. The LEDs will be controlled in by activating certain rows and columns that correspond to a particular light, rather than controlling each of the 2,000 LEDs individually. While this saves complexity and cost, this approach is not effective for static control of certain patterns of LEDs. Therefore, data needs to be transmitted in series of refreshing, where either columns or rows are continuously cycled ON, while the other pushes data at each iteration. Our final product will also use this approach and 'refresh' the screen at, again, a frequency that is too fast for humans to notice.

To handle all of this computing and provide a viable interface for the device software to reside, the Atmega328 processor will be employed on the PCB. The processor comes from an Arduino Mega, a development board made for electronics hobbyists. The advantage from this is that prototyping is extremely convenient with this development board, allowing us to quickly test the processor with the components that we are going to use in the final product. Another appealing characteristic is the amount of digital input/output pins, for which there 56 total, making for a wide allowable range of wires to control without concern of limited control capability. To make implementation even easier, thorough documentation is available from a vast range of online sources. All of this is offered in a package that is offered at a low price of around $10-$15, which is an easy decision among the selection of processors out there.

Once the entire PCB for the device is complete, the electronics will have to be secured by some form of hardware. The plan is to devise a slim, rectangular profile encasing that will not only provide protection for the circuitry, but also a clean look where users can easily find and interact with the ports on buttons available. The buttons will be set in the back of the encasing, close to the bottom for easy access when the user holds the handle. The handle will be attached underneath the display encasing, and will be removable for less bulky transportation. To make the handle comfortable for the user, the shape will be made cylindrical and the length will be such that the user can grasp the handle along their full palm.

The last objective that is crucial for the success of the device when presented is the longevity the device can operate after a full recharging of the battery. Life of the battery is needed in order for the portability of the device to be sufficient, at least for the battery life to last for around 45 minutes, a reasonable duration for mobile technology especially for devices that operate under such considerable power consumption. Our team is going to optimize this with two important characteristics: a battery that has maximum storage capability, and hardware/software components that are configured for the lowest electrical usage possible. Together, these characteristics provide the device the longest available lifespan, while still giving the user every available feature at full functionality. Considering all of the above aspects of this project, the device should definitely deliver on the promise for a great and fully functional product. The straightforward and clean layout designed for this entire project was made to keep the technical details organized and allow time for continuously optimize the design for increased performance. Our sponsor and project guide, Michael Young, expects a fully functioning product for which, if he is interested, will also purchase the final device at a price of $350. While our expenses have accumulated over testing trials, updated purchases, and additions to the original design, the goal is to keep the base price of the device down as low as possible. In that way, this project is not only a mission to complete tasks by a certain deadline, but to also produce the best viable product at the most feasible single unit price possible.

2.1 Device Model

A 2-Dimensional model for the entire device is shown below in a simplified computer aided design (CAD) drawing, referring to Figure 2.1. This was produced using the student software version of AutoCAD, which is a modelling tool used to make precise layouts of parts and facilities. In the drawing, the basic initial layout is shown, with intended dimensions and button configurations. There are not any electrical details in this model, only a physical representation to show the overall appearance and size of the final product. The purpose of this layout is to gain a perspective of the project plan, and to visualize the placement of the physical exterior components. This helps the team construct each piece of the device to a template or objective, so that the path for the design is clear, and that the sizing for every piece is verified.

As shown in the figure, the plan for the device is to have 32 x 64 LEDs in the display. To acquire an understanding of the size requirements for this many LEDs, an array in the drawing was made for that exact matrix of LEDs. Using the dimensions for each LED as specified in the datasheet for the selected LEDs (.12 in x .12 in), the display was drawn up. Based on a spacing of about 1 mm between LEDs and considerations for ¼'' margin from all edges, the PCB was determined to fit in a 5 in. x 13 in. total encasing. The dimensions suggested for consumer use was anywhere from 4 in. x 10 in. to 6 in. x 15 in. for a handheld screen device, so our designed encasing should easily fall in the allowable sizing range. This is from the requirements that the screen needs to be large enough for user readability, but compact enough for easy portability. From our design, we believe that our dimensions are not only feasible for these requirements, but also optimal as we chose the approximate 'middle ground' compromise for such conditions.

Another perspective that was necessary for graphical layout was the placement of the buttons. While this task may seem trivial, plenty of thought was given to how the buttons could be best placed for ease and practicality. The majority of the message control buttons, for example, were placed in the rear of the device but lowered close to the handle, so that the user could easily reach these buttons with their finger, while that hand holds the handle. This makes operation quick and convenient for the user, and an important first step for the design.

The different buttons needed for the device to operate include: four message preset buttons to quickly switch display programs, a Bluetooth pairing button, a power/sleep mode button, and two buttons for brightness control. The preset buttons were placed closest to the handle, as they would be the most used during standard operation. The next four buttons, Bluetooth, brightness up/down, and power, were placed just above the presets to not only provide symmetry for the layout, but also a set up that helps the user distinguish the placements of presets and auxiliary functionality. Otherwise, a convolution of button selections in the same row or tightly packed region would make usage confusing and undesirable. Between the buttons, a space of ¼'' was implemented in order to keep the button set close together, but far enough to void any chance of accidentally hitting more than one intended button at once. To find this optimal spacing, physical test markings were made to simulate the spacing, and ¼'' seemed the best minimal spacing for the mentioned constraints. The size of the individual buttons was kept to a ¼'' x ¼'' space for the presets, brightness, and Bluetooth, since this was a good minimal size for easy operation. For power, however, the button was made larger, taking around ½'' x ½'' space, to distinguish this from the normal control functions. This is commonly used in practice so that users can quickly identify the button to power on the device, and so that the power button isn't accidentally pressed from visual confusion. That way, the power button is separately designed from a display control type of input.

Also in the drawing is the placement of the USB port. This port will handle the programming and charging aspect of the device and is placed strategically on the side of the device encasing, rather than the back. The reason for this is that users will likely want to lay the device flat on its back or face when not in use, but either charging or programming. To avoid conflict with accessing the port to a cable in this case, the device has the USB connection on the side. Users will have the ability to conveniently lay the device down on the face or back, and charge/program the display, which also allows users to see their preset messages as they program the device. Placing this single, and only, device port will be accomplished by simply implementing a USB connector closest to the edge of the inner PCB, with the connector able to protrude through the encasing for user access. The port/connector for which the team will input the operational software is not shown on this drawing, because this is not intended for user access and can be embedded on the PCB. This will be placed wherever space can be made on the PCB inside the encasing.

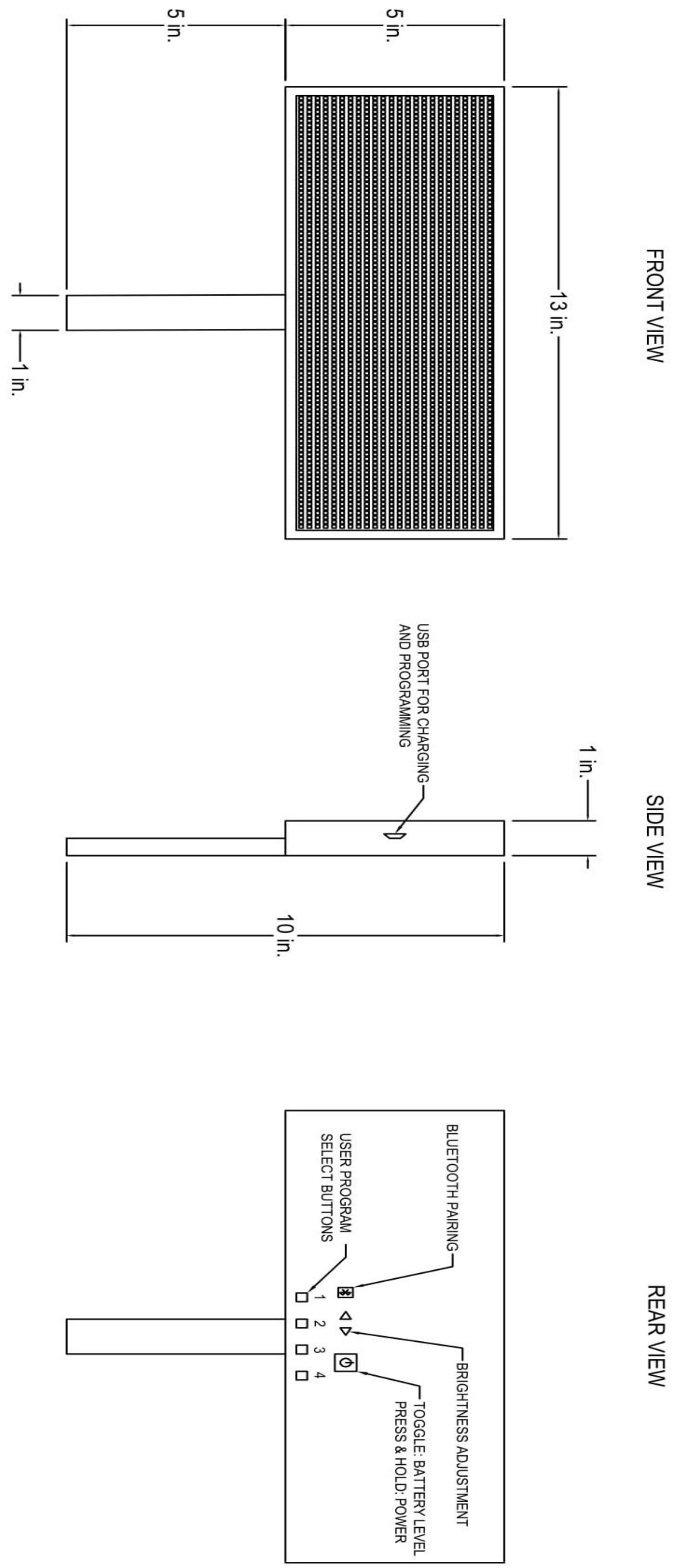


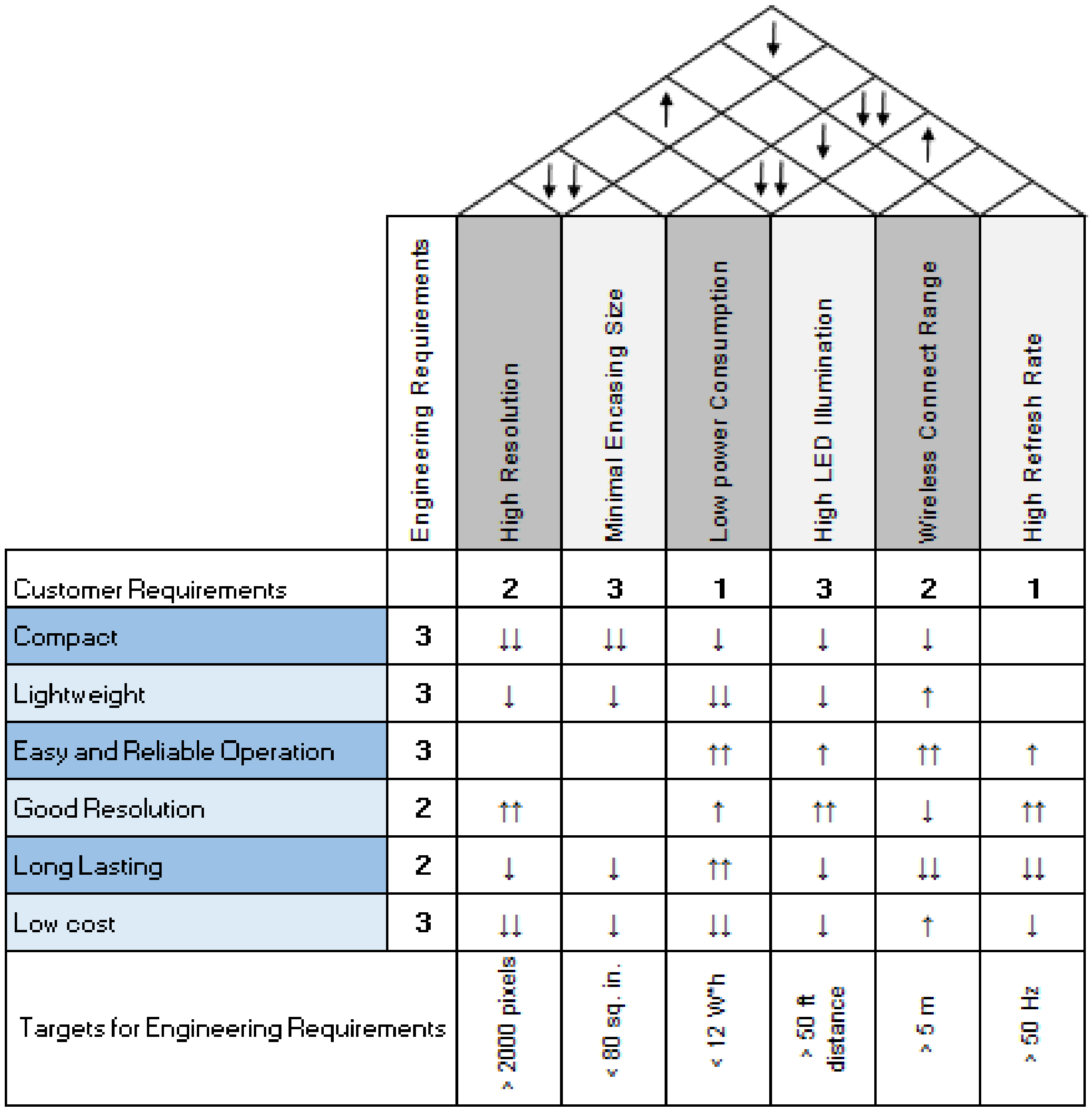
Figure 2.1 The original mockup for the device to help visualize its appearance.

The handle of the device is attached to the bottom of the rectangular encasing, intended for removable capability. The length of the handle was decided at 5 in. for the purpose of providing enough space for a hand to grasp, but not an unwieldy, excessive amount. The plan for the handle is to be cylindrical, but a silicon grip sleeve may also be added for extra handling comfort. Overall, the device is designed to meet the requirements of functionality, but to also save as much weight and size as possible. Given the requirements, maximum performance and minimal size will be achieved with this design, considering the spatial plans constructed, and the limits of sizing that is possible for reduction.

2.2 House of Quality

Most projects encountered in practical engineering are full of tradeoffs, decisions that require some form of compromise to achieve a particular task. For a project team, finding these tradeoffs is crucial for making a successful design, before anything is built. Such tradeoffs normally occur between the goal of a specific product feature, and the engineering ability to reach that goal. More generally, these tradeoffs mainly concern the expected operation of a device and the engineering constraints that are associated. A great way to visualize these tradeoffs is to compare the requirements set, or observed, by the customer and the requirements established by the engineers to design and manufacture the device. The House of Quality (HOQ) is the optimal chart to organize these comparisons. Teams use the House of Quality to fully list all of the requirements needed for the device to be successful, and to show the impact that each of the requirements has on each other. In this way, the House of Quality is an abstract layout of how the project goal can be accomplished.

The House of Quality is divided into several sections all connected by a single grid. The sections are as follows: Customer Requirements, Engineering Requirements, Targets for Engineering Requirements, and the Correlation Roof. Customer requirements are determined from the qualitative aspects of the device which is the desirable features and goals for the project to achieve. Engineering requirements are, conversely, the quantitative aspects of the project which determine the parameters of the device and set practical limits on these parameters. The targets for the engineering requirements are the goals for the theoretical parameters to reach in order for the device to operate successfully. Within the grid connecting these sections, simple correlation symbols are used to display a simple relation between every requirement of the device and, thereby, showing the various tradeoffs that will be faced. The completed diagram will then help the team sort any comprises early in project development so that any major decisions in the design process can be determined before a considerable amount of work is started. Below, in Figure 2.2, the House of Quality for this project is displayed, with every requirement and consideration that are encountered in designing this device.



Customer Requirements	Engineering Requirements	High Resolution	Minimal Encasing Size	Low power Consumption	High LED Illumination	Wireless Connect Range	High Refresh Rate
		2	3	1	3	2	1
Compact	3	↓↓	↓↓	↓	↓	↓	
Lightweight	3	↓	↓	↓↓	↓	↑	
Easy and Reliable Operation	3			↑↑	↑	↑↑	↑
Good Resolution	2	↑↑		↑	↑↑	↓	↑↑
Long Lasting	2	↓	↓	↑↑	↓	↓↓	↓↓
Low cost	3	↓↓	↓	↓↓	↓	↑	↓
Targets for Engineering Requirements		> 2000 pixels	< 80 sq. in.	< 12 W*h	> 50 ft distance	> 5 m	> 50 Hz

Figure 2.2 A house of quality that displays the tradeoffs between specifications.

LEGEND	
Classification	**Meaning**
↑↑	High Correlation - Positive
↑	Moderate Correlation - Positive
↓	Moderate Correlation - Negative
↓↓	High Correlation - Negative
1	Low Priority Rating

2	Moderate Priority Rating
3	High Priority Rating

The house of quality is a graphical representation of the tradeoffs encountered not only in terms of engineering, but also customer quality and pricing. In the left columns of the diagram is the listed characteristics that the customer desires from the device, while the column in the top of the diagram shows the engineering specifications that relate to those customer needs. The grid that results from the columns intersection is used to show the correlation between each customer need, and the device characteristic that is intended to achieve that. Arrows are used to show the magnitude of each correlation, and whether that correlation is a negative or positive relationship. The more arrows in each box, the higher the degree of influence the attributes have on each other. For example, a desired customer feature is to have a small, compact device. This characteristic is highly correlated to the number of LEDs, since this influences space on the board taken, and therefore the size of the board. As a result, two arrows are used to indicate the significance of this relationship. Also, the direction of the arrows is downward, meaning the effect of the LED number impedes the ability to keep the device compact. Meaning, as the number of LEDs increases, the less compact the device will become. This process is defined throughout the diagram, and provides a concise graphical summary of the goals to achieve, and how the engineering aspects will affect that. Using this table will help team members make optimal engineering choices in accordance with the objective and at the constraint of the final qualities.

Above the correlation matrix is the ‘roof’ of the House of Quality, which concerns only the relationship between engineering requirements, without regard to customer desires. Again, arrow symbols are used for type and magnitude of correlation between characteristics. These relationships are also important, but in a different sense, where tradeoffs are forced between the requirements of the same area. As decisions are made for the details of each requirements, the roof of the House of Quality shows how that decision may affect other technical aspects. This is different from the comparison within the center grid, since the correlation there concerns tradeoffs faced to produce desired operation and features, whereas the roof better shows the practical constraints of those requirements. By understanding the relationships in the roof, the full picture of not only requirements is clear, but also the initial limitations that are enforced by the engineering requirements themselves. The customer requirements are not compared to each other in this diagram, as those relations are not directly impacting of engineering design. In that sense, the roof contributes the last part needed to visualize the entire scope of the initial design requirements, and what choices must be made.

Below the grid on the House of Quality is the targets for engineering requirements. This data is quantitative to set a sort of tangible goal for each of the engineering requirements. Without these targets, part selection and design would become difficult to specify, since the rest of the house only provides abstract realizations of project goals. These targets are general values, but estimated as accurately as possible to narrow down part selection and set concrete objectives for the engineering team. The engineering team can then better realize the problem and design the product according to a particular value. For example, the target for the wireless connectivity range is at least 5 meters. At this distance, users can easily operate their device from across a room, which is sufficient for the intent of this device. This estimation was conducted without examining wireless modules so that the initial demand can be established before the module is designed to fit a parameter. Following this approach gives direction to the project, keeping the team from conducting any irrelevant work.

As far as presenting the final product, the engineering targets will also have to be achievable, so that proof of functionality can be shown. For demonstration, any of the engineering targets can be presented, but the most important are the wireless connectivity range, the encasing size, and the readable viewing distance. These are the fundamental demands for the device and are also viable for demonstration in prototyping and final review. The only requirements for demonstration are simple function tests and personal handling, nothing that needs special testing equipment or in-depth analysis. That way, customers can quickly determine the intent and capabilities of the device, as well as the project reviewers.

2.3 Goal

The overall goal of this project is to create a bright, consumer friendly LED sign that can be quickly and easily programmed and operated. We found in our research that most LED signs of this nature were made for retail use, leading to complicated user interfaces and functionality that wouldn’t be necessary for the consumer market. The general idea is that this should be a plug and play device without the need for anything besides a program to change the stored device messages and the device itself.

2.4 Motivation

The motivation for this product came from the sponsor’s desire to have a device that could allow people to signal in a vehicle without special hardware installed in their car. The sponsor wanted fully customizable messages as well as lights that were bright enough to see during the day. The product also had to be easy enough to use with one hand that it could be operated while in a moving vehicle.

During research for this project, the team came across many LED matrix implementations. While there were a lot of varied designs, it soon became clear that most of these designs were meant for store-front use. Many of these devices came with the functionality to load a message onto a USB stick using the proprietary software of the LED board. While this is very interesting functionality, it is only useful when applied to a stationary LED matrix that it would be easier to leave in place and move a USB drive to it.

Another common implementation was meant for stationary desktop usage, or as a home wall fixture. These were closer to the intended design, as they were smaller and more portable. They did not, however, have a handle or the ability to quick swap between multiple messages.

After seeing many available options and not finding any that fit the design we were interested in implementing, we became motivated to create a device that didn’t already exist on the market. We hope that should the device be made with enough dedication to it’s structural, electrical, and software integrity, that it could be introduced to a consumer market for a fair price. We would like to see people using our sign to signal at a sporting event, in a car, or on a hike.

3. Research and Part Selection

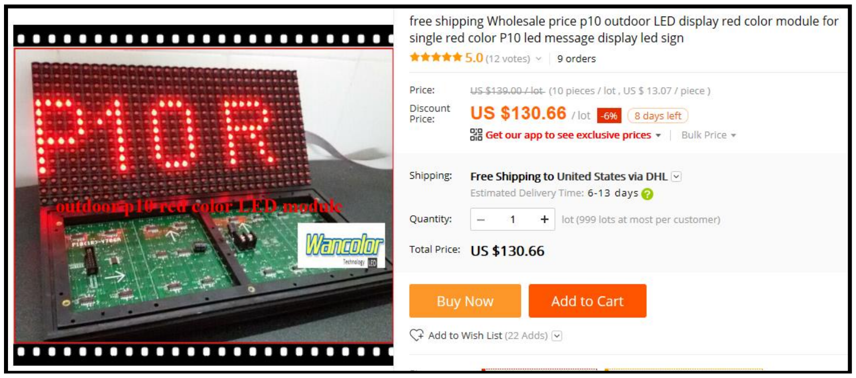
To create a functioning LED screen with multiple device compatibility and a rechargeable requires research in many different fields such as wireless Bluetooth connectivity, USB integration, battery charging, voltage boosting, and LED technology. By looking at the examples of other similar products found in the industry, we could get a general idea of the types of technologies we needed to look at to achieve our final product design. Many of the competing products, however, were aimed to be industry products instead of consumer products, which meant they had more functionality but a much more complicated interface. The research and part selection found here focusses on stripping down unnecessary consumer features and making the final product easy to use, while at the same time implementing a high-quality LED matrix.

3.1 Available Related Products

LED matrix technology today is not innovative by any means, with a multitude of applications and products available to consumers. Although most of these devices are designed with a consistent methodology, there are significant differences in the component selection and layout configuration that determine the performance capabilities and price of the final product. So, depending on the desired specifications, a wide range of features and qualities can be implemented or omitted from an LED screen device.

For the purpose of this particular project the goal is to reduce cost as much as possible for a hand-held, single color, variable brightness LED screen. This places a priority on cost efficient, but effective electronics and design, discarding excessive features that provide more than just the simplest functionality. To understand this technology and the conventions for digital screen design, research was conducted on items openly sourced for public observations. Such sources provided valuable insight on what electronics are preferable to use and what is the common design procedure. Additionally, these sources provided a template to improve upon for the application of this project, by using existing design as a basis for market cost and quality. Once this basis is decided, features and parts can be modified or interchanged as needed to reach the overall goal. The following examples are a few of the products that were reviewed:

3.1.1 Alien Express LED display


free shipping Wholesale price p10 outdoor LED display red color module for single red color P10 led message display led sign

5.0 (12 votes) | 9 orders

Price: ~~US $139.00 / lot~~ (10 pieces / lot , US $ 13.07 / piece)

Discount Price: US $130.66 / lot -6% 8 days left

Get our app to see exclusive prices | Bulk Price

Shipping: Free Shipping to United States via DHL

Estimated Delivery Time: 6-13 days

Quantity: 1 lot (999 lots at most per customer)

Total Price: US $130.66

Figure 3.1.1: LED display 16x32 pixels, front and back views shown with price.

The first example is a Raspberry Pi / Arduino compatible LED display made by an overseas vendor and sold through alien express. Not much documentation is provided, however, a unit was purchased by the project sponsor for better analysis. The device uses a series of shift registers to distribute the pixel logic to each individual column/row. There is also a multiplexer and transistor array that likely transfer current from the power supply to the connected pixels when they are selected, specifically the rows since the observable number of transistors could not support the numerous columns. Additionally on the PCB, there are arrows printed to indicate the flow of data and the correct orientation of the device. The 512 red LEDs (16 x 32) are controlled through a ribbon connection by the desired micro controller. Since documentation is not provided, no clarity is given on what the ribbon connection pins are, how specifically to load data into the device, and what maximum/minimum power requirements are necessary. The final asking price for this product, as shown in Figure 3.1.1, is listed conclusively as $130.66.

3.1.2 Adafruit LED display

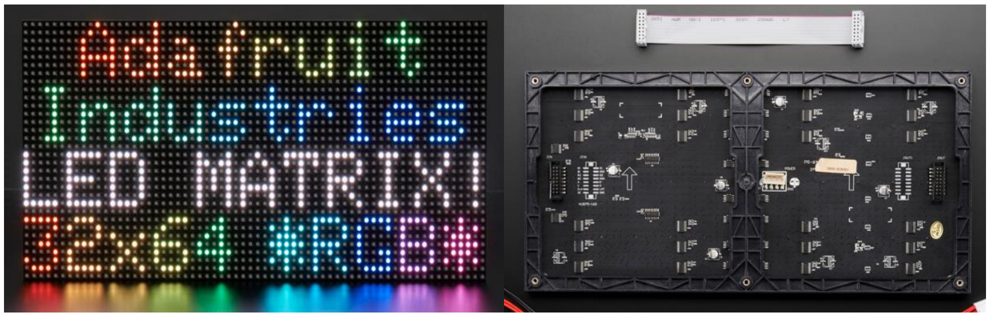


Figure 3.1.2: Adafruit Industries 32x64 LED display

Adafruit also offers programmable LED displays, such as the device shown above in Figure 3.1.2, but with added RGB color select for every LED. Similar to the product advertised by Alien Express, the device is compatible with Arduino microcontrollers through a provided ribbon cable. The data transferred from the microcontroller to the panel controls operation of ICs (MBI5024) that act like shift registers, only with a current limiting resistor port included for compact design. Additionally, counters are implemented on the PCB to continuously refresh the rows as data is sent to the registers for column control. The registers, synchronized with the rows, select the proper columns for illumination and determine the particular color of the LEDs. Adafruit Industries lists the product on their website for $79.99, with ribbon cable and guides for programming included. Like the Alien Express unit, orders do not include panel controller.

3.2 Relevant Technologies

Although relatively few features are to be exhibited from the final LED display, successful functionality of the device will require several important technologies. Decisions made concerning these technologies were based on minimal cost and sufficient, not excessive, capability. So while high precision components would appeal to project quality, they are unnecessary for completion and successful operation of the final product. Technologies that pertain to the operation of this device include wireless connectivity, portable power supply, LED lighting, PWM brightness control, and 3D printing. The following section reviews these technologies and how their operation contributes to the project.

3.2.1 Wireless Connectivity

Bluetooth technology offers the ability for devices to communicate over wireless channels. Communication occurs specifically between one “master” device and one or more “slave” devices. In order for devices to interact, the slave must first enter Discoverable Mode, where the slave transmits information to be detected by a master device. Once the master has identified the slave device, the two become “paired” or linked in memory to conveniently access communication between each other at any time without reconnecting. For security, pairing devices will usually also exchange passkeys, which constrain interaction to only the master and slave. The protocol operates by “hopping”, or rapidly changing frequencies to reach the necessary operating condition and to protect the communication from hacking. Protocols also differ depending on applications or data transfer types. For example, human interfaced devices (HID) such as keyboards, a mouse, or a Wii controller use the HID communication profile, whereas audio devices that are remote controlled will use the Audio/Video Remote Control profile (AVRCP). Over 20 communication profiles have been established, each dedicated for a specific data type or interaction process [4].

Devices using this technology normally communicate in short range scenarios (0-100m), but are classified by maximum range capabilities, which is dependent on power output. The classification is divided into three groups or “classes” with Class 1 for outputs of 20 dBm at 100 meters, Class 2 for outputs of 4 dBm at 10 meters, and Class 3 for 0 dBm output for the shortest range (< 1 meter) [4]. In addition to power, Bluetooth devices are also classified by their data throughput. Ratings in data transfer capabilities have increased as the technology has progressed over the years, from version 1.2 performing at around .7 Mbps, to version 3 rated at a maximum of 24 Mbps. Throughput, however, is costly in terms of power, which is especially paramount in battery powered devices. So the most recent Bluetooth technology, Bluetooth Smart, or otherwise named Bluetooth Low Energy (BLE), uses techniques to reduce power consumption, while still providing optimal performance in close range applications. This innovative technology, however, sacrifices throughput for low power measures, rated at 1 Mbps in most BLE modules. Range, however, is not comprised, with some BLE modules reaching over 100 m with the more advanced versions.

3.2.2 Power Supply

For the device to be portable, a battery must be provided to supply circuit power. Rechargeable batteries are preferable over disposable ones for the case of this project because Alkaline batteries require spring-loaded encasings that add unnecessary bulk, and rechargeable batteries are more cost efficient and convenient to test and operate. In terms of recharge ability, many types of batteries are available on the market to choose from, varying in types of chemical composition, voltage capacity, charge capacity, weight, size, and packaging. High performance batteries are ideal, but when considering cost, they are unnecessary. However, sufficient charge capacity for around 30 minutes of peak device usage is desired. Size and weight are also crucial for selection, since compact and lightweight device specifications is a goal of the project. Considering these factors and more, a particular type of battery needs to be selected.

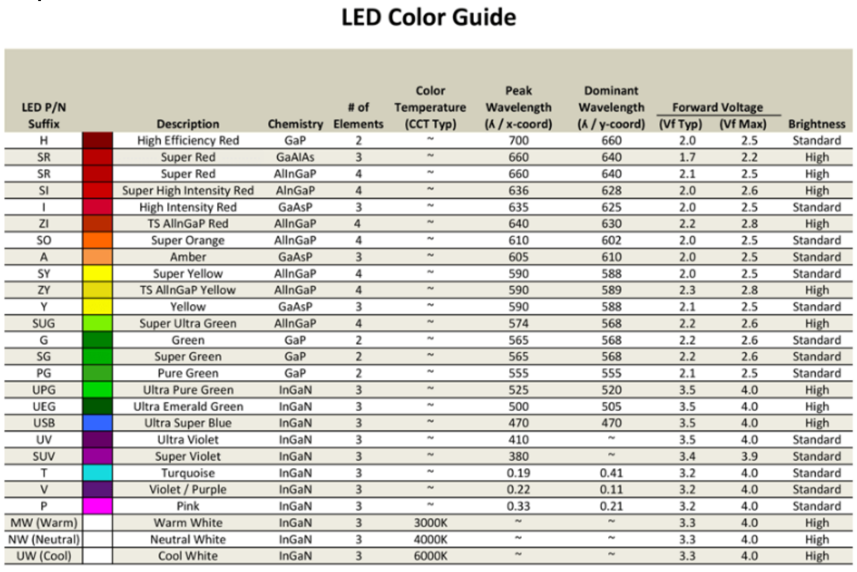
Upon searching for viable rechargeable power sources, the only rechargeable options were found to be Nickel-Cadmium (NiCd), Lead-Acid Gel, Nickel Metal Hydride (NiMH), and Lithium Ion (Li-ion) [Battery solutions]. NiCD and Li-ion batteries are for smaller device applications, and also non-toxic options, unlike Lead-Acid Gel and NiMH sources. NiCD batteries, normally offered in nominal voltages as A, AA, and AAA batteries, are acceptable for small devices, but are not packaged in small, thin profiles that are needed for compact design. Instead, Li-ion batteries proved the optimal choice since such batteries are packaged in thin configurations and generate necessary voltages for driving LEDs (2-4 volts). Additionally, Li-ion batteries have energy capacities up to 2-3 Amp*hours, which is plenty of storage considering the LED current draw which should peak at around 1 Amp maximum.

3.2.3 LED Illumination

The objective for the LEDs is to produce as much illumination intensity in the minimal bulb size possible. Depending on the chemistry of the LED, certain colors are emitted. For example, gallium arsenide is the semiconductor used to produce red LEDs [5]. Since the specification for the display requires only single-color LEDs, a decision had to be made on the specific color of the LEDs. Two important characteristics of colored LEDs are their light intensity (in candelas or lumens) and their current consumption. By selecting a particular color of LED, power consumption of the screen could be significantly reduced, while also considering overall brightness production. As a common rule of thumb, the shorter the wavelength of the visible color, the brighter the light will appear. The power consumption, however, tends to be costlier with brighter color. Figures 3.2.3.1 shows these trends with some commonly manufactured LED colors. Such considerations are important for part selection as the decision presents a tradeoff between device endurance and illumination capability.

Another important characteristic for LED selection is packaging. LEDs are offered in various shapes, sizes, and mounting, depending on the application. For the purpose of this project, surface-mount LEDs with dimensions as small as a fraction of an inch and needed. This is because the goal is to fit as many as possible in a closed, PCB integrated, encasing, with the illumination still effectively bright. From this, LEDs of different sizes and types were compared. LEDs are manufactured in many different shapes and sizes, depending on the application and design. These differences are classified by chip orientation, lens type, connection type, and packaging size. Orientation of the LED is mainly concerned with how the LED is to be mounted and where the illumination needs to be concentrated. For inverted or angled orientation, LEDs are produced as 90 degree configurations and reverse mounting. This allows for the contacts of the LEDs to be hidden, and for the ability to find clever methods of placing them on PCBs for unique applications. This also correlates with the next important attribute of LEDs, which is the connection type. Typically, this is more relevant with leaded LEDs, where the wires protrude outward from the diode itself. The wires are bent and clipped in many ways so designers can utilize the most optimal style for particular placement on the PCB. For example, gull wing LEDs have leads that bend downward, towards the PCB, and then flattened at the bottom to form convenient legs for the chip to rest on. Another type of lead is the z-bend, where the leads are inverted to that of the gull wing configuration, mainly used for mounting on the opposing side of the PCB, and emitting through the other side, hiding the wires in the back. Aside from all these wired products, and all other alternatives, SMD LEDs are also made without any outward leads at all. These are called no-lead LEDs and they use metallic contacts embedded on the packaging of the device, removing the needless space requirement of the leaded LEDs. The contacts of these LEDs are located below the emitter, making conventional soldering with a needle-tip iron difficult, but hot-air equipment resolves this issue, which can be in the form of an industrial oven or a hand-held blower. LEDs that are produced without wire leads also allow for smaller designs, which can be reduced to even less than a millimeter in length/width. Accordingly, more lights can be assembled in each space at a much higher efficiency and without the concern for faulty or broken wires.

An additional attribute that makes LEDs unique is the type of lens for which the light is emitted through. When LEDs are produced, the emitting diode is covered in a hard epoxy resin protecting the component and helping to optimize the lighting capability. This resin is considered the lens for the light. Various types of lenses can be specified based on two different factors, opacity and shape. While opacity may reduce the illumination effect that is not the complete intent for these types of lenses normally. By using a lens of a different composition, which may or may not be opaquer, the light from the LED may diffuse differently, resulting in illumination that may increase the radial spread of the light, or in an illumination that is filtered for eye protection for a more comfortable feel. This attribute has little effect in the technical sense for designing the PCB, but plays a major role when considering overall light quality of the device, especially when the effect is multiplied by over 2000 lights. The types of lens compositions that are offered include 'water clear' (transparent), diffused, and tinted. The water clear lens allows the most illumination as the resin is made completely transparent, providing no resistance to the emission of the light. The diffused lens provides a 'cloudy' type of filter for the light, which does decrease the light intensity of the LED, but spreads the focus of the light so that more area is covered evenly instead of just a narrow target. If producing a particular color of light without altering the chemistry of the LED is necessary, then a tinting lens is the ideal choice. Tinting is done by placing a colored or shaded film on the epoxy lens, which masks the intrinsic color of the light and forces the emission of light that matches the color of the film [11]. By selecting LEDs with a particular type of lens and light diffusion, the quality of the light can be maximized across the entire device as desired. While maximum emission and efficiency is a goal for this project, transparent LED lenses may result in harsh lighting on the human eye, since LEDs naturally tend to be incredibly bright considering their size and power consumption. More comfortable lightning may be achieved with slightly diffused lenses that provide the brightness necessary and scatter light emission evenly for an attractive display. As for the coloring of the LEDs, diode chemistry is not a major concern for selection, so colored films may be the advantageous option if such LEDs are more cheaply available. Combining these qualities, the search spectrum for LEDs was still broad, but at least there remains the ability to select the most accessible LEDs that provided the manageable performance needed.



LED Color Guide

LED P/N Suffix		Description	Chemistry	# of Elements	Color Temperature (CCT Typ)	Peak Wavelength (ʎ / x-coord)	Dominant Wavelength (ʎ / y-coord)	Forward Voltage (Vf Typ)	Forward Voltage (Vf Max)	Brightness
H	🟥	High Efficiency Red	GaP	2	~	700	660	2.0	2.5	Standard
SR	🟥	Super Red	GaAIAs	3	~	660	640	1.7	2.2	High
SR	🟥	Super Red	AllnGaP	4	~	660	640	2.1	2.5	High
SI	🟥	Super High Intensity Red	AlnGaP	4	~	636	628	2.0	2.6	High
I	🟥	High Intensity Red	GaAsP	3	~	635	625	2.0	2.5	Standard
ZI	🟥	TS AllnGaP Red	AllnGaP	4	~	640	630	2.2	2.8	High
SO	🟧	Super Orange	AllnGaP	4	~	610	602	2.0	2.5	Standard
A	🟧	Amber	GaAsP	3	~	605	610	2.0	2.5	Standard
SY	🟨	Super Yellow	AllnGaP	4	~	590	588	2.0	2.5	Standard
ZY	🟨	TS AllnGaP Yellow	AllnGaP	4	~	590	589	2.3	2.8	High
Y	🟨	Yellow	GaAsP	3	~	590	588	2.1	2.5	Standard
SUG	🟩	Super Ultra Green	AllnGaP	4	~	574	568	2.2	2.6	High
G	🟩	Green	GaP	2	~	565	568	2.2	2.6	Standard
SG	🟩	Super Green	GaP	2	~	565	568	2.2	2.6	Standard
PG	🟩	Pure Green	GaP	2	~	555	555	2.1	2.5	Standard
UPG	🟩	Ultra Pure Green	InGaN	3	~	525	520	3.5	4.0	High
UEG	🟩	Ultra Emerald Green	InGaN	3	~	500	505	3.5	4.0	High
USB	🟦	Ultra Super Blue	InGaN	3	~	470	470	3.5	4.0	High
UV	🟪	Ultra Violet	InGaN	3	~	410	~	3.5	4.0	Standard
SUV	🟪	Super Violet	InGaN	3	~	380	~	3.4	3.9	Standard
T	🟦	Turquoise	InGaN	3	~	0.19	0.41	3.2	4.0	Standard
V	🟪	Violet / Purple	InGaN	3	~	0.22	0.11	3.2	4.0	Standard
P	🟪	Pink	InGaN	3	~	0.33	0.21	3.2	4.0	Standard
MW (Warm)	⬜	Warm White	InGaN	3	3000K	~	~	3.3	4.0	High
NW (Neutral)	⬜	Neutral White	InGaN	3	4000K	~	~	3.3	4.0	High
UW (Cool)	⬜	Cool White	InGaN	3	6000K	~	~	3.3	4.0	High

Figure 3.2.3 LED color comparison based on intensity and voltage needed. Provided by ITW Lumex.

In considering PCB parameters, however, the important aspect of the LED lens is the shape and size of the lens. The reason for this is that the shape of the lens many times affects the dimensions of the LED. Such shapes are offered in flat or domed configurations. Dome shaped lenses are great for radiating light in a larger spread of surrounding directions, but usually require more height considerations in design, whereas flat lenses are more compact. Flat lenses, however, may not be optimal for tight spacing applications on the face of the PCB, as the circular profile of the dome lacks the space consuming corners of the rectangular, flat lens. Also, after extensively searching online distributors, flat lens SMD LEDs tend to be in higher production and less limited than the dome-shaped counterparts. This will not only result in cheaper prices for the flat lens variations, but also broadens the selection which gives more flexibility for the various other characteristics mentioned.

Based on the researched discussed, there is clearly a vast range of products and specifications to consider, even without including electrical properties. An ideal LED selection should allow for a display that provides exceptional performance while reasonable parametrically for construction. Illumination intensity is important since higher brightness capability results in higher light producing efficiency, but, at the same time, the intent is not to harm the eyes of the user. Also, light spread is important for even illumination in the direction of the target, which, as previously mentioned, can be achieved with the shape of the LED lens, or the composition of the resin for diffusion. In this project, though, the shape of the lens has a higher degree of influence on the design since that affects the LED height consideration, whereas the lens composition only influences the quality of the illumination. Therefore, compromising the diffusion of the LEDs with lens shape, rather than composition, is preferable. These considerations, of course, are limited to availability and cost, which is why selection was somewhat flexible in terms of LED characteristics. If LEDs are found that can support the essential lighting and electrical requirements, the options were left open to many considerations.

3.2.4 Dimming

When considering brightness control of lighting, the conventional method is to adjust the current or power supply to the light. The theory behind this technique is that by modulating the current input through the light, the lumen intensity of the light should change accordingly, since the operation of the light is a function of the current supply given a nominal voltage. Commonly referred to as analog dimming, this is the fundamental and intuitive method for dimming. This procedure, however, is undesirable in LEDs applications. The issue with analog modulation of brightness is the LED's current to illumination non-linear relationship. As the current changes, the intensity doesn't change proportionally, making precise linear adjusting difficult. LEDs also have narrow threshold limits for current driven operation, meaning that the range of adjusting is severely limited. Combined, these factors restrict lighting designers to few increments of dimming for a sometimes unpredictable or tuning-intensive result. In addition to these drawbacks, current modulation also affects the color "warmth", or quality. So as the brightness from analog adjusting changes, the color wavelength may not remain consistent.

Instead, current LED circuit designs utilize PWM outputs to control the intensity of the lighting. PWM is the method of continuously toggling an LED and adjusting the ‘duty cycle’, or the amount of time that the LED is ON versus OFF. When this is conducted at a high enough frequency, the human eye can’t distinguish from pulses, and thereby seems as if the LED is continuously on. Dimming is accomplished by reducing the period for which the LED is ON versus OFF, and inversely for brightening.

This solves the problem encountered with analog adjusting as the voltage and current remain constant with PWM, since only the period for a logic high/low is manipulated, not the source of power. Also, the relationship between PWM duty cycle and illumination intensity is entirely linear, allowing for easier and more precise tuning of LED brightness. With just implementation of a programmable timer that generates output at specified intervals, using PWM for LEDs is a generally straightforward approach.

3.3 Design Approach

As mentioned previously, LED matrix devices are not, by any means, new to the electronics industry. Accordingly, countless products with such technology have been devised and examined for many different applications. This provides an excellent foundation of where to begin with designing such a device, and reliable guidelines to follow along the way. Since simplified LED matrixes are not very commercialized, templates and tutorials are mainly accessible from individual designers who release their knowledge and projects as open source information. While such designs may not be reliable or applicable to this project, they provide a platform to work from and useful insight to understand the technology.

The initial concept to fully grasp when considering LED matrices is activating individual pixels. Matrices are constructed with LEDs by using common pins for all cathodes and anodes of the LEDs. The result is set of selectable rows and columns that control the entire matrix. To activate a single LED, or pixel, the correct voltage drop must be applied to the row and column that correspond with that LED. This means that the cathode is lower potential than the anode, usually with the cathode grounded and the anode driven to a high voltage. When users need multiple pixels in a certain row or column to be driven, though, this procedure becomes ineffective because for a particular pixel to be active, the entire row and column associated with that pixel need to be connected. So, if more certain pixels nearby are activated, there may be an excess in LEDs on at once than desired. To solve this problem, designers tend activate only one or few rows/columns at a time, and cycle through the data of the other rows/columns to active the rest of the panel. Such a scheme is described by an Adafruit Industries contributor, Phillip Burgess, in his tutorial reviewing LED matrix design. He explains that by continuously “refreshing” the display, or sequentially activating the rows while the data is pushed through the columns, faster than the human eye can detect, the panel can successfully display the data in a manner where the intended display graphic appears static. This involves not only coordination of the data between the rows and columns, but also a synchronization of cycling frequencies between lines, so that no bits of data are missed [9].

Now considering hardware, components are needed to properly select and deselect the rows and columns. Current that flows through the LEDs is provided from a source or “driver” and flows into the destination component called the “sink”. The anode lines should be connected to the driver side of the circuit with the high voltage potential, while the cathode lines are connected to the lower potential where current transfers to the sink, producing the effective voltage drop necessary for LED activation. To achieve this, components such as power source switches and shift registers are generally employed. Switching components are necessary because they convert a signal from a controller to connected or disconnected output, which is essentially the goal for line selection. As for the remaining lines, shift registers are used since strings of data can be transmitted and pushed quickly to the correct pins, especially when the design requires a vast amount of pins from microcontroller. In that case, a series of shift registers may be connected, forming even longer continuous strings of data that are capable of reaching every one of the remaining lines, with only a few pins from the microcontroller to supply the data. A simplified LED matrix design, from tutorial by Sohil Patel was observed for reference design.

There, only 8 pins are needed for the column control, so those lines were directly connected to the microcontroller. In applications with many more columns, however, shift registers are required to reduce the number of needed pins on the controller. For that reason, shift registers are typically assigned to control the columns, while the source switches handle manipulation of the rows.

From observing many additional schemes and considering electrical characteristics of the circuit, the decision was made to connect the anode LED leads to the drivers of the circuit, and to connect the cathode leads to the sinks. The main reason for the decision is that power switching devices are ideal devices for handling the forward voltage and current. Accordingly, the switches are assigned to connect and disconnect the power source to the anodes, and drive the current through the LEDs. On the sinking side, the cathodes are to be connected to the shift registers. This way the output of the registers only need to provide a minimal amount of voltage to correctly modulate the lines, which evidently most shift registers are limited in terms of potential capability. Finding shift registers to provide any more voltage output or power throughput has proven difficult, with such discovered registers listed high in price.

3.4 Part Selection

Selecting the correct parts for a comprehensive electrical/computer engineering project is imperative not only for physical and electrical characteristics, but also for software design. If the hardware is chosen such that implementation is understandable and compact, then all project members can more easily recognize the objective of their contribution. This, in turn aids the process in moving smoothly and keeps communication progressive in the group rather than constantly backtracking and tracing previous designs. Aside from effectiveness among the team, proper part selection also makes for efficient power distribution along with overall improvement in quality. The ensuing section reviews the parts selected, specifically circuitry components that contribute to the functionality of the project.

3.4.1 Microcontroller

When selecting a microcontroller, we had two major options on the table: an Arduino development kit, or a Texas Instruments board. Other options, such as the Raspberry Pi, were ignored due to difficult integration on a PCB or unfamiliarity with the product. Between these two companies, we selected a few viable options from each catalog.

Table 3.4.1

Part	Manufacturer	Freq. (MHz)	Flash (KB)	RAM (KB)	Price ($)
ATmega328P	Atmel	20	32	2	2.01
ATmega2560	Atmel	16	256	8	11.85
MSP430FR5962	Texas Instruments	16	128	8	2.35
MSP432P401R	Texas Instruments	48	256	64	8.07

Of the chips in the table, it is important to note that all manage to meet the minimum frequency requirements for a non-visible refresh rate. The flash memory will story the program that will run the device. This program will have a large amount of stored data for the display of each character that can be input to the device. For that reason, the larger the flash memory capacity, the more characters and character size the device can support. Immediately, the ATmega2560 and the MSP432P401R stand out as top choices.

Between these two chips, the MSP432P401R has significantly more RAM to work with. This is useful for the functional parts of the program, but as the large core data (the different characters and character sizes) will be stored in flash memory and only loaded into the program as necessary, RAM is not a necessary consideration so long as it is above a certain threshold. It is worth mentioning that the ATmega328P does not meet this threshold, and is therefore not a consideration as a viable microcontroller.

Another important consideration that is not on the table is the number of pin connections available on each chip. Of the three chips that are still viable, all of them have 60 or more connections, some of which are analog and more of which are digital pins. The most notable pin type for these chips is the availability of PWM, a feature all of these chips support. This feature allows us to dim the lights based on a high-speed flicker effect that, and is a core part of the device design.

The final major consideration is programmer familiarity with the software available to code these devices. While the Texas Instrument chips support C and C++, the Arduino chips also feature their own Arduino language. This language comes with support for common microcontroller functionality such as serial data transmission/reception, simplified interrupt control, and built in bit read/write bit functions. This functionality greatly simplifies the programming task and was a major factor in why the ATmega2560 was selected as the microcontroller for the device.

3.4.2 LEDs

As mentioned previously, the LEDs selected for this project must be small enough to fit within a small profile, while still providing sufficient illumination for visibility. Size not only includes the packaging, but the type of leads as well. So to reduce excessive board space, LEDs with wire leads are not preferred. These wires tend to protrude outward from the LED package, costing valuable board area. Instead, surface mount LEDs with integrated contacts are much more space efficient, as the solder contacts are embedded with the outer surface of the package. This removes considerations needed for extended leads and solder pad space, leaving only the area assignment to only the package itself.

The dimensions of the package are intended for less than 5 mm in length, and less than 2 mm in width. LEDs are offered in nominal size, and based on availability and desired dimensions, 1212 LEDs are the optimal choice. 1212 refers to the dimensions in terms of inches, correspond to a length of .12 inches and a width of .12 inches. In millimeters, these dimensions convert to about 3.2 mm long and 3.2 mm wide, meeting the desired criteria. Such packages are also classified by packaging height, which is not as crucial to consider as lateral measurements, but compact design is the reoccurring goal. From this, and again based on availability, the choice was made to choose a height of 1 millimeter, which is typically flat in design. Dome bulbs are also offered but such bulbs tend to take much more vertical space.

As for illumination intensity, color emission of an LED is yet another important aspect to consider. From the previous section containing background on LED wavelength and brightness, colors in the longer wavelength side of the EM spectrum have lower illumination potentials, but are much more efficient with power consumption, versus the much brighter colors near the UV spectrum. Although brightness is important to remember during the selection process, current consumption, translating to power usage, is imperative to the electronics on the panel for the functionality of the device. Also, since brightness is mostly related to the amount of forward current, LED colors that are slightly lower in intensity may be driven to produce marginally higher illumination than rated in the datasheet by simply adjusting the source to provide more current. Additionally, LEDs of even the dimmest color are able to deliver effective brightness at reasonable viewing distances, making illumination quality more trivial of a characteristic during lighting selection.

Based on the considerations above, the optimal selection of LEDs for the sake of the project is red, 1212, flat lens lights. Reducing the size any further of the device package would decrease the illumination capability of the panel overly to the point where the display message would be indistinguishable across a room. Also, choosing red for the color of the LEDs saves a substantial amount of device power after realizing the potential current draw with over 2000 of such LEDs. Any brighter color would drain the supplied battery significantly faster, detracting from the effectiveness of the product.

Table 3.4.2

Part	Manufacturer	Size (in.)	Typ. Volt	Typ. Current	Price per 3,000
SMLLX15HC	Lumex	.12 x .12	2.0 V	20 mA	$420.00
GR PSLR31	OSRAM	.12 x .12	6.75 V	150 mA	$534.60
GF CSSPM1	OSRAM	.12 x .12	1.85 V	350 mA	$3432.00
5050-R1200	Superbrightleds	.2 x .2	2.0 V	20 mA	$1110.00

Above in Table 3.4.2, four different 1212 SMD LED selections are compared. Selection for that specific size of LED in the red color turned out to be very limited, with only a handful of options to choose from. The options listed were the closest in terms of operating specifications and price. The LEDs manufactured by the company OSRAM are higher power LEDs, producing more luminous intensity for more industrial applications. This boost in performance is evidently reflected in the price comparison, in the most expensive case requiring more than $3,000 for an approximate number of LEDs necessary for the project. Such intensity is not necessary for this application and is too expensive for the scope of this project. Also, such LEDs require relatively high voltage and current draws, which, in turn, requires more expensive supporting components to handle that power consumption, and the price continues to rise. Another option is the LEDs from superbright.com, but these are also expensive for purchase and too large for PCB placement. From this, the LEDs provided from Lumex proved to be the best choice. They are relatively affordable for the amount needed, and they consume only about 40 mW per LED, which is small enough for a portable battery to supply for a reasonable duration of operation.

3.4.3 Current Sink – Shift registers

When attempting to control the many rows and columns of the LED matrix, it can be difficult to supply the number of inputs necessary to directly toggle every row and column, since processors have a limited number of output pins. Instead, shift registers are used to reach all data lines required for data write. Shift registers are needed because they can receive many bits of data, with only the input of one serial line, one clock line, and an output enable line. The data is then all pushed out in parallel to every output at the same time, controlling every column of the matrix in unison. This reduces the number of pins needed to control an LED matrix immensely, as cascaded registers only need the three data lines to transmit all of the data to as many lines as necessary. In this design scheme, like many other open source designs, shift registers will also be utilized as the current sink, which means the current from the LEDs will be driven into the outputs of the shift register. Both qualities are significant in selecting the registers because these characteristics will determine the number of LEDs handled by each register, as well as the amount of maximum current that travel through the component.

After calculating the typical and maximum currents expected to flow through each column of the matrix to the registers, the ratings for peak current capability of the registers was determined to be around 20 mA. Unlike the drivers supplying power to the rows, where the LEDs are in parallel, the current through the columns will be only the total current through one LED at most, meaning the maximum current could either be 0 mA or 20 mA depending on the state of the particular pixel, ON or OFF. Therefore, if the register is capable of receiving 20 mA at any given period, the component is sufficient for the sinking task.

Evidently, a vast selection of products for supplying LED sinking operation was available, all with different features, application types, data rates, and more. The options were eventually narrowed, knowing that RGB control was irrelevant to our device, high rates of data transfer are unnecessary, and automatic error correction is not needed. The choices left, although reduced in functionality, provided the specific device needs at hand, at a much lower cost than the more complex variations.

Table 3.4.3

Part	**Manufacturer**	**Outputs**	**Min. Supply Voltage**	**Max. Current**	**Price per unit ($)**
SN74HC595	Texas Instruments	8	2 V	35 mA	0.52
MAX6969	Maxim Integrated	16	3 V	60 mA	6.89
MM5450	Micrel	35	4.75 V	15 mA	6.18
TLC6C598	Texas Instruments	8	3 V	50 mA	.91

Table 3.4.3 shows a comparison of the various current sink devices considered for the final product. An important note to make is the number of outputs each component has and the price associated. More outputs translate to fewer components needed to support the number of column lines controlled, but also results in a higher unit price. Realizing this, a calculation was necessary to analyze the tradeoff, with how much the total cost would equate to considering the number of components needed. After crunching the numbers, the results easily favored the cheaper, less output registers. This is from that fact that registers like the SN74HC595 are very cheap from their simplistic design. Less outputs are available, but considering the space availability on the PCB, implementing the numerous registers is surely feasible. Another feature of the more expensive registers is that they include pins for integrated PWM and resistive current control.

The simpler registers do not explicitly include these capabilities, but they are luxuries that are easily solved with software and a few additional low-cost components. This reasoning was determined by majority from the monetary cost review. The design would become more simplified and straightforward to implement with more extravagant parts, but the price is just too large to compensate for. Also, the SN74HC595 requires a much lower supply voltage than the others, reducing the overall power consumption which is a major advantage in a battery reliant device. At just a minimum requirement of 2V, the power reduction is staggering relative to the other parts. The last parameter considered was the maximum current sink capability. If the LEDs sink more current than the register can handle, then the registers could permanently fail to where they become useless. Referring to the design of the LED matrix and based on the LEDs selected, only a maximum current of 20 mA can be transferred to the sink components. Accordingly, every component compared was picked based on the ability for the component to satisfy this, so each is capable of this standard. Therefore, the components that exhibit higher current sink maximums provided no extra advantages to others. Considering all these aspects, the SN74HC595 is the best choice because it provides all of the necessary functionality, at a significantly lower implementation price than all the other options. Also, even though a separate PWM is not included with the device, the output enable (OE) pin can be configured to provide PWM functionality. This is accomplished by toggling this pin at a high frequency, which continuously enables and disables the outputs, flashing the outputs just like a PWM module would. In that instance, the SN74HC595 would prove almost equal in functionality to the others, making this decision even more apparent.

3.4.4 Battery

The portable power source for this project needs to supply a voltage of at least 3 volts and must be rechargeable. This is determined from the minimum turn on voltage for the various circuitry and LEDs. Sources, or batteries, are offered in various chemistries and packaging. To reduce bulk where the battery will reside, slim packages are preferred with length and width being more negotiable, only restricted by the abundant space PBC space at the rear. After searching various batteries, Lithium Ion (Li-ion) was the only chemistry that was found in slim, rectangular form configurations. Nickel Cadmium batteries are sold in thick cylinder forms that are cumbersome for design, and would result in a larger encasing. Other chemistries are used for more energy costly applications, offered in packages too large and heavy. Although Li-ion batteries are expensive, they are sold in variations with suitable voltage and desirable dimensions.

As with many parts, selecting batteries depends partially on availability. Manufacturers of Li-ion batteries only really make a diverse range of products rated for 3.7 volts, which seems to be the optimal operating level of that chemistry. Additional voltages are produced, but are greatly more limited in selection. 3.7 volts, though, satisfies the required voltage level of at least 3 volts. Once voltage was determined, the next decision to make dealt with the most crucial characteristic for batteries, storage ability. For electronics batteries, units of batteries are given in mAh. For 3.7 volt Li-ion batteries, this specification normally ranges between around 100 mAh and 5000 mAh. With every LED in the panel on, the peak current that could be reached is approximately 1.5 A, taking also into account contributions from all other components. For this extreme case, a storage rating to retain the current for 1 hour would calculate to 1 hr * 1500 mA = 1500mAh. Since the goal for the device is to retain about 2-3 hours of charge in use, a rating of 1500 – 3000 mAh would be ideal.

Table 3.4.4

Part	Manufacturer	L" X W"	Height	Volt.	Charge Capacity	Price
ID: 258	Adafruit	2.4 X 1.3	0.2	3.7 V	1200 mAh	9.95
ID: 328	Adafruit	2.55 X 2	0.3	3.7 V	2500 mAh	14.95
ID: 2011	Adafruit	2.4 X 1.4	0.3	3.7 V	2000 mAh	12.50
932MIKROE	MikroElektronika	2.4 X 1.72	0.264	3.7 V	2000 mAh	13.90
UBP001	Ultralife	2.13 X 1.42	0.43	3.7 V	1750 mAh	15.70

Adafruit offers a line of Li-ion batteries that are rated at such values, and within a reasonable price range ($10 - $15). Such batteries are listed in the comparison table, Table 3.4.4. All batteries in the table are Li-ion since this was determined as the only viable chemistry for the project. Adafruit occupies the bulk of manufacturers in the table because they offer a wide range of Li-ion batteries of all different sizes, capacities and pricing. Sellers like mouser and digikey were also examined, but the selection was more limited, with unappealing dimensions and storage specs for what was available. At Adafruit, two batteries 3.7 volt batteries are of interest. One is rated at 2000 mAh and listed at a price of $12.50, while the other, rated at 2500 mAh, is priced at $14.95. At a price difference of only a little over $2, the 3.7 V 2500 mAh Li-ion battery is the optimal choice. While 2000 mAh would have been sufficient for the application, 2500 mAh is a safer choice as full circuit implementation may exceed our expectations in terms of current consumption, which could deplete the smaller battery in a fraction of the desired time. Also, considering the affordable price of these batteries, fitting these batteries into the budget is easily manageable. All of the other batteries have lower charge storage ratings and also are similar in price, which presents the same decision as stated before for the competitive Adafruit option. Also, the dimensions of every battery complies with the restriction of the encasing, with a maximum width tolerance of 6", maximum length of 11", and a maximum height of .5".

3.4.5 Screen Refresh Components

Referring to the methodology of the design, the data transmitted to the LED panel must be continuously refreshed at a sufficient frequency to display the graphics as a constant image. For this to occur, a circuit must be implemented that sequentially activates the series of lines opposing the shift register data. There are many ways of accomplishing this, but a constraint associated with most ideas is the limited number of pins on the microcontroller. To use the minimal number of pins and cycle through lines of the matrix, there are a couple of approaches.

The initial scheme for automatic refreshing, as seen with Mike Szczys reference design, was to use a counter with a decoder to increment select lines. Output lines from the counter would connect to the input of the decoder, with the outputs of the decoder to activate the selected lines. This works from the fact that counters output incrementing binary values, which is decoded at the decoder input to select a specific output to activate. If the counter counts from 0 to a final value of, then the decoder, using 'n' inputs, would select one output from 0 to . The only pins necessary for this idea would be a clock line and a voltage supply. The issue with this idea, however, is that a counter and decoder would be required for each row of the matrix, taking up a considerable amount of board space and resulting in higher spending.

Instead, a ring counter will be produced for the purpose of counting through the rows. A ring counter is a device that carries a single logic '1' through each of its outputs, and then starts back at the beginning. The advantage of this device is that less components are necessary and the overall design becomes simpler. Connections from the counter could transfer directly to each of the rows, and the counting would only depend on a single initialization step and the clock signal to keep the process going. Ring counters, however, are not sold as physical devices, because they are typically constructed from simple reconfigurations of shift registers. By connecting the output signal straight to the input, the shift register will continue to cycle the same data through, a logic '1' with all other '0' for example. With the clock running continuously on the device, the parallel outputs would effectively transfer the '1' along the outputs, incrementing through the outputs as desired.

Table 3.4.5

Parts – Texas Instruments	**No. of Inputs**	**No. of Parts**	**Price ($)**
Shift Register (74HC595)	3	4	0.52
Counter/Decoder(SN74LV393ADR/CD74HC237)	3	8	1.50

Many types of shift registers are available from manufacturers for various applications, such as providing secondary storage, inverting outputs, providing many input selects, and supplying data at high frequency. For the purpose of the project, a standard, serial-in parallel-out shift register is sufficient for the task needed, only with a serial output for data feedback. The serial output will be connected back to the input for data cycling, and the parallel outputs will be connected to the LED leads. The part chosen to satisfy such characteristics, at the most affordable price, is the TI CD74HC595 8-bit shift register. Telling by the part number, this part is simply a variation of the 74HC595, one of TI's most basic and widely used dual-inline mounting shift registers. The variation provides packaging viable for surface mounting and low-profile implementation. This is also advantageous for application from prototype to final project from the similarity of the through hole and SMD devices, making scaling up of the design consistent and feasible.

Not only does this make the refresh circuit design easier, but the use of the 74HC595 register keeps the entire PCB consistent as these are also used for the current sinks. This means that more of that component can be ordered for better use of bulk pricing, rather than purchasing few instances of various parts. Aside from that examination, the single unit price is also much cheaper than the counter/decoder design anyway, shown in Table 3.4.5. Consequently, the choice was promptly made to use the shift registers for the automatic refresh circuit. The implementation only requires 1 initialization step, and is both simpler and cheaper to include in our design. Alternative shift registers are also viable, however the competitive pricing analyzed previously with the current sink devices already proves how desirable the 74HC595 is over any other option.

3.4.6 Source Switching Device

The circuitry used for refreshing the LED screen will provide the logic decisions for which lines will be selected, but are not capable of supplying the necessary power to illuminate any of the LEDs. Consequently, the decision was made to use a type of electronic switch, which will only supply power to the screen when activated by the ring counter. Transistors are perfect since they are normally used for supplying power in switching applications. For transistors to supply power, they need to acquire a voltage at the base terminal, which is provided from the selection of the controller. Once this happens, current flows from the collector to the emitter, in bipolar junction transistors (BJTs). The current flow through the transistor is current, and is directly taken from the source, which is the ideal characteristic for turning one the LEDs.

As mentioned by electronics enthusiast Oscar Liang, in his tutorial explaining the difference between BJT and MOSFET transistors, BJT's are more commonly produced for small scale electronics, whereas MOSFETS are typically utilized for higher power dissipation applications[3]. The reason for this is behind the various electrical characteristics of the transistors. But, in short, BJTs tend to have faster switching periods, and require a much lower threshold to turn ON, for currently mass-produced devices.

Table 3.4.6

Part	Manufacturer	No. Outputs	Technology	Max. Current	Price ($)
MPQ3725	Central Semiconductor	8	BJT NPN	1 A	4.42
ULN2064	ST	8	Darlington NPN	1.5 A	2.71
MIC2981	Micrel	8	BJT Driver	0.5 A	2.42
TBD62083	Toshiba	8	DMOS	.5 A	0.65
TLC59213	Texas Instruments	8	Darlington BJT	.5 A	1.75

Table 3.4.6 shows some of the transistor arrays, both BJT and MOSFET, compared for sourcing the matrix. BJTs have desirable characteristics, as mention before, especially for the application towards this project. If only one transistor were to be utilized for an entire row of the LED matrix, the maximum current that could be drawn is over 1.2 A. From this, choosing transistors that could accommodate such a rating is ideal, but, as shown in Table 3.4.5, transistors that are capable of supporting such transfer are increasingly more expensive, especially if multiple are ordered for the operation of the matrix. This quickly eliminates the choices for arrays rated over 1 A capability, as the capacity does not compensate enough for the expensive listing. That leaves the remaining arrays rated at 500 mA. The reason 500 mA was chosen as a standard for comparison was that production availability seemed to support mainly 500 mA for electronics transistors as the next step down from 1 A. Also, every transistor picked was an NPN configuration. NPN transistors are called NPN because of the configuration of their semiconductor elements. This means that the P-side, or electron hole excessive side, is sandwiched between two N-side compounds, which are, conversely, electron excessive. When a voltage is applied at the base of the transistor that is higher than the voltage at the emitter, current is permitted to flow from the collector, where the supply is connected, down through the emitter. This makes for a type of electronic switch that only needs a small amount of voltage to activate. In a PNP transistor, the opposite condition is needed to induce current flow. When the voltage at the base is at high, or close to the voltage at the emitter, the current is not permitted to flow. Current, thereby, only flows when the voltage at the base is pulled down to a threshold below the voltage of the emitter. These relationships characterize NPN transistors as “low-side” transistors, or active high, and PNP transistors as “high-side” transistors, or active low. Since the desired active transistors are those that are active, while all others are OFF, power consumption is better achieved with NPN transistors. This means that when all transistors are OFF, no voltage necessary be applied. In this project, only one transistor will ever be ON at any given moment. So, with NPN transistors, only one will be active at a time, which is inverted for PNP transistors where only one transistor would be OFF at a time, wasting unnecessary voltage on keeping current from flowing. For that reason, only NPN transistors are considered in comparison for this project.

Out of the remaining 3 viable NPN transistors that support 500 mA, the option that is obtainable at the lowest price would normally be chosen. This is because all other differences in the transistors ae negligible for this application. In this case, the optimal transistor would normally be the Toshiba TBD62083 transistor array. The issue with this selection, however, is the limited availability of the product among sellers, and the lack of documentation available in the datasheet, which raises skepticism of the reliability for ordering and using this device. Also, packaging for the Toshiba array is limited to the surface mount variation, making prototyping of this device difficult and costly.

On the other hand, the Texas Instruments TLC 59213 is a fully documented, actively produced, and prototyping capable device. At a price that is comparable to the Toshiba array, this device provides an increased confidence for implementation in this project. Both through-hole and surface mount configurations are available, with two SMD packages available to choose from. Since this device only supports 500 mA, and each row has a potential to draw aver 1 A, the matrix will have to be divided where each transistor is able to fully support the number of LEDs in the row. After several calculations and considerations, the decision was made to divide the matrix into 4 sections, where each section would require 4 transistor arrays. This is from the fact that there are 32 rows, and each array contains 8 outputs. The transistors will receive the activation signal from the refreshing register, and power each corresponding row of LEDs.

3.4.7 Battery Recharger Integrated Circuits

While researching different methods of circuit designs to be able to recharge a 3.7-volt lithium ion battery, I came across two different integrated circuits that are purpose made, specifically for the function of recharging lithium ion batteries.

The first circuit that I found was based around the TP4056 integrated circuit. This integrated circuit was designed specifically for the use of recharging lithium ion batteries, requiring only a few extra external components in its circuitry to do so. In order to design the circuit to specifically recharge 3.7-volt lithium ion batteries, a programmable resistor is connected to a pin on the integrated circuit that is made specifically for the programmable resistor, and the resistance value of that resistor is set to whatever value you need depending on the voltage of your lithium ion battery.

In our case since we were using a 3.7-volt lithium ion battery, the desired resistance value that we needed for our programmable resistor was 1.2 kiloohms. Other than that programmable resistor, the only other required external components for the TP4056 integrated circuit are two ten microfarad capacitors used for noise filtering and voltage spike protection, and a resistor in series with the input power supply for the TP4056 going into the vdd pin for power correction. So that is one of the upsides to using the TP4056 integrated circuit, not many external components are required to implement the respective battery recharger circuit.

One of the downsides of the TP4056 integrated circuit is that, while shopping around for it online, there were no available single units to buy of the integrated circuit, it was only available to buy as a pre-made module with the external circuit already assembled. Along with it being already assembled, the input to the circuit was a mini or micro USB input, and not a regular USB input like our design and sponsor desired. So, in order to get the integrated circuit chip by itself, we had to buy the whole pre-assembled module first, and then de-solder the TP4056 off of the module itself. And on top of that, the leads of the TP4056 that we de-soldered off the module are not long enough for breadboard testing.

The other integrated circuit that we found for use in our battery recharger circuit is the BQ24250RGET from Texas Instruments. This is another integrated circuit that is specifically made for the purpose of recharging lithium-ion batteries. It is a single-input I^2C, stand-alone switched-mode battery recharger with power-path management.

One of the upsides to using this integrated circuit is that it is easily purchasable as a single chip unit, so no de-soldering will be required once the item is purchased and obtained.

However, one of the downsides to using the BQ24250RGET integrated circuit is that it requires more external components to integrate as a battery recharger circuit. Depending on the functionality of the battery recharger circuit that we want, this integrated circuit can require up to fourteen external components as opposed to the four components that the TP4056 requires. Even though most of the components are simple resistors in series with a pin or capacitors for filtering purposes, that still requires a fair amount more of time and money required.

Another upside to this second integrated circuit is that it is cheaper than the TP4056. It costs $3.47 per unit, as opposed to $5.80 per unit for the TP4056. This is most likely because the TP4056 is in a pre-assembled module and the BQ24250RGET is just a single integrated circuit chip.

Table 3.4.7

Part	Manufacturer	Package & Size	No. of Parts	Price ($)
TP4056	NanJing Top Power ASIC	SOP8	4	$5.80
BQ24250RGET	Texas Instruments	VQFN 4mm x 4mm	14	$3.47

Table 3.4.7 shows the two battery recharger integrated circuits compared with a few of their characteristics. As discernable from the table, a few of the reasons we chose the TP4056 was because it has a greatly reduced required number of external components required for integration into our device. It is also not much more expensive, so the slight price increase is not too big of an issue that we need to take into consideration. Also, that fact that the TP4056 first comes as a pre-assembled module makes pre-prototype testing even easier since no external work is done to complete the implemented circuit design.

3.4.8 Bluetooth Modules

While researching design for our Bluetooth circuit, we came across two Bluetooth modules that could possibly be used for implementation in our design. The first of these is the CC2564MODACMOG Bluetooth host controller interface module by

Texas Instruments. The other device we looked at is the LBCA2HNZYZ-711 Bluetooth Low Energy Module by MuRata.

One of the pros that comes with using the CC2564 module is that the supply voltage range for it encompasses the battery we will be supplying its voltage with. Our battery is a 3.7-volt lithium ion battery that we will be using to supply our device with. The absolute maximum supply voltage range for the module is -0.5 volts to 4.8 volts, which fits perfectly with our 3.7-volt battery source.

However, one of the cons of using the CC2564 module is that the recommended printed circuit board stack-up from Texas Instruments requires a four-layer printed circuit board. This will greatly complicate our project and greatly increase the price of it also, since most projects only use a one or two-layer printed circuit board.

One of the cons that comes with using the LBCA2HNZYZ-711 module is that the absolute maximum supply voltage range for the module ranges from -0.1 volts to 3.6 volts. This is an issue for our design because, our supply source is a 3.7-volt lithium ion battery, which is above even the absolute maximum rated voltage for this module. This means that if we tried to use this module it would most likely suffer damage pretty quickly and possibly not work correctly since we would be supplying it with a voltage slightly above its absolute maximum rated voltage. There is a possibility that we could circumvent this issue with some design changes, but that is extra time and money that could be saved in our project if we just use the other module.

One of the pros to using the LBCA2HNZYZ-711 module is that it would require a more simplified printed circuit board stack-up. It would only require one or two layers on our printed circuit board, while the CC2564 module has a recommended four-layer printed circuit board stack-up that Texas Instruments recommends. This would greatly save time in our design of our printed circuit board and in time it would take for a company to manufacture it. This would also save a decent amount of money too, by only requiring half the layers on a printed circuit board.

Table 3.4.8

Part	**Manufacturer**	**Package & Size**	**Integrated Antenna**	**Max. Voltage**	**Price ($)**
CC2564MODACMOG	Texas Instruments	MOG (35)	Yes	4.8 V	$9.93
LBCA2HNZYZ-711	muRata Electronics	MOG	Yes	3.6 V	$10.53

Table 3.4.8 shows the two Bluetooth modules that we researched for our Bluetooth wireless interface circuit for our design. As shown, they are pretty similar in dimensions and price, so those factors are not too big of an issue when deciding. However, the main factor that helped us make our decision was their supply voltage range. As noted before, our voltage supply is our 3.7-volt lithium ion battery. This voltage is above even the absolute maximum for the LBCA2HNZYZ-

711, so that is the main reason on why we decided to go with the CC2564MODACMOG Bluetooth module.

3.4.9 Voltage Boost Circuit

In order for every electronics component in the device to become active, a certain amount of supply voltage and current is required. While the on-board battery is able to supply the necessary current in accordance with its charge capacity, the voltage of the battery is limited to a nominal value. Batteries for portable electronics are normally offered in voltage ratings of 1.5 V, 3.3 V, 3.7 V, 9 V, and so on. Choosing a battery for a particular voltage is separate from the consideration of current capacity, but together these ratings translate to the overall power discharge of the batteries over a particular duration. Unfortunately, though, choosing batteries with high power capabilities quickly becomes space costly, especially with certain chemistries and voltage limitations. For example, 9 V batteries can be made as small as can fit in someone's hand, whereas 12 V car batteries are almost the size of a shoebox and weigh around 25 pounds. As a result of this, only particular battery voltages and chemistries are viable for electronics device applications. From a previous analysis of battery selection, based on these limitations, a 3.7 V Li-ion battery was chosen to power the device.

Considering the supply requirements for the LEDs in the display, 3.7 V is plenty of voltage from the supply battery. The problem, though, is concerned with all of the ICs that control the circuit logic. These discrete chip modules require higher voltage supplies (around 4 to 5 V) that the 3.7 V battery cannot produce. The higher voltages are required to operate these logic chips since the inner circuitry does not raise the voltage given by any means, only providing a maximum of the input voltage. Also, some of the complex circuitry inside these circuits require higher degrees of voltage to distribute power among many outputs or modules, which cause voltage losses translated to the output. The conclusion, then, was that since the battery is unable to supply the necessary voltage to all of the ICs in the circuit, and since no other battery voltage rating is available for our application, the voltage needs to be boosted from the battery.

In order to boost, or raise, the voltage from a source to a higher potential, a certain circuit must be implemented. The fundamental approach to this circuit is to use an inductor, capacitor, and switch together to generate an increased voltage. The inductor is used in series with the input voltage, which uses a magnetic field to store energy. This occurs with the switch in the short circuit condition, connecting the inductor to ground from the source. The inductor in this state continues to collect charge until the switch is toggled to an open circuit. The open allows the inductor to discharge current to the alternate stage of the circuit, where the following inductor terminal is connected to the capacitor and output. In accordance with the properties of an inductor, this discharge causes the inductor to switch polarity, which causes a reversal in voltage drop, or a voltage gain. That voltage, thereby, is added to the supply input, and the result is seen at the output. The result is an output voltage that is higher than the input, producing the boosting effect. This voltage is supplied for a short amount of time as the inductor completes discharging of stored current. To retain this voltage after discharge, the capacitor, which was charged by the inductor current, supplies the remaining supply for output voltage. The switch in the circuit is in the 'short' condition at this point, which allows the inductor to recharge from the supply, as the capacitor discharges on the other side of the disconnected circuit. At the fully recharged point of the inductor coil, the switch is made an open again, repeating the discharge process once again [12]. The process is continued over the course of toggling the switch for the optimal period of time, which is determined by the charging rate of the inductor and capacitor. The constant switching in the IC is modulated using a PWM pin that supplies the switching frequency and duty cycle of the ON state versus OFF. By picking the correct values for these parameters, the output can be supplied and maintained at a voltage that is double, triple, or even higher multiples of the input, making the supply useful for the rest of the circuit.

Implementing this circuit only requires a few fundamental pins on the booster IC chip for the most basic case: the voltage input pin, the ground pin, the voltage output pin, the device enable pin, and the frequency (PWM) pin. Other chips included additional pins for more features but this is the intrinsic scheme for boosting voltage. The capacitive and inductive elements of the circuit are not included internally in the device, so separate components need to be acquired for external circuit connection. The reason for not including these elements in the chip is that the specifications for these components depend on the application of the chip. So if a chip is capable of converting voltage in a certain range with a certain allowable frequency, the designer is able to select certain values of passive components to specify the operating conditions of the chip for their circuitry needs. As a result of this flexibility, datasheets for such booster chips provide equations and resources to find the necessary values based on the output needed. Once these values are obtained, the particular components can be acquired to match those values, and the circuit can operate continuously as designed.

When searching for a voltage boosting component in the stock of various distributors, two types of circuit components were found: comprehensive modules and ICs. The comprehensive modules are fully inclusive chips that contain a fully assembled booster circuit, with only a few pins needed to implement the booster circuit to the application. This scheme is appealing since the component selection and assembly is already complete, removing the need for any calculations or part searches. On the other hand, ICs contain only the single booster chip, with implementation of additional parts necessary for operation. Products for comprehensive chips seemed to be more difficult to find than the ICs, but finding the particular IC that is applicable to this project was complex since a multitude of options are available in all different part numbers, manufacturing, and specifications. The table below, Table 3.4.9, shows a comparison of all possibilities for booster circuits and ICs. Since the comprehensive circuit configuration of these devices is very limited, only one such device was available for comparison, the

Texas Instruments PTN04050C, which is made as a convenient development tool for this type of device. The remainder of the devices are ICs.

Table 3.4.9

Part	Price (USD)	Type	Vin (V)	Vout (V)	I max (A)
PTN04050C	$16.73	Comprehensive	2.95 - 5.5	5 - 15	2.4
TPS61089	$2.85	IC	2.7 - 12	< 12.6	2.0
adp1613	$2.01	IC	1.8 - 5.5	< 20	1.9
TPS6104	$1.44	IC	1.8 - 6	< 28	0.4
LM27313	$1.59	IC	2.7 - 14	< 30	1.25
TPS6306	$2.39	IC	2.5 - 12	2.5 - 8	1.3

The first decision to make when considering the selection shown above was whether to use a comprehensive circuit or an IC. As mentioned before, comprehensive circuits are convenient to implement since they are pre-assembled, but the price for a single unit is over 5 times that of the ICs, shown in Table 3.4.9. Additionally, the size of the comprehensive circuit is large compared to all of the ICs. The ICs are supplied in SMD packaging that has dimensions at less than 1/8”, whereas the comprehensive circuit has a height of 1/3” and linear layout of about 1” x ½”, which is massive in comparison. n that regard, the comprehensive option is very costly in terms of money and space, outweighing the benefit of convenience over the ICs. Also, since the booster circuit is pre-assembled, there is no flexibility to adjust the parameters of the circuit, limiting the range of operation to what was already designed. If any changes are determined for the circuitry of the device, this rigidity may pose a problem for implementation in the manufacturing process.

Instead, ICs provide comparable, if not better, performance than the comprehensive circuits, at a much lower space and monetary cost with the ability to create and adjust intended design. Unlike the comprehensive circuits, there were many ICs to choose from, so a group of ICs that best fit the needs of our device at a reasonable price were compared, as seen in Table 3.4.9. The most important characteristics of the ICs to compare were the minimum input voltage and the maximum output voltage ratings, as well as price for a single unit. Since the battery supplies 3.7 V maximum, an IC that allows around 3 V for input at the least is ideal, considering possible drops in voltage from the battery over the course of discharge. The selected chips reflect this, additionally with an output voltage that needed to be greater than around 6 V, where a nominal 5 V is needed for the logic chips, and between 5 and 6 V is needed for the MCU supply. Current rating is fairly significant, but not crucial since these chips only need around 1 uA in order to operate, and most of the selected booster ICs supply currents of 400 mA and above.

Looking at Table 3.4.9, the comparison clearly shows how similar the characteristics are between each of the choices, all within the ideal operating ranges needed for the project. This leaves only price and output current. Current is not necessary is the magnitudes of Amps and is wasteful if current is lowered by a resistor, using spending electric force on a passive component. Therefore, increased output currents are not only unnecessary, but costly to the longevity of the battery. Of all the ICs, the Texas Instruments TPS6104, thereby, proves the ideal option for boosting the battery voltage. All other aspects satisfy the operating requirements, and the chip is able to reduce current expense, only transferring 400 mA to the output terminal, which is plenty for device operation. Also, the TPS6104 turns out to be the cheapest option, with the next economical choice at a price difference of $0.14. Although that may not seem like much in terms of savings, consistent economic considerations will prove very beneficial after all costs are added. The only requirement left for implementation once the chip is obtained, is the assembly of the remaining circuit inductor, resistors, and capacitors, which will be connected externally to the booster chip. To choose these components for this booster design, Texas Instruments also provides equations and online resources to create the circuit quickly for the application that we need.

4. Constraints and Standards

Constraints and standards, along with client specifications covered later in this document, are the three main forces that govern the execution of this project. They provide a framework within much the design must fit, unless it violates one of many requirements. These constraints and standards are set by political and engineering institutions, along with the corporations that produce products vital to the operation of the LED sign device. The design choices found in section five are all made with the intent of meeting these constraints and standards.

4.1 Realistic Design Constraints:

At the end of our engineering curriculum, us students are required to plan and build a senior design project in a group setting. This project has its own requirements and deadlines that must be met to pass the class and graduate. However, while taking all this into account, it is important to realize what realistic constraints will be impactful on your design, and to properly account for and plan for them to not get caught up in unrealistic goals and end up in failure. The modern engineer has become more and more concerned with realistic design constraints since our society has advanced and is more well-off than ever. This adds another layer of complexity to the engineer's task, since they need to now plan and shape their design to account for the greater amount of constraints. Engineers need to be more imaginative and ingenious than ever, and need to try their best to predict the most unlikely of hazardous situations when it comes to using and building their design, and in doing so, can prepare for it by trying to account for these constraints in the development of their design. The effect of realistic design constraints on an engineer's design can be better visualized in this diagram:

These constraints can show up in many ways, such as economic or budgeting constraints, environmental constraints, political constraints, social constraints, timing and deadline constraints, sustainability constraints, health and safety constraints, ethical constraints, and many more that could take quite some time to list out. Now, to go into detail on some of these constraints.

4.2 Economic/Budgeting Constraints

Economic and budgeting constraints are one of the most common constraints when designing, since every single design must follow these constraints, and it is one of the first things that any engineer will think about when designing a device. These constraints include a plethora of fiscal variables such as:

- The total estimated price of the final design.
- The prices of current related or related products on the market and the cost and profit of your design.
- The available budget that the engineer has at their disposal for their design.
- The potential impact to the local and United States economy, and, if applicable, the world economy.

Our design is impacted by economic and budgeting constraints in a few ways, just like every design. One way it is impacted is by our sponsor's method of reimbursement. The agreement with our sponsor was that he would pseudo-sponsor our project, meaning that if at the end of our design and fabrication processes, if our project satisfied his desired requirements and specifications, he would pay us a total of $350 as reimbursement for our work. However, if we failed to deliver a satisfactory product at the end, then we will end up having to have paid for the whole design out of our pockets. So, for our total estimated price of our final design, we are hoping and trying to keep it under $350, so that if we do satisfy his desired final presentation, that we will not have to pay for any of it by ourselves; and in an even better scenario, might even profit off of our efforts.

Another way economic and budgeting constraints affects our project is, we have to account for the price of each component that we purchase to use in our design. Each part can have multiple vendors, so we need to shop around for each part at different places in order to make sure that we get it for the cheapest price possible.

Also, each part can have another part that is similar or at least can perform all the same duties required of the original part, and if that different part is cheaper, we can purchase that part instead to save money; of course, this requires more research into each part, but might be worth it for the sake of the budgeting constraints.

Another economic constraint is actually one of the main purposes of our project, and one of the main qualities that our sponsor is interested in us making sure that we design around. We need to research the cost and price of other similar devices on the market -- which is an arduous task since there is a great amount of LED handheld message devices on the market – and make sure that our device is cheaper to fabricate and also cheaper for consumers to purchase. Our sponsor has even already purchased several of these competitor devices for several reasons including, us being able to see what our final design should roughly look like, how our final design might similarly function, and to take it apart to get an idea of how to build our own device, and learn how handheld LED message devices work in the first place so that we can figure out how to build our own version. Inversely, in dealing with profitability of our device, that constraint does not necessarily affect us, since the purpose of this design and project is to gain experience in designing a device as an engineer in order to graduate and receive our degree. However, our group can possibly profit off of our project, if we satisfy our sponsor's desires at time of completion and spend less than the reimbursement total of $350; but that is not a budget constraint on our design since it is not a requirement for us to profit.

Our available budget for our design is not a specific amount, it is just the amount that us in our group can realistically afford. The reimbursement amount is just a sort of 'reward' if we do well on our project, and not necessarily a budget that we have to worry about.

Since our design is just a small device that will not be purchased by a consumer or business directly from us, and is just being done for a required project in order to graduate, that means we do not have a direct impact on the local or United States economy. So, in this case, that is not a constraint that we have to worry about. However, if our sponsor likes our device and decides to purchase it and pursue it in the marketplace, then he might have to take this constraint into consideration, but for all intents and purposes, we do not.

4.3 Environmental Constraints

Environmental constraints are also realistic constraints that virtually every engineering design will have to take into consideration, since it affects every design in some way, no matter where in the world you are. In the modern era, environmental constraints are becoming more and more important to engineers when designing projects, since us as human beings are becoming more and more aware every day of the impact our daily lives and devices are having on the environment. Engineers also not only have to worry about the impact their final device will have on the environment once in consumer hands, but also how much impact the creation and manufacturing of their device will have on the environment too.

However, environmental constraints can extend beyond just your local and planetary outdoor surroundings. These constraints also include the environment that your product will be used in, which includes the workers making your design, the consumers using your designed device, the general public not even using your device but still near enough to be affected by it, and anything else that can come into contact with your device in any way.

Environmental constraints can impact an engineering design in many ways, since so many aspects of the world can be affected. Some of these constraints include:

- Vibration induced noise to workers and product users.
- Vibration induced noise to the public.
- Air pollution.
- Water pollution.
- Effects on the landscape such as waste parts.
- Global warming from manufacturing or use of final product.
- Manufacturing waste collection and processing.
- Space debris.
- Control of energy saving devices.

Luckily with our project, it is a relatively small and noiseless electrical device, so we do not need to worry about any constraints dealing with vibrations, since any vibrations that are being made by our device are so miniscule that they are negligible to any person or environment.

Both air pollution and water pollution are also not factors in the design or use of our project, because we are just buying parts from the internet and assembling them into a device that displays messages on an array of light emitting diodes. On one hand, the manufacturers of the parts that we are purchasing online could definitely be having an impact on air or water pollution whenever they are making the parts, but that is beyond the scope of our project, and would require plentiful, difficult research for us to even find out.

Our project also won’t have much of an impact on the landscape, if at all, since it is just one solid piece of equipment with no wasteful parts that could come from it, unless you disassembled it. Of course, it can contribute to light pollution when used at night, but our light emitting diode array will produce such a small amount of light compared to the overall amount of light pollution that it’s overall contribution will be negligible.

Our project’s contribution to global warming will be relatively miniscule, since the heat from our device will just come from the microelectronics on our printed circuit board like resistors, transistors, and the microcontroller, along with the light emitting diodes. In the overall scheme of things, our heat dissipation will be negligible.

Since in this course, we only design and built our one project and will not go into manufacturing, no manufacturing constraints such as waste collection and processing will be a factor in our design and building. Our device will also have no factor with space debris since at no point will we ever contribute to any matter being in space.

We do have one energy saving device included in our project, and that is a pulse width module. This device alters the timing on the pulses being used to turn the light emitting diodes on and off, which in turn dims the brightness which saves energy.

4.4 Social Constraints

Social constraints are more prevalent when an engineer is designing a mass-produced product or government funded project. These constraints can include:

- Designs in favor of certain people but against others.
- Worker union versus employer.
- Government codes are to protect society.

In turn, since our college project of a small handheld electronic sign pertains to none of these situations, social constraints do not affect our design.

4.5 Political Constraints

Political constraints are arguably the most important constraints for an engineer to take into account when designing depending on what the design is and where the engineer is working and where the intend the final product to end up. If the engineer is being funded publicly, mass producing a product for a certain demographic, or designing a product for questionable customers, then these constraints are of the upmost importance. These constraints include but are not limited to:

- Design using software/hardware developed under public funding.
- Products (e.g., computer games, marks on clothes) that profile negative sides of a specific race or gender.
- Products for use in space that use on-earth patent protected designs/concepts.
- Products for customers who are against the United States.
- Products that are against United States homeland security.
- Products that are physically and/or mentally destructive for people.

Since our project is designed using only privately-funded tools that are purchased by us, we do not need to worry about the ramifications of using publicly funded software and hardware.
Our project is not realistically able to possibly negatively profile any race, gender, religion, etcetera just in its unused state. Yes, users can use the device to display messages that may discriminate against others, but that is not something that can be anticipated while we are designing and is at the discretion of the user.
Our product is not intended for use in space, so we do not need to worry about using designs/concepts that are patented on earth any more than we already have researched into that.
Our product is for our sponsor and our professors, which none are against the United States, so we do not need to worry about our product being used by enemies of the United States.
Our product is not against United States homeland security, and we can be sure of that because there are similar products that are not against homeland security, and our product also is not in any violations of homeland security.
Our product does not affect the mental or physical health of any users, so we do not need to worry about it being physically or mentally destructive for any users.

4.6 Ethical Constraints

Ethical constraints affect engineering designs as much as they affect any other field. Engineers need to constantly keep this type of constraint in mind as they are designing, as they could have to completely scrap their design, or get in serious trouble with harsh punishments if they violate these constraints. These ethical constraints can include but are not limited to:

- Designs without considering safety and health of workers, consumers, and/or the public.
- Products implicitly using patent protected designs/concepts
- Products that use radioactive materials.
- Products that use materials that have better appearance but are toxic.
- Under design for profit.
- Products for secrete survey of personal private life.

Since our project is relatively small and basically completely harmless in typical usage, we do not need to take the health and safety of any workers, consumers, or the public while designing our project.
While our project is similar to other already made electronic handheld signs, we are paying close attention to patent protected designs and concepts. We are making sure that we do not infringe upon any of these patents in our design.

We know for a fact that our project is not at risk of using any radioactive materials, so we do not need to worry about that, and we also do not need to worry about using a material that has a better appearance but is toxic.

Since our project is not intended to make a profit and is only for our sponsor and professors, the risk of us under designing for profit is not an ethical constraint that we need to take into consideration.

Since the only way our project can connect or gather any information from any users is through Bluetooth connectivity, we also do not need to worry about not secretly surveying personal private life with our designed project.

4.7 Sustainability Constraints

Sustainable engineering is the process of designing or operating systems such that they use energy and resources sustainably, at a rate that does not compromise the natural environment, or the ability of future generations to meet their own needs. Sustainability constraints stem off from this, as when an engineer is designing, they need to keep in mind the sustainability of their current design, and if it is not realistic, may need to cut back on any of the current needs of their device in order to make it sustainable. Sustainability constraints can include but are not limited to:

- Can the business survive?
- A well-defined lifespan under the assumed normal operation conditions.
- Consideration of actual environmental factors (extreme working temperature, corrosive fluid, abrasive air, severe radiation in space, etcetera) in their final design.
- All parts need to have a similar designed lifespan.
- Machines require perfect suppression of vibration to function.
- Reliability and durability of the product’s supposed function and lifespan.

Since we are just designing one device for a project, the question of the business can survive is not applicable to us. Consideration of actual environmental factors is not as extreme as corrosive fluids or extreme temperatures since the sign will just be used in everyday life, but we can do some environmental testing that includes things such as view distance, nighttime testing, window tint affecting the visibility, visibility in fog, and other, more average, everyday life environmental testing.

Since all of our parts that we are using are microelectronics, they all have relatively similar lifespans; since they all last for years if not decades, surviving tens of thousands of hours of use, and that is only if the device were to be on the whole time, which it will not be.

Once again, the vibration in our device is negligible, since it is not a machine, so that is not a constraint we have to worry about. However, we do need to worry about the durability and reliability of the product’s supposed function and lifespan. We will design around this constraint by using the best materials that we can afford for durability, and test for reliability by doing some prototype testing; making sure that everything is working correctly before we send it off to the sponsor and professors.

4.8 Timing and Deadline Constraints

Timing and deadline constraints are arguably the most prominent constraints that we must worry about while designing and building our device. The main timing constraints we have are the deadlines put in place by our professors for the document milestones; and once the document is finished, then the deadlines that are put in place for our project building milestones will be the main timing constraints. We also have other, smaller timing constraints that we have made for ourselves along the way in order to keep on track for the bigger deadlines, all of which are listed out in our milestones and task list tables in their respective sections of this document. Timing constraints can include but are not limited to:

- Documentation deadline.
- Prototype deadline.
- Design schedule.
- Development schedule.
- Production schedule.
- Delivery schedule.
- Availability of engineers.

Our project is constrained by document deadlines for the entirety of the document writing phase. The first deadline that was implemented was the ten-page deadline, which was in mid-September. The next documentation deadline was the sixty-page deadline, which was November third. After that one, the next deadline was a few weeks later for the one-hundred-page milestone. Lastly, the finalized project document of one-hundred-twenty-pages was due on December fourth. Along with documentation deadlines, our design is constrained by prototype deadlines. Once our documentation is finalized, we will move on to building the actual prototype. Once we get involved with this, it will have milestone deadlines just as the document did, with check ins on progress every month or so, and will have to be finished in time for graduation sometime in May.

Design schedule constraints consist of things such as project planning and project control. We have done project planning for our design and device, which is laid out as dates in the milestone list and task list, so that we can stay on track in order to finish on time. Project controls are not constraints that affect our design, since we are not going to be gathering data on management and processes, since that is not necessary or technically possible for our project.

Development schedule constraints consist of design detailing, compliance tests, and more. Since our design is just made by us four students for our school project, it is not applicable to compliance testing or design detailing, so these constraints do not affect our design. Also in our project's case, production schedule is virtually the same thing as our design schedule and class deadlines and milestones, so production schedule constraints fall under that. Lastly, delivery schedule constraints do not apply to our design either, since we are just going to turn in our project to our sponsor and professors once we are done designing and creating it.

4.9 Health and Safety Constraints

Health and safety constraints need to be taken into consideration by an engineer if his designed device will get into the hands or affect any person in any way, or will be made by other people. The health and safety of people using the design, near the design in public, any people that will be used to make the design in the future. Many businesses and organizations have their own health and safety warnings and methods, as shown below. A decent amount of the warnings and methods, while constraints on the engineer when he is designing, are also standards, which will be discussed in their own section. Health and safety constraints can include but are not limited to:

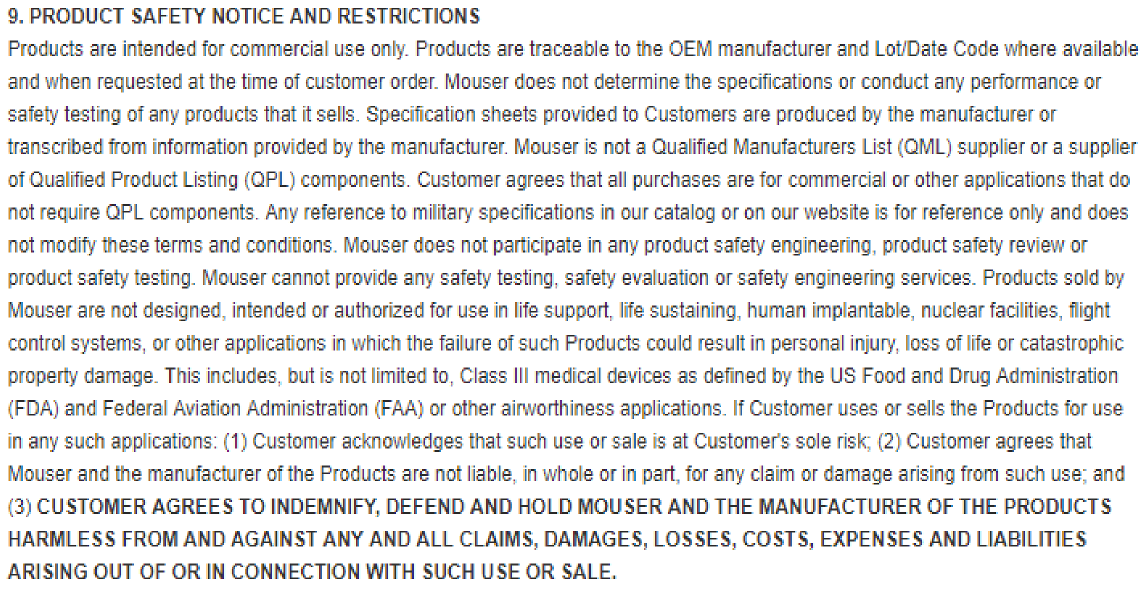
- Safety of consumers and workers.
- Safety of the public.
- Noise that causes hearing loss.
- Hazardous materials and environment for workers.
- Products that require the use of radioactive materials.
- Products that use materials that have better appearance but are toxic.
- Products for infants/children that require special safety considerations.
- Design of a control system with acceptable stability margins for machinery where safety is of concern.

Since our project will just be crafted by our group, we do not need to worry about the safety of any workers. However, we do need to worry about the safety of any users of our device; such as making sure that they do not get shocked from any of the electronics, have possible damage to eyesight from brightness, encounter any harm from sharp or protruding parts of the design, etc. In order to design for these constraints, we will make sure that all of the electronics are safely enclosed inside of our casing; make sure that the brightness of the light emitting diodes cannot get too bright as to cause harm to eyesight; and make sure that our designed device has no protruding edges or anything of the sort that can cause bodily harm during use.

We also need to take the safety of the public into account, because even if they are not the user of our device, they can still be affected by it if they are near enough to the user. In this case, the only real possibility of harm would be a result of the sign being too bright, so once we design the sign to make sure that it will not be too bright for the user, that will ensure that the public will also be safe from it since they will be further away from the device than the user is.

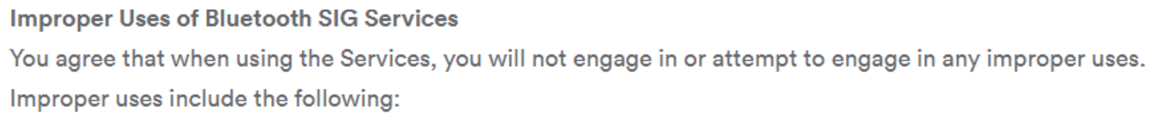
Since our device makes no noise, there is no risk of hearing damage or loss that needs to be taken into consideration while designing our device. We also don't need to worry about hazardous materials for any workers since the device will be assembled by our group, and the environment is at no risk from hazardous materials since none are included in our design.

Since we are using light emitting diodes from the company Mouser for our message array, these are the product safety notice and restrictions on the Mouser website, located in the sale terms section.


9. PRODUCT SAFETY NOTICE AND RESTRICTIONS

Products are intended for commercial use only. Products are traceable to the OEM manufacturer and Lot/Date Code where available and when requested at the time of customer order. Mouser does not determine the specifications or conduct any performance or safety testing of any products that it sells. Specification sheets provided to Customers are produced by the manufacturer or transcribed from information provided by the manufacturer. Mouser is not a Qualified Manufacturers List (QML) supplier or a supplier of Qualified Product Listing (QPL) components. Customer agrees that all purchases are for commercial or other applications that do not require QPL components. Any reference to military specifications in our catalog or on our website is for reference only and does not modify these terms and conditions. Mouser does not participate in any product safety engineering, product safety review or product safety testing. Mouser cannot provide any safety testing, safety evaluation or safety engineering services. Products sold by Mouser are not designed, intended or authorized for use in life support, life sustaining, human implantable, nuclear facilities, flight control systems, or other applications in which the failure of such Products could result in personal injury, loss of life or catastrophic property damage. This includes, but is not limited to, Class III medical devices as defined by the US Food and Drug Administration (FDA) and Federal Aviation Administration (FAA) or other airworthiness applications. If Customer uses or sells the Products for use in any such applications: (1) Customer acknowledges that such use or sale is at Customer's sole risk; (2) Customer agrees that Mouser and the manufacturer of the Products are not liable, in whole or in part, for any claim or damage arising from such use; and (3) CUSTOMER AGREES TO INDEMNIFY, DEFEND AND HOLD MOUSER AND THE MANUFACTURER OF THE PRODUCTS HARMLESS FROM AND AGAINST ANY AND ALL CLAIMS, DAMAGES, LOSSES, COSTS, EXPENSES AND LIABILITIES ARISING OUT OF OR IN CONNECTION WITH SUCH USE OR SALE.

Another constraint that we should consider while designing are the constraints that Bluetooth put upon us. When you agree to their terms of use and code of conduct, they have a list of agreements that you must adhere to. These are listed below, taken from the code of conduct section of their website.


Improper Uses of Bluetooth SIG Services

You agree that when using the Services, you will not engage in or attempt to engage in any improper uses. Improper uses include the following:

- Violating any applicable law or regulation;
- Uploading, transmitting, or otherwise sharing any content you do not have the right to upload, transmit, or share;
- Uploading, transmitting, or otherwise sharing content that infringes any trademark, patent, trade secret, copyright, publicity, privacy, or other right of Bluetooth SIG or of any third party;
- Uploading or transmitting content that is unlawful, fraudulent, untrue, stalking, harassing, libelous, defamatory, abusive, tortious, threatening, obscene, hateful, abusive, harmful or otherwise objectionable as determined in Bluetooth SIG's sole discretion;
- Causing damage to Bluetooth SIG's business, reputation, employees, members, facilities, or to any other person or legal entity;

- Attempting to intercept, collect or store data about third parties without their knowledge or consent;
- Deleting, tampering with, or revising content posted by any other user unless otherwise permitted;
- Accessing, tampering with, or using non-public areas of the Services;
- Attempting to probe, scan, or test the vulnerability of a system or network or to breach security or authentication measures;
- Attempting to access or search the Services or any network or system provided by Bluetooth SIG or used to operate the Services with any engine, software, tool, agent, device or mechanism other than with software or search agents provided by Bluetooth SIG or generally available web browsers;
- Sending or attempting to send unsolicited commercial messages, including promotions or advertisements for products or services, "spam", "chain mail" or "junk mail";
- Using or attempting to use the Services or any network or system provided by Bluetooth SIG or used to operate the Services to send altered, deceptive or false source-identifying information;
- Attempting to decipher, decompile, disassemble, or reverse engineer any of the software comprising or in any way making up a part of the Services, any network or system provided by Bluetooth SIG or used to operate the Services, or any user content provided in object code form;
- Interfering or attempting to interfere with the access of any user, including by sending a "virus" to the Services or any network or system provided by Bluetooth SIG or used to operate the Services, or overloading, "flooding," "spamming," "crashing," or "mailbombing" any of the foregoing;
- Impersonating or misrepresenting your affiliation with any person or entity;
- Using the Services to make fraudulent offers to sell or buy products, items, or services or to advance any type of financial scam such as "pyramid schemes," "Ponzi schemes," unregistered sales of securities, and securities fraud; or
- Excessively high-volume data transfers or bandwidth consumption, hosting of a web server, internet relay chat server, or any other server, and non-traditional end user activities.

4.10 Standards

A standard is a document that defines the characteristics of a product, process, or service, such as dimensions, safety aspects, and performance requirements. It is important for an engineer to be knowledgeable of these standards while designing, because if they violate them at any point, they could have to scrap their design and start over from the beginning. There are quite a few organizations that handle engineering standards and other types of standards, and for our project we will delve into several of them to find the standards that apply to our project.

4.10.1 IEEE 802.15 Standard

One of the standards that our design must comply with is IEEE standard 802.15.1 since we incorporated Bluetooth into our design. This standard is for wireless medium access control (MAC) and physical layer (PHY) specifications for wireless area networks.

The scope of this standard is that it defines physical layer (PHY) and medium access control (MAC) specifications for wireless connectivity with fixed, portable, and moving devices within or entering a personal operating space (POS). A personal operating space is the space about a person or object that typically extends up to ten meters in all directions and envelops the person whether stationary or in motion.

The original goal of the IEEE 802.15.1 Task Group was to achieve a level of interoperability that could allow the transfer of data between a WPAN device and an IEEE 802.11 device. Although this proved infeasible, IEEE standard 802.15.1-2005 does have mechanisms defined to allow better coexistence with IEEE 802.11b class of devices.

Both this standard and the previous version are based upon technology originally developed by the Bluetooth Special Interest Group (SIG). This standard describes some of the following topics in order for Bluetooth devices to be compatible with other wireless networks: the functions and services required by an IEEE 802.15.1-2005 device to operate within ad hoc networks, the Logical Link Control and Adaptation Protocol (L2CAP) layer, and the 2.4 GHz industrial, scientific, and medical (ISM) band PHY signaling techniques and interface functions that are controlled by the IEEE 802.15.1-2005 MAC to define the quality of the system and provide compatibility between the radios used in the system.

Pictured below is the service access point interface used by Bluetooth. The illustration shows the events and actions performed by an implementation of the L2CAP layer that is used by Bluetooth wireless connection.

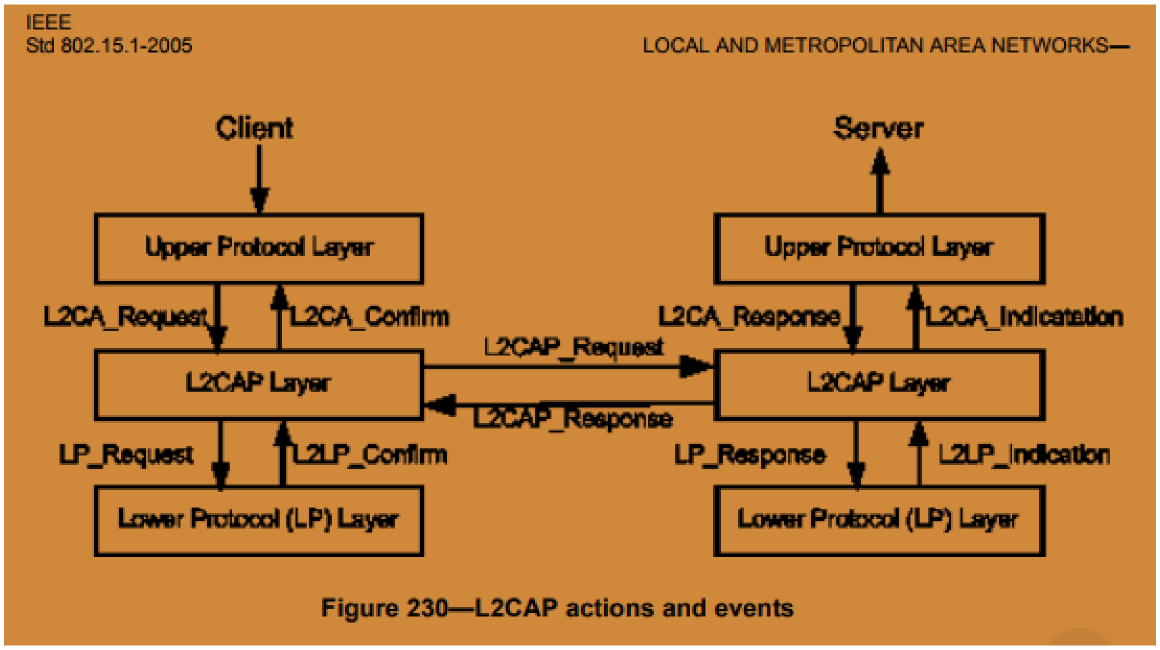


Figure 4.10.1 A diagram of the Bluetooth communication protocol.

As seen in this pictured L2CAP diagram, there are three layers each for the client side and server side of the Bluetooth protocol, all of which are specifically defined in the IEEE standard 802.15.1-2005 to make sure that these protocols are used and set up correctly

.

4.10.2 Bluetooth in IEEE 802.15.1

According to the Bluetooth website, Bluetooth is a low-power wireless connectivity technology used to stream audio, transfer data, and broadcast information between devices. Bluetooth transmits over short-wavelength ultra high frequency radio waves in the industrial, scientific, and medical radio band from 2.4 to 2.485 GHz. It was invented by telecom vendor Ericsson in 1994, and it was originally conceived as a wireless alternative to RS-232 data cables. There are two flavors of Bluetooth technology, Basic
Rate/Enhanced Data Rate (BR/EDR) and Low Energy (LE).

4.10.3 Basic Rate/Enhanced Data Rate (BR/EDR)

Basic Rate/Enhanced Data Rate (BR/EDR) Bluetooth is point-to-point and enables continuous wireless connections and uses a point-to-point (P2P) network topology to establish one-to-one (1:1) device communications. Bluetooth BR/EDR audio streaming is ideal for wireless speakers, headsets, and hands-free in-car systems.

4.10.4 Low Energy

Low-Energy Bluetooth enables short burst wireless connections and uses multiple network topologies, including point-to-point, broadcast, and mesh.

4.10.5 Point-to-Point

Point-to-point (P2P) is a network topology used to create one-to-one (1:1) device communications. Bluetooth LE P2P topology is ideal for data transfers and well suited for connected device products, such as fitness trackers and health monitors.

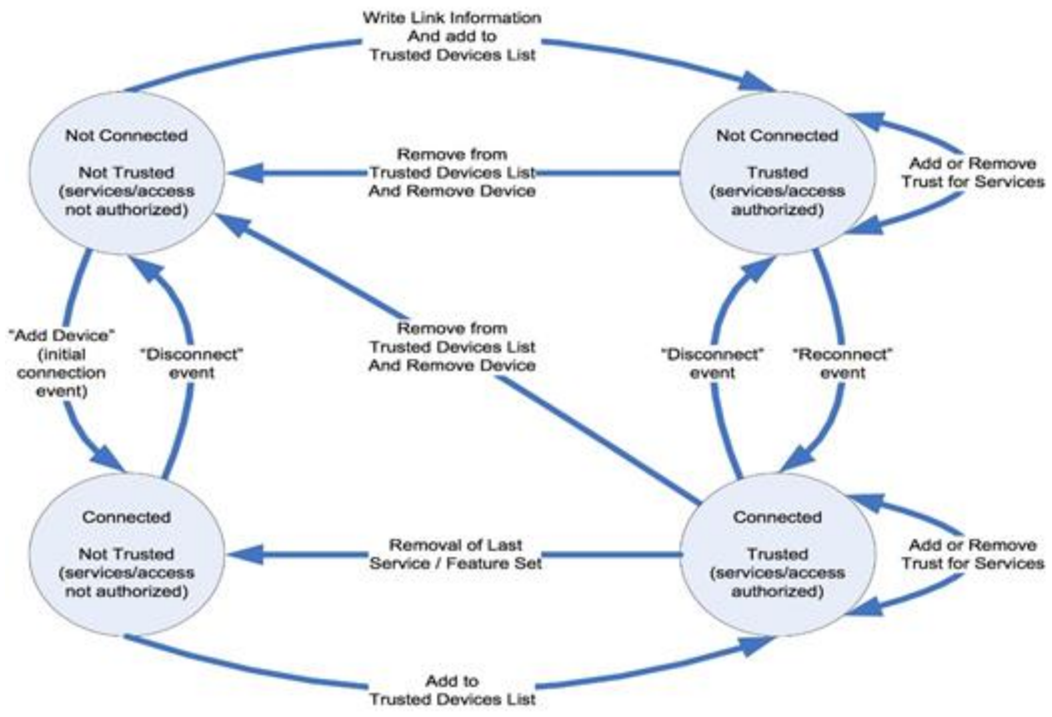
4.10.6 Broadcast

Broadcast is a network topology that establishes one-to-many (1:m) device communications. Bluetooth LE broadcast topology optimizes localized information sharing, making it ideal for beacon solutions, such as point-of-interest (PoI) information and item and way-finding services.

4.10.7 Mesh

Mesh is a network topology for many-to-many (m:m) device communications. Bluetooth Low Energy mesh topology creates large-scale device networks tailor-made for building automation, sensor network, asset tracking, and any solution where multiple devices need to reliably and securely communicate with one another.

Our design will most likely incorporate the Bluetooth Low Energy poin-to-point network topology since we only require a point-to-point communication, the two points being between an Android device and our Bluetooth receiver circuit on our handheld LED device. Shown below is how a Bluetooth device connects two devices in a simplified state diagram from the website stackoverflow.


Write Link Information And add to Trusted Devices List
Not Connected Not Trusted (services/access not authorized)
Remove from Trusted Devices List And Remove Device
Not Connected Trusted (services/access authorized)
Add or Remove Trust for Services
"Add Device" (initial connection event)
"Disconnect" event
Remove from Trusted Devices List And Remove Device
"Disconnect" event
"Reconnect" event
Connected Not Trusted (services/access not authorized)
Removal of Last Service / Feature Set
Connected Trusted (services/access authorized)
Add or Remove Trust for Services
Add to Trusted Devices List

4.10.8 IEEE Standard 802.15.4

Another standard that our design could possibly have to comply with is IEEE standard 802.15.4. Our design will have to conform to the rules of this standard if we decide to use ZigBee as our wireless provider device. ZigBee is a low-cost, low-power, wireless mesh network standard targeted at battery-powered devices in wireless control and monitoring applications, like our device. The IEEE 802.15.4 standard is a standard that defines the operation of low-rate wireless personal area networks (LR-WPANs) like ZigBee. It specifies the both the physical and media access control layers for low-rate wireless personal area networks and has been doing so since the standard was defined in 2003. Other low-rate wireless personal area networks that are defined by this standard besides ZigBee include ISA100.11a, WirelessHART, MiWi, SNAP, and Thread.

4.10.9 ZigBee in 802.15.1

ZigBee uses a ZigBee coordinator at the center of the communication network to link all of the connected devices together. In between the ZigBee coordinator and all of the interconnected devices, are ZigBee routers that handle the transfer of data from the coordinator to and from the other connected devices. ZigBee end devices are connected to the interconnected devices by way of star links, while the ZigBee routers are connected by way of mesh links. The mesh links connect the Zigbee coordinator with the ZigBee routers, and the ZigBee routers to the ZigBee end devices, and the end devices are in their own personal network in a star linked network. The way ZigBee wireless communication works in simplified in this diagram below, showing the different types of devices and the different types of links.

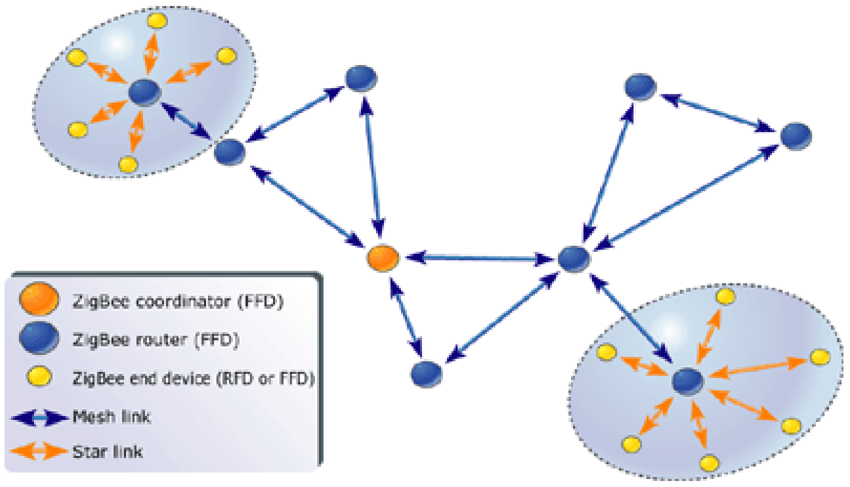


Figure 4.10.9.1 A diagram showing the different connection points of the wireless ZigBee system as it pertains to Bluetooth connectivity.

According to the Wikipedia page for the IEEE standard 802.15.4, the standard intends to offer the fundamental lower network layers of a type of wireless personal area network which focuses on low-cost, low-speed ubiquitous communication between devices. It can be contrasted with other approaches, such as Wi-Fi, which offer more bandwidth and require more power. The emphasis is on very low-cost communication of nearby devices with little to no underlying infrastructure, intending to exploit this to lower power consumption even more.

The basic framework conceives a 10-meter communications range with a transfer rate of 250 kbit/s. Tradeoffs are possible to favor more radically embedded devices with even lower power requirements, through the definition of not one, but several physical layers. Lower transfer rates of 20 and 40 kbit/s were initially defined, with the 100 kbit/s rate being added in the current revision.

Even lower rates can be considered with the resulting effect on power consumption. As already mentioned, the main identifying feature of IEEE 802.15.4 among WPANs is the importance of achieving extremely low manufacturing and operation costs and technological simplicity, without sacrificing flexibility or generality.

Important features include real-time suitability by reservation of guaranteed time slots, collision avoidance through CSMA/CA and integrated support for secure communications. Devices also include power management functions such as link quality and energy detection.

IEEE 802.15.4-conformant devices may use one of three possible frequency bands for operation (868/915/2450 MHz).

In dealing with this standard, devices are conceived to interact with each other over a conceptually simple wireless network. The definition of the network layers is based on the Open Systems Interconnection model; although only the lower layers are defined in the standard, interaction with upper layers is intended, possibly using an IEEE 802.2 logical link control sublayer accessing the MAC through a convergence sublayer. Implementation may rely on external devices or be purely embedded, self-functioning devices. The protocol stack for IEEE standard 802.15.4 is shown below.

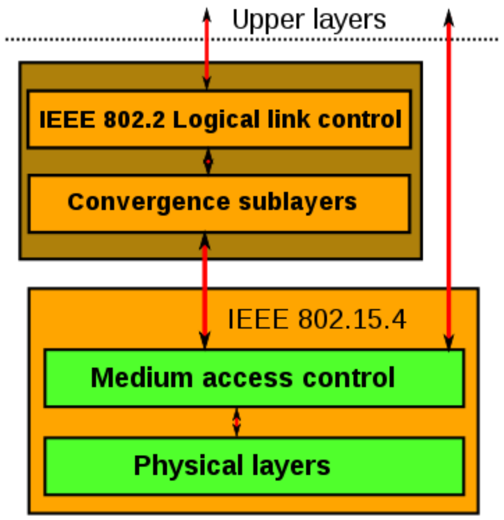


Figure 4.10.9.2 A diagram that displays the different layers of the IEEE transfer protocol.

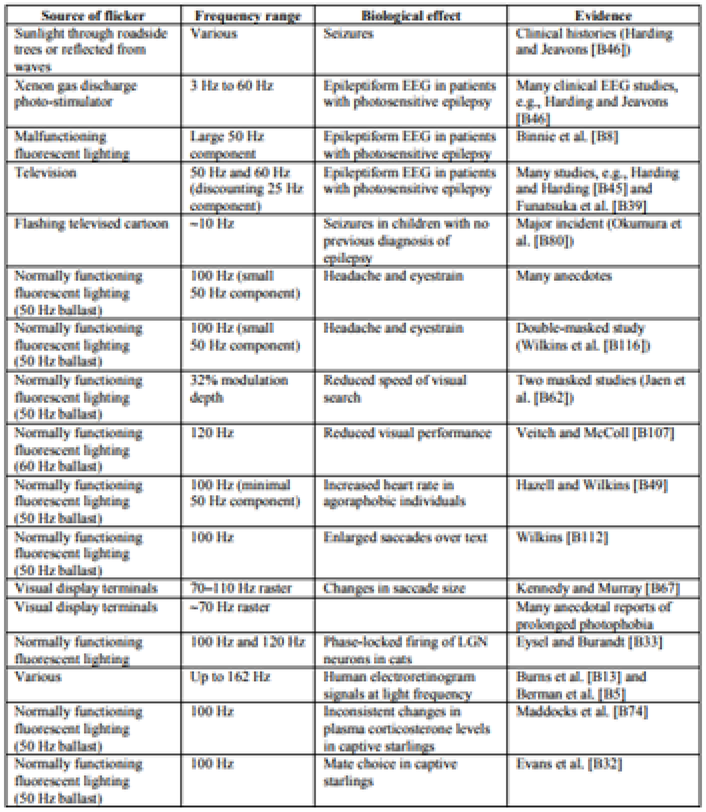
As seen in the figure, the IEEE standard 802.15.4 only defines the standards for the lower layers including medium access control and physical layers, but it is intended for these lower layers to communicate with the upper layers. That is why IEEE standard 802.2 is included in the upper layers through IEEE standard 802.2 logical link control, so that all layers are standardized.

4.10.10 IEEE Standard 1789-2015

Another standard that our design must take into consideration is IEEE standard 1789-2015, IEEE Recommended Practices for Modulating Current in High-Brightness LEDs for Mitigating Health Risks to Viewers. Our device utilizes an LED array with many LEDs, and we will then be using a phase-width modulator to then vary the pulses of the LEDs to therefore adjust the brightness to the user’s desired level of brightness.

The abstract of this standard states that the IEEE standard 1789-2015 includes a definition of the concept of modulation frequencies for light-emitting diodes (LEDs), a discussion on their applications to light-emitting diode lighting, a description of light-emitting diode lighting applications in which modulation frequencies pose possible health risks to users, a discussion of the dimming of light-emitting diodes by modulating the frequency of driving currents/voltage, and recommendations for modulation frequencies (flicker) for light-emitting diode lighting and dimming applications to help protect against known potential adverse health effects.

Light-emitting diodes can have adverse effects on user's health if the brightness they emit is too high, which is why pulse-width modulation is a function we are including in our design. Blue light coming from a light-emitting diode causes a photochemical risk to the eye depending on the accumulated dosage. The International Commission on Non-Ionizing Radiation Protection suggests that you should not look directly into a light box from up close for longer than a maximum of one hundred seconds. Different frequencies at which light sources flash affect humans in different ways, which is why pulse width modulation alters the frequency of the light-emitting diodes. The different effects of light sources at different frequencies are elaborated upon in the table below.



Source of flicker	Frequency range	Biological effect	Evidence
Sunlight through roadside trees or reflected from waves	Various	Seizures	Clinical histories (Harding and Jeavons [B46])
Xenon gas discharge photo-stimulator	3 Hz to 60 Hz	Epileptiform EEG in patients with photosensitive epilepsy	Many clinical EEG studies, e.g., Harding and Jeavons [B46]
Malfunctioning fluorescent lighting	Large 50 Hz component	Epileptiform EEG in patients with photosensitive epilepsy	Binnie et al. [B8]
Television	50 Hz and 60 Hz (discounting 25 Hz component)	Epileptiform EEG in patients with photosensitive epilepsy	Many studies, e.g., Harding and Harding [B45] and Funatsuka et al. [B39]
Flashing televised cartoon	~10 Hz	Seizures in children with no previous diagnosis of epilepsy	Major incident (Okumura et al. [B80])
Normally functioning fluorescent lighting (50 Hz ballast)	100 Hz (small 50 Hz component)	Headache and eyestrain	Many anecdotes
Normally functioning fluorescent lighting (50 Hz ballast)	100 Hz (small 50 Hz component)	Headache and eyestrain	Double-masked study (Wilkins et al. [B116])
Normally functioning fluorescent lighting (50 Hz ballast)	32% modulation depth	Reduced speed of visual search	Two masked studies (Jaen et al. [B62])
Normally functioning fluorescent lighting (60 Hz ballast)	120 Hz	Reduced visual performance	Veitch and McColl [B107]
Normally functioning fluorescent lighting (50 Hz ballast)	100 Hz (minimal 50 Hz component)	Increased heart rate in agoraphobic individuals	Hazell and Wilkins [B49]
Normally functioning fluorescent lighting (50 Hz ballast)	100 Hz	Enlarged saccades over text	Wilkins [B112]
Visual display terminals	70–110 Hz raster	Changes in saccade size	Kennedy and Murray [B67]
Visual display terminals	~70 Hz raster		Many anecdotal reports of prolonged photophobia
Normally functioning fluorescent lighting	100 Hz and 120 Hz	Phase-locked firing of LGN neurons in cats	Eysel and Burandt [B33]
Various	Up to 162 Hz	Human electroretinogram signals at light frequency	Burns et al. [B13] and Berman et al. [B5]
Normally functioning fluorescent lighting (50 Hz ballast)	100 Hz	Inconsistent changes in plasma corticosterone levels in captive starlings	Maddocks et al. [B74]
Normally functioning fluorescent lighting (50 Hz ballast)	100 Hz	Mate choice in captive starlings	Evans et al. [B32]

Figure 4.10.10 A table displaying many of the proven effects that different lights and their frequencies have on the lives of humans and animals.

As you can see from the table above, our light-emitting diodes fall into many of these categories of frequency ranges, which can have many adverse biological effects on the human body. These effects include:

- Seizures
- Epileptiform EEG in patients with photosensitive epilepsy,

- Seizures in children with no previous diagnosis of epilepsy,
- Headache and eyestrain
- Reduced speed of visual search
- Reduced visual performance
- Increased heart rate in agoraphobic individuals
- Enlarged saccades over text
- Changes in saccade size
- Phase-locked firing of LGN neurons in cats
- Human electroretinogram signals at light frequency
- Inconsistent changes in plasma corticosterone levels in captive starlings
- Mate choice in captive starlings

4.10.11 UL Standard 60065—Standard for Audio, Video, and Similar Electronic Apparatus: Safety Requirements

The last standard that will be talked about in this document that our design will need to take into consideration is the UL standard 60065, which is the UL standard for audio, video, and similar electronic apparatus and the safety requirements that accompany these apparatuses.

This standard is an international safety standard, and it applies to electronic apparatuses designed to be fed from the mains, supply apparatus, from batteries, or from remote power feeding. Our device is a handheld electronic message device utilizing a light-emitting diode array that is fed power from a battery, so it fits within the constraints of this standard.

The devices that this standard applies to are also intended for reception, generation, recording or reproduction of audio, video, and associated signals. Our handheld electronic message device utilizing a light-emitting diode array is a device that receives and reproduces an associated signal from the user by way of either Bluetooth or the input buttons, and it generates a video signal and other associated signals.

This standard also applies to apparatuses designed to be used exclusively in combination with the previously mentioned apparatuses. Our device is also to be used exclusively in combination with one of those devices, that device being any compatible Android device that will be connected through Bluetooth.

This standard is mainly concerned with apparatuses that are intended for use in the household and similar general use, but is also covers commercial apparatuses and professional apparatuses. These concerns are perfect for our device because, our device prototype will be a household or general use item, and—if bought later—can be marketed to turn into a commercial apparatus. It is important to note however, that this standard is only concerned with the safety aspects of these mentioned apparatuses, and not their style, performance, or other matters relating to them.

Some examples of apparatuses that fall into the scope of this standard include:

- Receiving apparatuses and amplifiers for sound and/or vision.
- Independent load transducers and source transducers.
- Supply apparatuses intended to supply other apparatuses covered by the scope of this standard.
- Electronic musical instruments, and electronic accessories such as rhythym generators, tone generators, music tuners, and the like for use with electronic or non-electronic musical instruments.
- Audio and/or video educational apparatuses.
- Video projectors.

This standard also only applies to apparatuses with a rated supply voltage that does not go higher than 250-volt A.C. signal phase or D.C. supply, which our device does not exceed.

5. Design

The design of the handheld LED sign is based on the previous standards and constraints, as well as the given specifications from the sponsor that are listed in the following section. All the design choices made in this section are done so to meet these criteria. Many of the specifications are met in a roundabout manner, such as the weight specifications which only mattered once the design was complete. Other specifications, such as the handle design, were handled as a separate task from the main device design because it did not affect the functionality of the rest of the device but did affect the weight specifications.

The electrical requirements were made to meet the base functionality requirements of the device. The use of buttons for control and a switch for power, along with the button placements on the back of the device near the handle, meet accessibility requirements. Battery charging circuits meet battery requirements. A USB connection and Bluetooth chip meet connectivity requirements. The refresh speeds of the screen which are controlled by the clock rate of the microcontroller along with the refresh speeds of the internal components meet display requirements.

All of the electrical printed circuit board design choices are made with an understanding of the software and microcontroller functionality. With 64 lights per row to refresh, the refresh rate of the device could be too low, so software and electrical design choices were made to alleviate this issue by allowing certain lights to be loaded in parallel. Other software design choices, such as a building the code in the Arduino language, were made to take advantage of some of the helpful built in functionality available.

5.1 Design Specifications

The purpose of the project is to create a handheld LED display board to easily relay information to visible parties. The client has provided a few specifications that they require the design to adhere to. The specifications include:

- A display height of at least 5 inches
- A display width of at least 10 inches
- Comfortable carrying weight – less than 2 lbs.
- Easily accessible controls – must be held and operated comfortably with one hand
- Manual brightness controls
- Ability to quick swap between 4 user-inputted, preloaded display messages
- Ability to use both a PC or Android phone to input the 4 messages that will be stored on the device, over USB and Bluetooth
- Rechargeable battery

This section will go over each specification and how it is being met, and will also document the design choices that exceed the required specifications.

5.1.1 Height Specification

Both the display height and width are constrained by the types of LEDs chosen for the display. As discussed in the research section, we chose 1206 red LEDs. These LEDs are 1.6mm x 3.2mm. To avoid contact between two LEDs, they will be spaced 1mm apart. They will also be spaced uniformly in both the height and width direction to make sure the display looks even and not compacted in any way. Therefore, there will be one LED every 4.2mm. For a height 5 inches, there can be a maximum of 30 LEDs. For a width of 10 inches, there can be a maximum of 60 LEDs.

These numbers technically fit the design constraints, but we took the liberty of adding slightly more width and height to the display to fit two more LEDs vertically and four more LEDs horizontally, for a total of 64x32 LEDs. With an extra quarter inch to either side of the printed circuit board, this makes for final dimensions of 11in x 5.5in. The reason for the extra LEDs is to make full use of the onboard circuitry as well as the data constraints present with the microcontroller. As most electrical components come with connections in powers of two, with eight being the most common incremental size, it is prudent to construct the display to be divisible by eight to allow for ease of scalability and maximized efficiency.

5.1.2 Weight Specification

The device is required to be comfortable to carry by someone with a single hand. They should be able to hold up the sign for short bursts easily. To that end, the requirement of being under 2 lbs. was made. To accomplish this, we decided to go with surface mount LEDs on a printed circuit board, with the other elements located on the back and to the sides of the board. The whole board will be wrapped in an aluminum housing with a plastic handle bolted to the bottom. These elements are all lightweight, coming well under the 2 lbs. limit.

5.1.3 Ergonomic Specifications

The device is required to be easily controllable by the hand that is holding it. We accomplish this by placing all the control button on the rear side of the device, near the handle. The buttons can be reached with a thumb while holding the handle without causing the user discomfort. The buttons themselves are placed with at least an inch between each button, allowing for each one to be pressed separately without the user worrying about pressing two buttons simultaneously. The handle itself is a 3D printed plastic with a pistol grip design, which allows for the user to orient the device without having to see the direction which it should be facing.

5.1.4 Control Specifications

The device must allow the user to quick swap between 4 preset messages, as well as control the brightness and connect to Bluetooth. These controls are accomplished with the use of seven buttons on the backside of the device. The buttons are each wired to the voltage source and to their own unique input pin on the Arduino board. When a button is pressed, the input pin will register a voltage change which will then throw an interrupt and execute the associated command. The detection will exclude any simultaneous button presses beyond the first, and will ignore detection of new inputs until any commands currently in execution are complete. Pressing a message preset button will refresh the display, load in the message from storage, and

5.1.5 Communication Specifications

To receive new messages to display, the device must connect to an external computer that can send messages to be stored. According to the specifications, this must be accomplishable by both a PC and an Android application. Therefore, the board needs to have a USB port for PC connectivity and a Bluetooth Low Energy (Bluetooth LE) chip for phone connectivity. The USB port will be hardwired to input/output pins on the microcontroller, as well as the Bluetooth LE chip. When there is a new message to be stored on the board, the USB or Bluetooth connection will signal that it is ready to transmit and wait for the microcontroller to reply. Once it sends a transmission accept, USB or Bluetooth connection will send the message number to be stored, and then the message through serial input, one character at a time. The input messages will be no longer than 120 characters. Once completed, a message termination signal will be sent, which will allow the device to return to normal operation.

5.1.6 Battery Specifications

The battery will be a lithium-ion flatpack battery that rests along the upper back side of the printed circuit board, above the buttons. It will be attached to the printed circuit board with double-sided adhesive Styrofoam. The battery itself will output a 9V charge and should last at least 2 hours of constant display time. It will be chargeable through the USB port. The device will always run off the battery, even while it is plugged in and charging. The microcontroller has EEPROM memory, which allows it to retain its data even if the battery were to fully discharge.

5.1.7 Additional Specifications

Outside of the specifically requested specifications, there are a few more that the team feels appropriate to include. Starting with the brightness specification, there are clear tradeoffs with the color of LED to use. Blue LEDs are significantly brighter than red LEDs, but are also more expensive as well as having a higher drain on the battery. For this reason, we chose a low brightness specification to allow for red LEDs.

5.2 Electrical Design

Designing the hardware for a LED display with a significant amount of user friendly functionality poses a larger challenge than someone might originally think. LED displays are common part of the everyday world, but designing one from the ground up requires research into many implementation methods. This is also true for each subsystem, as there are many ways to achieve similar results, but only a few that can be considered optimal. This section will cover each section of the hardware of the printed circuit board and why a certain implementation method was selected.

5.2.1 LED Matrix

The LED Matrix consists of 2,048 red 1206 LEDs mounted on the surface of the board in a 64 wide by 32 high rectangle. These LEDs are placed in the center of a box 4.2mm by 4.2mm to allow for space between the contact points. Even though the LEDs are only 3.2mm by 1.6mm, the spacing between them should be even in all directions, thus resulting in an even 4.2mm in every direction. These LEDs will be wired together on the anode side by row and the cathode side by column. Each row will connect to a transistor connected to both the ring counter of shift registers and a 5V voltage source, with each row getting their own connection through a transistor to a shift register. Each column will connect to the serial input array of shift registers, with each column getting their own connection.

5.2.2 Row Selection Ring Counter

A ring counter is a series of shift registers connected in a line. The unique functionality of a ring counter is that the serial output of the final shift register is wired to the serial input of the first shift register. In doing this, and feeding only a single on bit in the first shift register before returning the input voltage to LOW, we create a series of interconnected shift registers that always has only a single output line set to HIGH. To move that bit into the next spot in the shift registers, all that is needed is to pulse the clock pin on all the registers at once. The purpose of using a ring counter in this way is to control the row selection of LED matrix with as few pins and commands from the microcontroller as possible, and therefore as efficiently as possible.

The serial input pin is wired to two different inputs, only one of which is ever on at the same time. The first is already mentioned to be the serial out of the last shift register. The second is an output pin on the microcontroller. The reason for this is to reset the ring counter when it needs to return to the first row. When the ring counter needs to be reset, it receives a pulse to the clear command along with a clock pulse. Then it will receive another pulse to the serial in from the microcontroller as well as a LOW pulse from the serial out of the last shift register which will be overridden by the HIGH microcontroller pulse. After a clock pulse, the first bit in the first shift register will be set to 1, and the microcontroller will return the output to serial in to LOW. When the 1 bit rolls around to the serial out of the last shift register, it will override the LOW output of the microcontroller and set the first bit of the first shift register back to 1.

5.2.3 Transistors

The purpose of using transistors connected between the ring counter, the common collector voltage (VCC), and LED rows is to create a third floating state beyond the binary HIGH/LOW voltages providable by out pins. To keep rows of LEDs beyond the desired row from turning on, the transistors provide that third floating state by essentially disconnecting the rest of the rows from the circuit, neither connecting them to ground or VCC. Without the transistors, the rows would simply be wired to ground and would display the correct image on the selected row, but also the inverted image of the selected row on every other row.

5.2.4 Individual LED Control Shift Registers

There are eight shift registers along the bottom of the LED array. These shift registers are connected to the columns of LEDs by their cathode. They serve to sink the input current to ground. For this reason, the input to these registers is inverted, with a 0 corresponding to an active LED and a 1 corresponding to an inactive LED. Since the ring counter ensures that only a single row is active at a time, each output of the shift register will control a single LED.

These shift registers are connected in pairs of two, with the serial out of the first shift register connected to the serial in of the second shift register. The serial in of each of these pairs of shift registers is connected to its own output pin on the microcontroller. This allows each pair of shift registers to be fed data simultaneously, with each of the other control functions of the shift registers wired to the same output pins on the microcontroller. For example, when the shift register clock pin is pulsed, the clock pulse is sent to all 8 shift registers at once.

The latch pin on the shift registers allows for their parallel output to be locked while the registers themselves are being loaded with new data. Until the pin is pulsed, there will not be any change in the output. Once pulsed, the output will be set to the current internal state of the shift registers. The value of doing this comes in keeping a row of LEDs in the on state while the shift registers are loaded instead of having to turn of the LEDs to load data.

The sequence of events for the shift registers will then begin with a data load. Two bytes of data is loaded simultaneously into each of the four pairs of shift registers synchronously. Then the ring counter is simultaneously clocked along with the latch pin on the LED control shift registers. This moves the active row while also updating the parallel data outputs of shift registers.

5.2.5 Buttons

The buttons on the device are wired to the VCC on one end and an input pin on the microcontroller on the other. When the button is depressed, contact is made between one end of the button and the other and the input pin receives a HIGH signal. This will prompt the microcontroller to execute the associated functionality with the depressed button.

An issue with most buttons is the concept of a button bounce. This can occur for multiple reasons, but the most common is a physical/mechanical limitation. When a button is very near its actuation distance, the signal from the button may rapidly fluctuate from on to off. This can be because it is very close to making contact and the electricity begins to jump across the air gap between the button and its connectors. This can also be due to the ramp up time of capacitors located inside the microcontroller.

Regardless of why, the issue must be handled, which introduces the concept of button debouncing. There are a few implementations of button debouncing, both hardware and software related. This device will be using software debouncing, so it will be discussed in more detail in the Software Design section.

5.2.6 Power Switch

The device will be using a switch to power on and off. The other possibility for this functionality was a button, however using a button adds a lot of unneeded complexity to both the circuit design and software design. A button would require a circuit that, when depressed, allows power to transmit to the microcontroller. Then, when the button is released, the power circuit would need to continue providing power to the microcontroller.

A power button would then need to also function as an off switch. The microcontroller would need to have an input pin wired to the power button to detect whether it is being depressed while the device is on. It would then have to check that button repeatedly over an interval of a few second, and if it was confirmed to be pressed, power off the microcontroller.

While all of this is technically possible, it is much simpler to employ the use of a switch, which allows the user to manually connect the circuit to the power supply. This reduces development time and cost of building the final product.

5.2.7 Battery Charging Circuit

Since our device will be using a 3.7-volt lithium-ion battery as its power source, and our sponsor's desired device has recharging capabilities, we have to design and implement a battery recharger circuit to recharge our lithium-ion battery.

According to Diksha from EngineersGarage, a battery recharge circuit is a circuit that recharges whatever kind of battery is being used as a power source by resupplying that battery with charge carriers (in this case electrons) to said battery. The way you charge the battery, how long it takes, and other factors all depend on the type of battery being recharged. This is because many different types of batteries use a myriad of different materials in their inner workings.

Regardless of what type of material a battery is using or how the battery works, all of the different types of batteries share a standard problem of constantly either overcharging or over discharging when being recharged or being used as a source of power once they have been recharged. Some types of batteries have higher or lower limits of how high or low they can be charged or overcharged, and some of these types of batteries are so sensitive with respect to the limits that they can be overcharged to, that if they are overcharged beyond their sensitive limits, then they can even explode once they exceed that threshold.

However, there are ways that these overcharging, or over discharging issues can be circumvented. In order to avoid these problems, an engineer designing his device that uses a rechargeable battery needs to design a smart circuit that will know to shut off once the battery's overcharging limit has been reached, and it also needs to know once it has reached that overcharging limit of the battery in the first place. That is why our battery recharger circuit will use an integrated circuit chip that will be responsible for charging out 3.7-volt lithium ion battery, and once the overcharging limit on our lithium ion battery has been reached, it will know to shut off the circuit and stop supplying the respective battery.

5.2.7.1 Battery Charging Circuit Design

For our device, we are designing and building a circuit that will be on-board our main device that will recharge the battery that supplies power. We will be using a 3.7-volt lithium-ion battery to supply power to the board, and in order to recharge this battery, the main part of the circuit will be the integrated circuit TP4056. The TP4056 integrated circuit will be the "brains" of the circuit, which is how it will know when the battery has reached its overcharging limit, and will tell the circuit when to shut off once that limit has been reached.

Since in our designed device we will be using a 3.7-volt lithium ion battery, that means that the maximum nominal voltage of our battery is 4.2 volts when being recharged. That means once our battery reaches 4.2 volts when being charged, the integrated circuit TP4056 will detect that the battery is fully charged and will shut down the circuit as to not overcharge the battery. The TP4056 integrated circuit detects when our battery reaches this voltage by continually measuring the terminal voltage. If lithium ion batteries are not carefully monitored this way when recharging, then they are at risk of catching fire, so that is why it is of upmost importance that the battery is monitored like how it is by the TP4056 integrated circuit.

The TP4056 is an integrated circuit specifically designed for recharging 3.7-volt lithium ion batteries. The specific name of this component is called a linear battery charger controller with constant current and constant voltage. If you add a programmable resistor to this integrated circuit, it is how it is used specifically with 3.7-volt lithium ion batteries as a recharger. The voltage that the integrated circuit outputs to charges with is set at 4.2 volts, since that is the maximum nominal voltage that the 3.7-volt lithium ion battery can reach before it is in danger of catching on fire. The current that the integrated circuit outputs to charge with is set by the designer of the circuit by adding resistors and capacitors as external circuitry to reach the desired charging current that we want for our device, in our case to match the charging current needed by our 3.7-volt lithium ion battery. However, the maximum charging current that the TP4056 integrated circuit can output is one ampere, so if a battery requires more than that to recharge efficiently, then another circuit would be required.

The TP4056 also provides internal thermal protection, current limitation, and negative charge current. The TP4056 protects the battery when it is fully charged by shutting down its outputted charger current once the battery reaches 4.2 volts. When this current is shut off by the TP4056 once the battery is fully charged, there is a chance that current can flow back from the battery into the TP4056. In order to combat this issue, an internal PMOS transistor blocks the negative charge current that would flow into the integrated circuit. Since there is an internal PMOS transistor, there is no need to add an external blocking diode to the circuit to prevent this current from entering.

The TP4056 integrated circuit comes in a small outline package (SOP) package, which is a big plus for us since it will fit well in our small portable device. Small outline packaging is a family of packaging for small outline integrated circuits (SOIC). According to Wikipedia, a small outline integrated circuit is a surface mounted integrated circuit package which occupies an area of about 30-50 percent less than an equivalent dual in-line package (DIP), with a typical thickness that is 70 percent less. They are commonly able to be found in the same pin-outs that the corresponding dual in-line package counterparts are found in. Small outline packages have pins that have spacings that are less than 1.27 millimeters, and can come in plastic small outline packaging (PSOP), thin small outline packaging (TSOP), and thin-shrink small outline packaging (TSSOP). Another upside to the design and packaging of the TP4056 integrated circuit is that is requires very few external components, just some resistors and capacitors, and requires a range of 4 volts to 8 volts input for its operation, which fits perfectly in line with our USB 5-volt DC voltage input.

The small outline packaging that the TP4056 integrated circuit is packaged in has eight pins that have the following configurations:

The first pin is named TEMP. The main function of this pin is to sense the temperature input. If the voltage input to this pin is below 45 percent or above 80 percent of the supply voltage Vin for more than 0.15 seconds, then that means that the temperature of the battery is either too high or too low. If this happens, then the charging of the battery is stopped by the TP4056 integrated circuit for safety. This function of the TEMP pin can be suspended by grounding this pin.

The second pin is named PROG. The main function of this pin is to monitor the constant charge current and provide the outputted constant charge current. This pin is used by connecting a resistor from this pin to the ground pin. The voltage on this resistor can be used to measure the charge current that is being output by TP4056.

The third pin is named GND. As in accordance with standard electronic component naming conventions, this is the ground terminal on the integrated circuit and is used for grounding in the circuit.

The fourth pin is named VCC. This pin is where the positive input supply voltage is received from. If at any time the input voltage going into VCC falls to a range within 30 millivolts of the fifth pin, which is the pin that has the voltage of the battery, then the TP4056 will enter low power sleep mode, since that means that the battery is fully charged, to protect the battery; and the pin with the battery voltage will have a current that falls to less than two microamperes.

The fifth pin is named BAT. This is the pin that is in connection with the battery, as stated in the summary of the fourth pin. This is the pin that is connected to the positive node of the battery. When the TP4056 integrated circuit is providing an output current to the battery to charge it, that charging current enters the battery through this pin. This pin also provides the voltage along with the current to charge the battery, set at 4.2 volts as previously stated.

The sixth pin is named STDBY. The function of this pin is the open drain charge status output for when the battery is fully charged. That means that this pin is pulled low by a switch that is internal to the TP4056 integrated circuit once the battery has reached its maximum nominal voltage of 4.2 volts. However, whenever the battery is not at its nominal maximum voltage, this pin is in a high impedance state.

The seventh pin is named CHRG. The function of this pin is the open drain charge status output for when the battery is in the process of being charged. That means that this pin is pulled low by a switch that is internal to the TP4056 integrated circuit when the battery is in the process of being charged. However, whenever the battery is not currently being charged by the integrated circuit, this pin is in a high impedance state.

The eighth and last pin is named CE. This pin is the chip enable output pin. This is the pin that controls whenever the TP4056 integrated circuit is operating normally and enabled, or disabled and shut down. This is a high input pin, and whenever it is high, the TP4056 will operate as normal, otherwise, it will disable the TP4056.

Another feature that is included in the TP4056 integrated circuit is automatic recharging. The way this feature works and is activated is, once the internal battery voltage drops to around 4.05 volts down from 4.2 volts, the TP4056 will detect this and start the charging cycle of the battery over again. Another way that the TP4056 integrates the automatic recharging feature is through temperature monitoring. The way this work is that the first pin, the TEMP pin, is connected to the negative temperature coefficient thermistor output that is inside of the lithium ion battery. If the voltage on this pin goes below 45 percent or above 80 percent of the supplied voltage for more than 0.15 seconds, then that means that the battery temperature is at dangerous levels of either being way too low, or way too high. Once this happens, the TP4056 will detect that the temperature is not within acceptable limits and it will shut down the recharging of the lithium ion battery.

Now for the way the TP4056 integrated circuit will be integrated and the respective circuit will be designed for its use. For starters, the TP4056 needs to have a supplied input power, with which will then be passed onto the battery that the TP4056 is charging. The third pin, ground, and the fourth pin, vcc, are the two pins that are designed to functionally interact with this inputted power source. In our case, this input power supply will come from a USB cable, which has ground and vcc terminals already internally supplied. Once this supplied voltage of 5 volts DC from the USB drops to within 30 millivolts of the voltage in the battery, the Tp4056 will turn itself off and go into sleep mode and decrease the charge current that it supplies to the battery to less than two microamperes.

Along with the supplied power from the USB to vcc and ground, a capacitor with a value of ten microfarads is connected from vcc to ground as a filter, to get rid of noise and voltage spikes that can be an issue to the integrated circuit or the battery. However, since this capacitor has a relatively large capacitance value, this requires a resistance from a resistor in series to decouple, with a value from 0.2 ohms to 0.2 ohms in order to reduce ripple voltage that would result from the supply and capacitor.

The next most important thing for bringing functionality to the TP4056 integrated circuit for battery recharging is that the chip needs to be enabled. In order to do this, the chip enable pin, pin 8 (CE), is used. To activate the chip with this pin, a high input in provided that enables the TP4056; and to deactivate the TP4056, this pin is pulled low. So, to enable the TP4056, we will connect the chip enable pin directly to the supplied voltage.

Once power has been supplied to the integrated circuit and the TP4056 chip has been enabled, the charging current that the TP4056 outputs in order to charge the battery needs to be set to the desired value according to the battery. This is where the second pin comes into play, pin 2 (PROG). This pin is what provides the outputted charging current, and it also sets this current that it provides. The charging current is set by altering the value of the programmable resistor that is connected from the PROG pin to the ground pin. This pin is always providing a constant voltage for reference, with a value of one volt (called VPROG), which is provided for this external programmable resistor. For our design, we will set the programmable resistor to provide a charging current that is compatible for our 3.7-volt lithium ion battery. When the TP4056 is in constant current mode, the voltage is set a regulated voltage of two volts. When the TP4056 is not yet charging the battery, then the voltage is set at a regulated voltage of one-fifth of a volt at this programmable pin. Whenever the TP4056 is outputting the charging current to charge the battery, that charging current being outputted is equal to 1200 times the current that is going through the programmable resistor. The voltage at this programmable resistor is used to measure the current at this pin during both constant current charging mode and recharging mode when the current is drastically reduced. These currents are calculated using this method shown here:

- Current at the programmable resistor/pin (Iprog) = (Vprog)/(Rprog)
- Charging current (Icharge) = (Iprog)(1200)

Since we are using a 3.7-volt lithium ion battery, we will be using one ampere as the value of our charging current, since that is what value is required by our battery. When a one ampere charging current is used, the programmable resistor is connected to the programmable pin (pin two), and to the ground pin. We calculated the value that we desire for our programmable resistor using the calculations shown here:

- Icharge = (Iprog)(1200)
- Icharge = ((Vprog)/(Rprog))(1200)
- Rprog = ((Vprog)/(Ibattery))(1200)
- Since our Icharge is equal to one ampere, and our programmable voltage is equal to one volt, then Rprog = ((1/1))(1200)
- Therefore, our programmable resistor requires a value of 1.2 kiloohms.

Now that all of these values have been calculated, the next step required for this circuit is to connect the TP4056 integrated circuit to the battery that it will be charging itself. To do this, the battery pin (pin five) is used. On this pin, the positive end of the battery is connected, and at this pin, the 4.2-volt regulated voltage and the regulated charging current that are supplied to the battery for recharging. Once again, as similar to the supplied voltage from USB to the TP4056 itself, we will connect a capacitor of value ten microfarads in parallel with the battery pin and ground, in order to filter out voltage spikes and noise that could cause problems to damage the circuit or the battery itself while recharging.

One of the functionalities that the TP4056 provides is the current charging status of the battery being charge with the charge pin (ping 7). In order to do this, the charge pin goes low whenever the battery is charging, and if the battery is not charging then it remains in a high impedance state. A red light emitting diode can be connected in series with this pin so that it will light up when the battery is charging, and the pin is pulled low, and once the battery is fully charged, it will be in an off state when the pin is in a high impedance state. Along with that light emitting diode, a green light emitting diode can be connected in series with the STNDBY pin (pin 6), since this pin goes low whenever the battery is fully charge and the green light emitting diode will light up. However, since in our design, this whole circuit will be enclosed inside a casing of our device, and since the current battery charge level will be displayed on our light emitting diode array message area when desired by the user, these light emitting diodes will be excluded from our design.

As for the rest of the external components for the rest of the circuit design, that is all that is required for this battery recharge circuit. This is another upside to the TP4056 integrated circuit. The TP4056 requires very few external components in order to do its job of recharging a lithium ion battery. Since it includes the features of having an automatic cut off feature once the battery has reached its nominal maximum voltage, an internal overvoltage protection and internal thermal protection, automatic recharging capabilities, and a fully charged and charging indicator to detect and display the battery level when it is connected to the battery. However, it is extremely important that the correct battery terminals be connected in the correct position, as the TP4056 integrated circuit does not have any reverse polarity protection circuit capabilities built in.

It is important to take note that the battery should not be connected to a load for discharging while it is also being recharged by the TP4056 integrated circuit. Doing this can harm the battery, drastically reducing the functional lifespan of the battery, and it can also harm the TP4056.

There are three different modes in which the integrated circuit can be operating in during charging of the battery. One of these modes is trickle mode. The integrated circuit enters this mode when the battery voltage reaches a point that is less than 2.8 volts. This mode brings the voltage of the battery into safe mode. While in this mode, the charging current reduces to 13 percent of the full-scale current. Once the battery voltage reaches above the trickle voltage, which is equal to (Vtrickle(2.9V) + Delta Trickle(0.08V)), the integrated circuit enters back into constant current mode.

When this circuit is in trickle mode, the charging current drops down and can be calculated as shown here:

1. Itrickle = 13 percent of Icharge (the charging current)
2. In the case of this circuit for our device, the charging current Icharge is equal to 1 ampere.
3. Itrickle = (13*1)/100

4. Therefore, Itrickle = 130mA

The next mode that this circuit can be in is constant current mode. The programmable pin, PROG on the integrated circuit chip, will have a constant current flowing out of it in this mode. This constant current from the programmable pin is used to charge the battery when it is in recharging mode, and it called charging current. For our needs and purposes, the charging current will be set at one ampere (Icharge), which is set by the programmable resistor at pin two of the integrated circuit chip, and the battery will be charged through that constant current of one ampere until the terminal voltage of the battery will reach its maximum rated voltage of 4.2 volts.

The next mode that this circuit can be in, and will probably be in most of the time while operational, is constant voltage mode. This mode is triggered when the battery voltage has reached the peak value of 4.2 volts. Once the battery reaches this peak voltage, it tries to exceed the 4.2 volts. Once the battery tries to exceed this voltage, the integrated circuit will not allow more current to flow through to the battery so that it does not exceed this voltage. The current while in this mode starts slowly dropping downwards by maintaining a constant voltage of 4.2 volts at the battery as you can see in this graph. A little after one hour has passed on the graph, you can see that the constant current starts slowly descending in a downwards slope, and at the same time, you can also see that the constant voltage levels out and stays at the same level at 4.2 volts at the same time as the constant current is slowly going down. And then once the charge is terminated, the constant current immediately drops down to a value of zero from a previous value of around one hundred. You can see all this in the graph below.

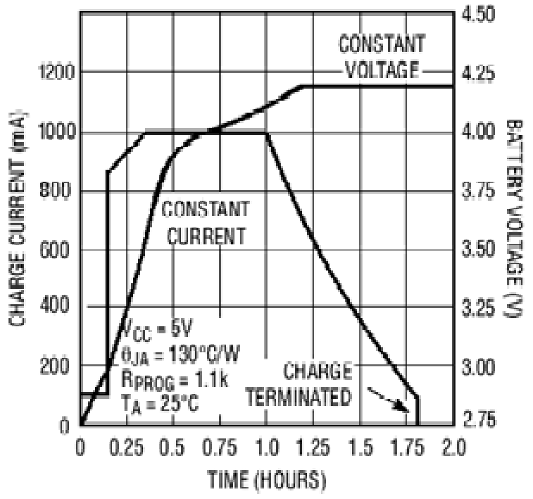


Figure 5.2.7.1 A graph showing the voltage and current over time during battery usage.

As seen in this graph, the constant current deactivates and shuts off to zero amperes once the constant battery voltage has reached the desired constant voltage of 4.2 volts.

The fourth mode that this circuit can operate in is called standby mode. In this mode, the integrated circuit automatically stops charging the current when the charging current drops to one-tenth of the preset programmed current and charging current after the maximum constant battery voltage of 4.2 volts is reached. In our case, this standby mode current can be calculated by:

1. Istandby = (1/10)*(Icharge)
2. Istandby = (1/10)*(1) since Icharge is equal to one ampere
3. Istandy = 0.1 amperes

In this standby mode, the integrated circuit will only draw a maximum one hundred microamperes current as stated in the datasheet.

The fifth, and last remaining mode that this circuit can operate in, is called shut down mode. When the programmable resistor pin is not connected to any output, and the input voltage is less than the battery voltage, then the integrated circuit will automatically be in shut down mode since no current of any kind is required, and it would have nowhere to go.

That is the general explanation of how the battery recharge circuit will operate. While switching back and forth between these five modes, the TP4056 integrated circuit makes sure that the battery will not go too low in internal voltage and die, and it also makes sure that it will not overcharge and catch fire or even explode. It does all of this automatically with several internal monitoring methods, and that is what makes this such a perfect circuit for our battery recharging needs.

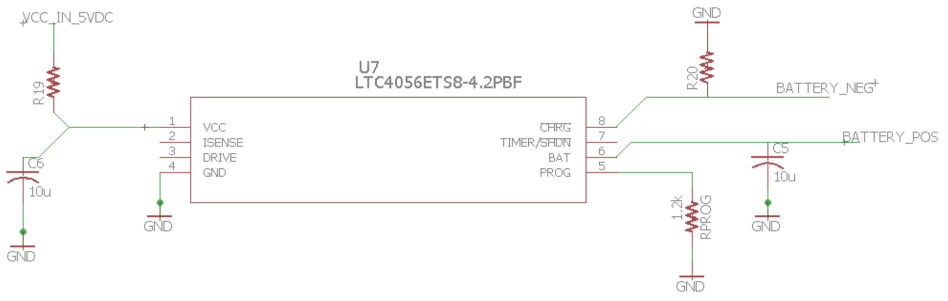


Figure 5.2.7.1

5.2.8 Bluetooth Circuit

For our device’s design, our sponsor specified having Bluetooth capabilities as one of the requirements that our design had to fulfill. This integrated Bluetooth interface had to have the capabilities to communicate with an Android smartphone so that the user could connect their Android smartphone to our device wirelessly, and then control the device with their phone instead of the input buttons on the case of the device.

While researching how to design and implement a Bluetooth circuit in our device, we found out that the best way for us to approach this task was to purchase a Bluetooth host controller interface (HCI) module that had an integrated antenna and no on-board microcontroller. It was important that our module did not have an on-board microcontroller because our device has its own microcontroller that is used as the brains of our device to control the light-emitting diode array and all the accompanying circuits and components that are required to make our device correctly work. So, our Bluetooth module will instead send its data to our own microcontroller to send instructions to the device, instead of sending itself the instructions which wouldn't allow the device to work wirelessly.

As explained in the research section, the Bluetooth host controller interface module that we decided to implement in our design is the CC2564MODACMOG from Texas Instruments. This Bluetooth host controller interface module has an integrated internal antenna, no internal microcontroller, and integrates flawlessly onto our printed circuit board, which were all enormous factors in selecting the device that fit our requirements. It is also a fully certified module for Federal Communications Commission, Industry Canada, Conformite Europeene, and Bluetooth 4.1, so we have no standards that are at risk of not being followed while using this module.

Implementing a Bluetooth circuit into our device's design meant designing a circuit that was capable of functionally outputting the standard Bluetooth protocol for sending and receiving data via a 2.4GHz wireless link. This 2.4GHz puts Bluetooth in the same frequency band that other common wireless protocols also exist, like wireless fidelity, more commonly known as WIFI. However, Bluetooth has its own specific standards and rulesets that make it different from other wireless protocols, and possibly the reason it is more well-known around the world.

The Bluetooth network that our Bluetooth circuit will be using to function uses a master and salve model to control how, where, and when the data is sent between different wireless nodes. Another standard rule that Bluetooth devices have is that they all have their own different 48-bit address, commonly known as BD_ADDR that is unique to each device; but it is usually represented as a twelve-digit hexadecimal value.

When we first try to use our Bluetooth capabilities in our device, we will have to go through the typical connection process that all Bluetooth devices go through. If two Bluetooth devices know nothing about each other at first, which will be the case when we first try to use it in our device while testing it, then one of the devices must run something called an inquiry to find the other device. While that one device is transmitting its inquiry signal, the other device will be listening for the inquiry, and when it finds that inquiry, it will respond with some of its unique personal information, such as its address, name, or other information.

Once the devices know each other in some capacity, after the inquiry has been done between the two devices, the next thing they do to connect will be a process called paging. Paging is when the two devices will actually form the connection required to communicate back and forth between each other using their respective addresses as endpoints. Once this paging process has finished and the connection has been established, then the devices will enter the actual connection state. While the devices are connected, they can be actively functioning in the Bluetooth communication, or they can be sent into a low power standby/sleep mode.

Beyond simply connection two Bluetooth devices to each other, if the two devices will be commonly connected to each other, such as one of our group member's own smartphones and our device, then these devices can do something called bonding and pairing. Once two devices are bonded and paired, they can automatically establish a connection when they are close enough in proximity to each other. When devices pair with each other, they send each other their information including their addresses, names, and profiles, and then they store this information in their memory; then they also send each other a secret pass that they use for their specific pairing for use in the future. Once the pairing is done, the bond between the two devices is made.

There are several Bluetooth profiles, including serial port profile (SPP), human interface device (HID), hands-free profile (HFP), headset profile (HSP), advanced audio distribution profile (A2DP), and A/V remote control profile (AVRCP). Since our device will be using Bluetooth to connect a smartphone that will send data to our microcontroller through a serial communication interface of either RS-232 or a UART, we will be implement serial port profile.

5.2.8.1 Bluetooth Circuit Design

One of the upsides to using the CC2564MODACMOG host controller interface module is that applying it for use in a circuit requires relatively few external components to accurately function. Since our device will not implement audio from the Bluetooth functionality, we do not need any resistors for the AUD_IN_1V8, AUD_OUT_1V8, AUD_CLK_1V8, and AUD_FSYNC_1V8 pin signals, and we can either just leave those pins as no connects or ground them, depending on what the datasheet says to do if those pins are unused.

One of the components that will be required to design this circuit will be a 0.1 microfarad capacitor. We will name this capacitor C17. This capacitor will connect between the VDD_IO pin where the input power will come from, and the ground pin. The purpose of this pin will be to filter out any noise and protect against voltage spikes that may incur during use.

Another external component we will require is another capacitor, but this one will have a value of one microfarad. We will name this capacitor C21. This capacitor will go in between the VBAT pin, where the battery will have its voltage recharged, and the ground pin. This capacitor will serve the same function as the other capacitor, to filter out unwanted noise and to protect against voltage spikes, but this time to protect the battery that is being recharged instead of the CC2564MODACMOG module.

A few more external components that will be required are four zero-ohm resistors. These resistors will be in series with the HCI_CTS_1V8 pin, HCI_TX_1V8 pin, HCI_RX_1V8 pin, and the HCI_RTS_1V8 pin. These will be inserted in case we want to add a resistor value to these signals later, or if we want to disconnect the signal on the printed circuit board.

And for the last of the external components that will be needed by us, will be for the clock that the CC2564MODACMOG module will need. An oscillator that will provide the standard 32.768kHz +/- 25ppm clock signal to the module will be needed. Along with that oscillator, a ten kiloohm resistor that will go in between the power supply being sent to the oscillator and the OE pin on the oscillator will be needed to protect it, and another capacitor with a 0.1 microfarad value will be placed between the same power supply going to the oscillator and ground, once again for filtering out unwanted noise and protecting against voltage spikes that could damage the oscillator providing the clock signal to the module. And as for the last component, another zero-ohm resistor will be placed between the clock output pin of the oscillator and the slow_clock_in pin of the module, once again in case a different resistor value will want to be placed there later, or if the clock signal wants to be disconnected on the printed circuit board after it has been implemented.

For printed circuit board layout, this modular circuit will require four layers in the printed circuit board. The top layer will contain the radio frequency and output and input signals, and this layer will be the layer that the module will be placed on and is also the layer where the signal traces will be routed. The radio frequency trace must be run on this layer of the printed circuit board.
The second layer of the printed circuit board will be a solid ground layer. The third layer of the printed circuit board will be the power layer. This layer will be used to route power traces or place power planes for the module.

The fourth and last layer of the printed circuit board will be another signal layer. This will be a second routable layer used to run signal traces that exclude the radio frequency traces that have to be done on the top layer. These four layers of the printed circuit board are shown below. The board thickness that is recommended by Texas Instruments is 62.4 millimeters, and substrate dielectric of 4.2. This printed circuit board stack-up should be based on a standard flame-retardant four (FR4) material.

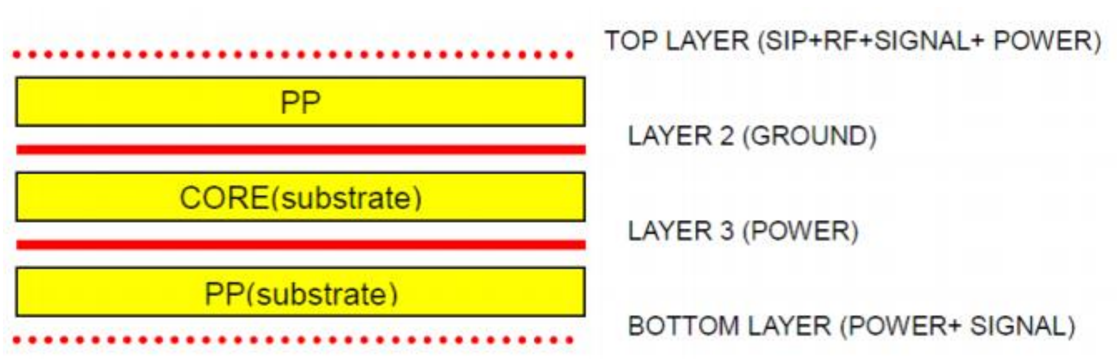


Figure 5.2.8.1 A visualization of the necessary layers of a printed circuit board to support Bluetooth.

The 62.4-millimeter thickness is composed of the following properties. The solder masks on the top and bottom should each be 0.4 millimeters; the top copper, second layer copper, third layer copper, and fourth layer copper, plus all their respective platings, should each be 1.4 millimeters, and each of the three substrates should be ten millimeters.

Our CC2564MODACMOG module includes an internal antenna, model ANT3216A063R2400A. In order to correctly wirelessly interface with our device, we should know the properties of this antenna, which are as follows. The frequency it transmits at is the standard 2.4 GHz, its maximum gain is 0.63 dBi, its average gain is -2.19 dBi, and the efficiency of this antenna is 57/03 percent.

Now, once all of these design methods and requirements have been met, our device will be able to connect with any Android smartphone that the user of our device desires, using the aforementioned connection methods that Bluetooth uses.

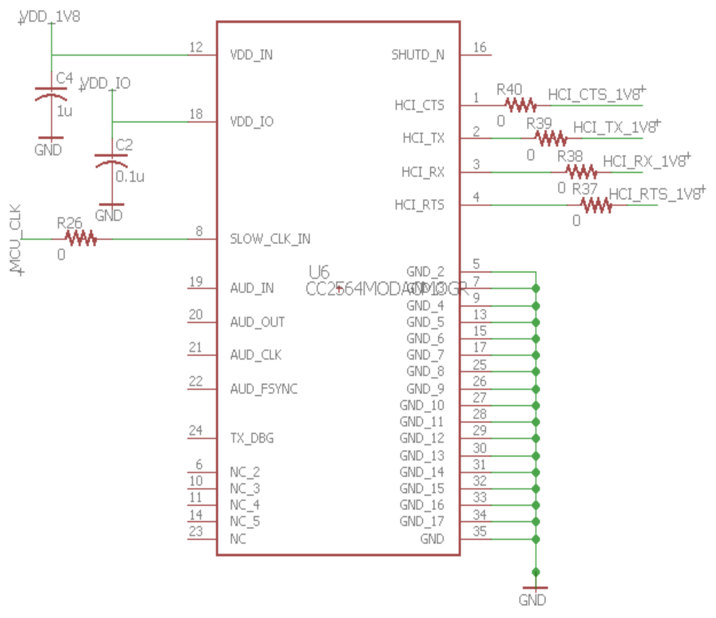


Figure 5.2.8.1 Bluetooth Circuit Schematic

5.2.9 LED control Circuit

The diagram shows the electrical schematic of the main LED matrix control logic, the components that control the operation, and the connections that are necessary. Only two of the LED matrix blocks (128 LEDs) are shown for visual simplicity. The complete matrix circuit is the same logic, only scaled up to more of the present components. As shown in the schematic, the columns of the matrix, the cathode leads, are connected to the 74HC595 shift registers. The registers are cascaded by connecting the final output of one shift register to the intput of the next register which allows the data to by sequentially transferred down to every register. The data is loaded into all of the registers with the same clock input, as shown in the unified connection of the clocks pins in the schematic. The latch clocks are also interconnected, shown using the same label for the common connections. To limit the current flowing through the matrix columns, 56 Ohm resistors are placed between the register output and the column cathodes. For standard chip operation the registers are connect to a common ground (GND) and common voltage supply (Vcc).

On the anode ends of the matrix, the rows, the refreshing circuit is placed and connected to each of the leads. As shown, the refreshing circuitis just a 74HC595 register with the last output connected to the input to cycle a logic ‘1’ continuously through the register and output lines in a loop. When a logic high is seen at the output of the register, the corresponding transistor becomes active, allowing current from the supply which is connected to the collector terminal. The current then flows through the collector and emitter to every necessary LED in the 16 LED row (64 total in the actual device). To show continuity of the matrix, the rows of the LED blocks are simply connected.

All of this data that is needed to selected the columns and keep the rows refreshed Is provided by the ATMega processor. Again for visual simplicity, the processor is not shown in the schematic but is represented instead by pin header connections at the top left of the schematic. The procesor will be connected to the same ground and supply and will contribute the uniform clock, serial data, and the latch clock. This goes to show how few pins are needed from the processor in order for this circuit to fully function. This leaves out unnessary complexity in both hardware connections and software which is ideal for swift manufacturing later in the project.

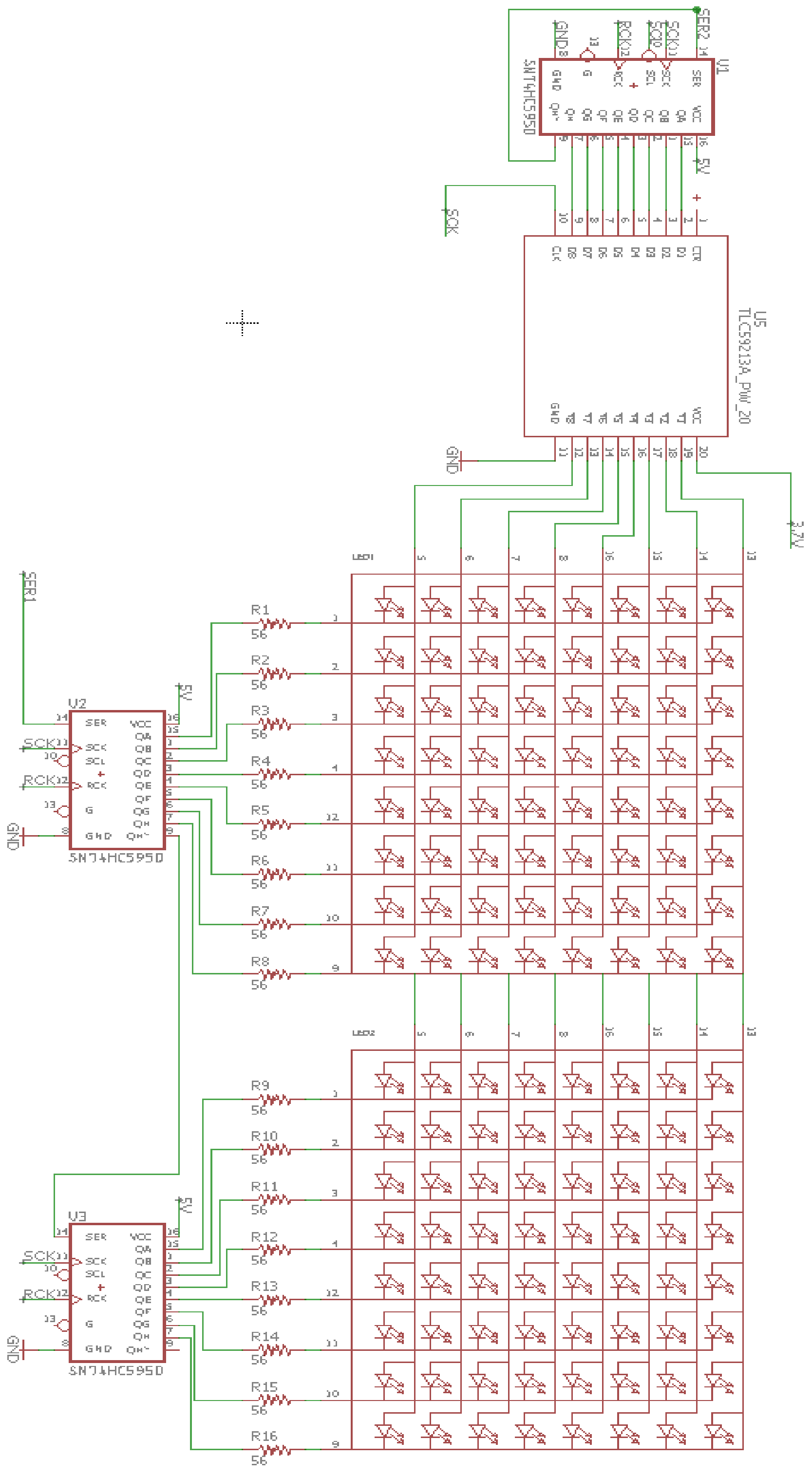


Figure 5.2.8 Eagle Schematic of LED Circuit

Section 5.2.10 Boosting Circuit

Power for the circuit is sourced from the battery, but a separate circuit is made to distribute a higher voltage than the battery can deliver. The issue with the battery is that only 3.7 V is available for the circuit, but some components need more. Such components include the ATMEGA processor, the shift registers, the Bluetooth module, and the recharging circuit, which all need over 4 V for normal operation. To accommodate the voltage need of these chips, the TPS61040 voltage booster chip was implemented, as shown in Figure 5.2.8 in the upper left. Surrounding the chip are supporting components that manage and help deliver the necessary voltage to the output. The reason that an all-inclusive type of chip - with the various inductor, capacitors, and resistors built-in – was not sought out or available for this project is application flexibility. If pre-determined values for such components of the circuit were included in the chip, there would be incredible difficulty in finding the specific part for a particular situation, and impossible to adjust. With the chosen set-up, however, the configuration can be tested, adjusted, and easier understood for the necessary task.

The voltage boosting circuit requires one inductor, three decoupling capacitors, and two external resistors. The device works by using a PWM signal to charge and discharge energy in the inductor and capacitors. When this discharge occurs, large amounts of energy in the form of current are produced. The current is then transferred through a voltage-divider configuration of resistors, which are large in resistance value (normally hundreds of kOhm) to generate the large voltage for output. The capacitors are in place to either provided energy storage for discharge to the output, or for circuit protection from input/output voltages. All of values selected for these components were generated using resources from TI, including open source circuit simulation, and datasheet formulas. After determining the specific contribution of every piece and integrating the entire system together, the circuit was completely designed in the above schematic, ready for boosting the battery voltage to a rating that is sufficient for the logic chips. The LEDs are the only components not operating on the output voltage from the booster circuit, as the 3.7 V straight from the battery will provided plenty of voltage for LED illumination. This is from the fact that the selected LEDs only require 2.2 V for standard operation, making a wired connection straight from the battery a sufficient source for voltage.

As shown in the schematic, there are five pin connections that are required on the TPS61040. The connections are labeled SW, VIN, FB, EN, and GND in the diagram. The SW pin is the output of the internal MOSFET switch, which provides the toggling polarity for the inductor to time the storage and release of the magnetic field energy. This is connected to the inductor and the anode of a Schottky diode. The FB pine is the feedback of the output voltage, back to the TPS61040 module. The module uses this feedback to determine if the output capacitor has completely discharged its energy, which requires more energy from the inductor that is being charged. Internally, the MOSFET switch will then toggle from this feedback, which then triggers the SW pin to send out voltage for power dissipation to the output. The feedback pin need only be connected to the output charging capacitor to receive voltage signal with regards to the output. VIN and GND are the input and ground pins which is simply connected to those sources accordingly. Lastly, the EN pin is the device enable pin. Since we need the device to always operate when the device is powered ON, this pin is directly connected to the source voltage.

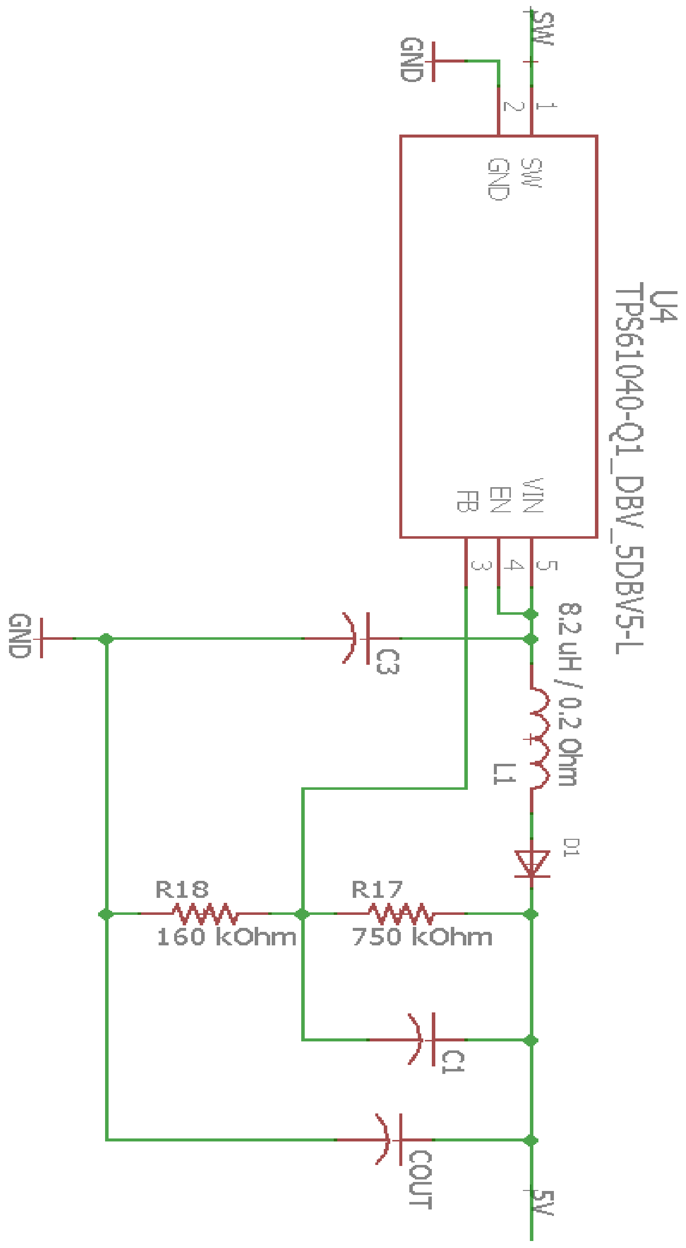


Figure 5.2.10 Voltage Boosting Circuit

5.2.11 Full Device Electrical Schematic

Figure 5.2.11 shows the schematic of the previously discussed circuits, all put together. The diagram includes the voltage boost, the battery recharge, the LED control, and the Bluetooth module circuit as well as the Atmel MCU that will be used as the device processor. To reduce confusion and schematic complexity, wiring was reduced to labeled nodes that show where the connection would be made, without actually drawing all of the physcial wires. This is sufficient since the Eagle will recognize the connections with simply names and arbitrary labels. So, especially with the 52 input/output Atmel MCU, a simplified labeling system like this makes visualization and design of the circuit much more convenient. Placement of the components are not determined by the schematic, but rather the board layout which is generated from the schematic. Once the schematic is completed, the board layout is drawn up and connected based on the connections defined in the schematic. Components are placed as desired, mainly with the intention to reduce space consumption and to make reasonable location selections.

Most components in the circuit have certain pins that share common functionality, such as VCC, GND, SER, and so on. For the case of ground, GND, this will be shared among every component, and implemented with a ground plane as one of the PCB layers. VCC will also be implemented as a layer plane, but since the 3.7 V will be used by all of the LEDs, which totals to over 2000 connections, this voltage plane would be optimal for making a plane from. The other VCC, which is labeled 5 V on the schematic will supply 5 volts to the rest of the components, such as logic chips, MCU, bluetooth module, and others. Making this a plane is unecessary and also requires an additional layer on the PCB which costs a significant amount of money to add. So instead, these connections, 5V, SER1, SER2, that share common pins will simply just be wired together, while the signals 3.7V and GND will be configured as planes that will be connected to the necessary pins.

This simplified circuit only shows a portion of the device electronics for simplicity. Realistically, there will be many more shift registers, LED blocks, transistors, and passive components. The additional components will simply be cascaded to u-scale the circuitry, and more components will be supplied accordingly. The auxiliary circuits, however, will be implemented exaclty as shown since no additional repetitions of these parts are required for full device functionality. Such circuits referred to include the Bluetooth, voltage boosting, battery recharging, and MCU modules.

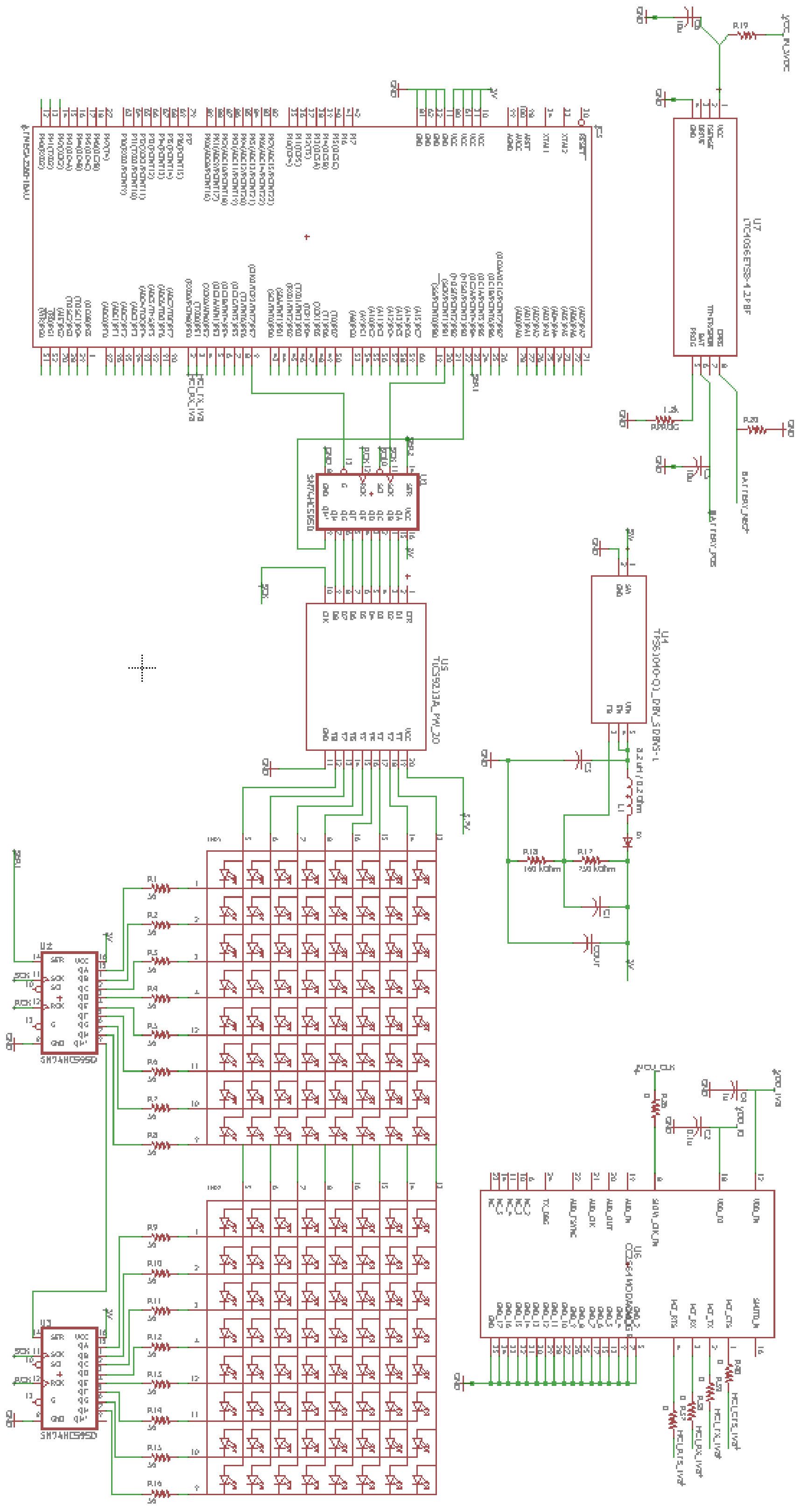


Figure 5.2.11 Full Schematic of the device

5.3 Hardware Design

There is more to the device than its internals, the outer casing and grip must also be designed. While the housing itself cannot be designed until there is a completed circuit board, saving the housing for last will not significantly impact the delivery time of the final product. The general idea for the housing is to make both it and the handle grip out of 3D printed plastic. The reason for this is two-fold: one, it is easier to take a design to a finished product using 3D printing, and two, a die does not need to be made to 3D print as would be necessary if creating an aluminum or molded plastic housing. The problem with creating a die is that doing is cost intensive and generally not a good idea if a small number of the product will be made, as is the case with this device. The following sections will detail the different possibilities for the housing design and why each one was chosen or rejected.

5.3.1 Wood Body

Members of the design time have enough experience with woodworking and access to tools that implementation of a wooden body would be fairly trivial. The added benefit of sanding out imperfections and applying a painted finish would also lend a professional look to the finished product, something akin to the appeal of rich mahogany. It also allows for easy redesign in post-production, where if something doesn’t quite fit as intended it can be cut again to suit the needs of the device.

These benefits seem great on paper, but they are unfortunately heavily outweighed by the drawbacks. The main problem being weight itself. Wood is simply too dense a material to use for a lightweight design. Even a thin sheet of particle board has a decent amount of weight, and sheets of wood that thin provide almost no structural support. The housing will need to support the entirety of the device, as well as provide some level of protection to the internals. With this in mind, wood is not a viable option for the housing of the LED sign.

5.3.2 Die-Cast Aluminum

Die-cast aluminum is incredibly lightweight. Through the use of a die and punch machinery, a die-cast aluminum housing can be refined down to a width of millimeters. In mass-production, this is a low-cost and material efficient strategy to allow for large quantities of the device to be made. The issue with this is that this strategy is only efficient in mass production, something not viable at all for low numbers of production.

Creating a die for the casting process is expensive and arduous. It must withstand high temperatures and have high structural support to endure the high pressures of the casting process. All of this together means that a single die would cost as high as $22,000. This die would make anywhere between 100,000 to 1,000,000 copies of the housing at a fairly low material price once created, meaning that each housing would only cost between $.22 and $.022. However, this project does not have nearly enough initial funding to afford a die, and is therefore considered unviable for this design.

5.3.3 CNC Drilled Aluminum

CNC stands for computer numerical control, and refers to any cutting machine that is controlled by computing and not a human. Using a CNC, it is possible to cut aluminum to the exact specifications required for the device, and then screw together the resulting pieces to create a completed housing. This process is popular is automotive parts and allows for very specific pieces to be milled in low quantities, all that is required is a digital design.

The issues with drilled aluminum are the cost of the finished product as well as its rigidity. It is very difficult to make a thin sheet of CNC drilled aluminum due to the stress caused by the drilling process. It is also not very cost effective, as each part will cost anywhere in the realm of $100 to $300 dollars. Another large issue is that curved designs still need to be manually pressed, which would add more cost in man hours to the product. For all of these reasons, CNC drill aluminum is not a viable option.

5.3.4 Extruded Plastic

Extruded plastic involves heating up plastic to a near liquid, jelly-like state and pushing it through a molding device. This device would then pressurize the plastic and conduct the heat away from it until the plastic had hardened back to its room-temperature state, leaving a perfectly molded shell. This is a one step process that can create two halves of the housing design in one mold.

The issues with this implementation are the same as the die-cast aluminum, as the process is very similar but applied to plastics. The initial cost of the mold is incredibly high, although not quite as high as the aluminum die as it does not need to be made out such expensive material. While a nice option for a mass production method, it remains just as non-viable as the die-cast aluminum for comparable reasons.

5.3.5 3D Printed Plastic

3D printed plastic refers to the process of using a computer controlled plastic extruder to place plastic a small amount at a time onto a metal plate. Over time, this plastic material would be built up into the design programmed into the printer. This process is very time consuming when measured by mass production standards, taking hours to produce a single housing. The produced plastic is also not as structurally sound as a plastic created through an extruded plastic method. This could cause cracking in the housing if the device is handled vigorously by the grip or if the housing impacts a hard object with high velocity. It would also break more easily under sheer stress if pressed hard enough.

On the other hand, there are multiple benefits to 3D printed plastic. The single most important benefit is the low cost per product when making a small number of products. The cost is limited to the cost of the plastic and the time needed on a 3D printer. Certain institutions such as the UCF college can also supply a 3D printer on rotation to students which would further reduce the cost of implementation.

The design of the 3D printed materials can also be from open-source resources. The 3D printing community has created many designs intended to be used as a platform to help garner increased interest in 3D printing. This leads to many free designs on the internet that can be used to create the housing for the LED sign. The pistol grip will be created from a modified version of a pistol design that can be found on the website pinshape.

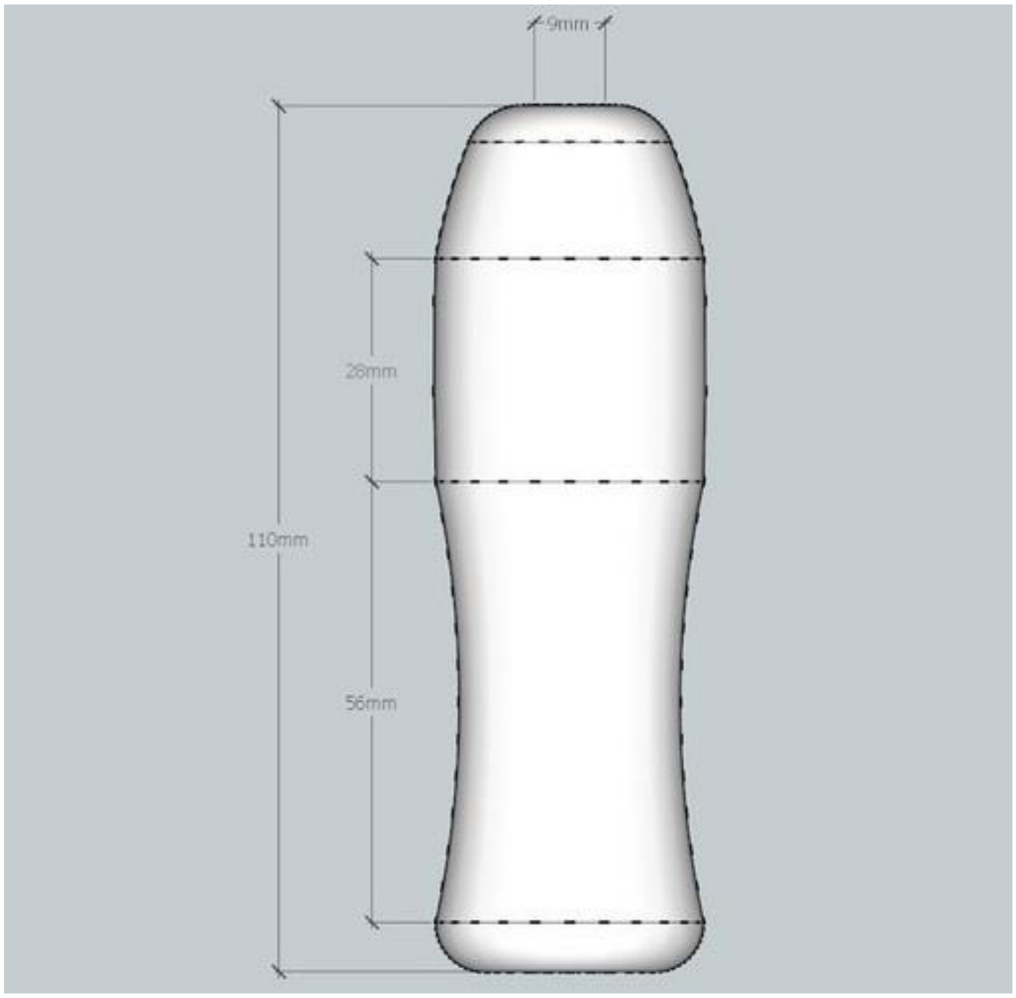


Figure 5.3.5 An example of an open source grip design available for 3D printing.

The housing itself will also be made in a similar manner by finding an open source design that is close to what the device needs, and then editing it to the exact specifications necessary. After the housing has been made to fit the printed circuit board and its buttons, the handle will be attached to the housing by a screw through the bottom of the housing into the handle. A thin plexiglass sheet will be inserted across the front of the housing in a slot that was 3D printed in its interior. The printed circuit board itself will also be screwed into 3D printed holes with shallow screws. Once every piece is in place, the final "top" piece of the housing will be placed onto the device and screwed into place with quarter-inch screws. This will keep dust from getting into the circuitry as well as keeping all of the piece inside the housing.

5.4 Software Design

There are two main elements to the software design: the input GUI and the microcontroller code. The input GUI will be created in Unity, a platform that is easily portable between both PC and Android allowing us to minimize the amount of effort needed to create the communication software. It is an object-oriented language built on the .NET framework. The microcontroller software is made in the Arduino language, an imperative language that is very close to C in its structure and is implemented in C.

5.4.1 GUI Software

For the device one of the most important parts is the graphics user interface. The purpose of this GUI is for any user to be able to edit the output on the hardware device without having any type of technical experience. If someone can use a computer and its basic applications, they will be able to use this GUI and program the device to have different outputs. The software should be simple to use yet also in depth enough to get all the requirements needed.

There will be two different GUI's for two separate operating systems. The first GUI to be made is one for Windows devices. The other GUI is for android phones which run on Linux. The front-end for both software's will look very similar but since they run on two separate operating systems, they will be made differently.

The GUI itself will make it so the user can easily program the LED display however they want. It has four character input boxes that the program will apply to the four numbered buttons on the device respectively. The user will be able to add any ASCII characters they want into these texts box. At the bottom of the screen there will be two buttons. One reads "Program Hardware" the other reads "Clear Presets" The 'Program Hardware' button sends whatever is in the texts box of the GUI over to the device so whichever button is pressed on the device, the respective string is displayed. The 'Clear Preset' button clears anything in the text boxes.

The GUI will be programmed with Unity programming language. Unity was chosen for this project because it gives the most benefits for what we are trying to achieve. Unity is cross platform, so we can program one app for both the computer desktop and an Android phone. On top of this Unity has a lot of good additional libraries that help with programming hardware and serial inputs and outputs. The unity code is written in C#, JavaScript and uses the Microsoft .NET Framework.

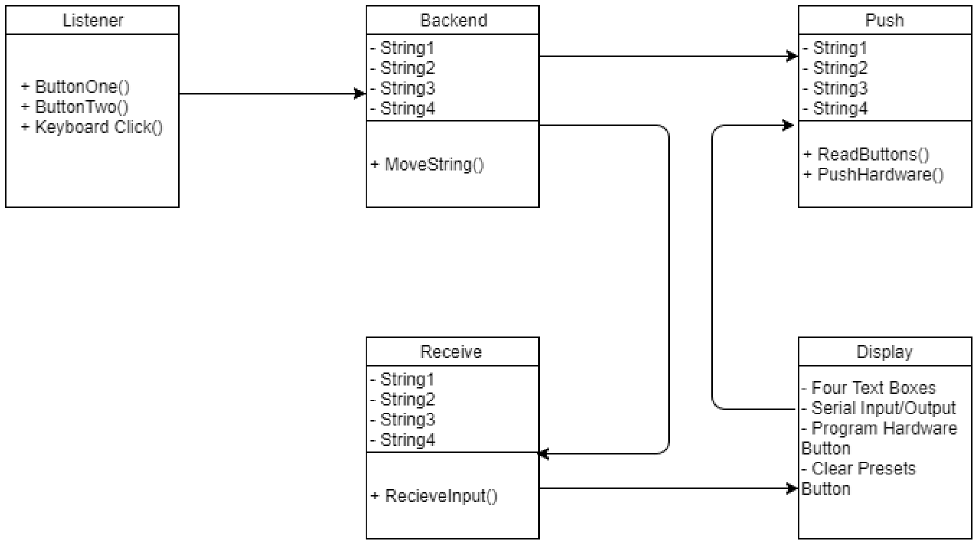


Figure 5.4.1.1: Class diagram of the GUI

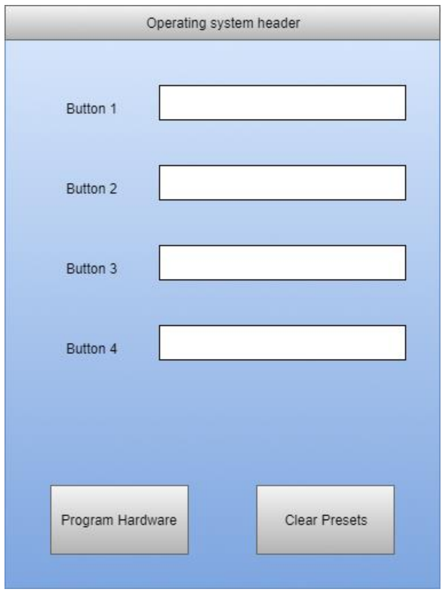


Figure 5.4.1.2: The look of the GUI on both PC and Android

5.4.1.1 C# GUI Language

C# is a programming language first released by Microsoft in 2000 within its .NET initiative. C# is an imperative object-oriented programming language. Unity, which we will be comparing C# to uses C# as its primary scripting language. It is considered to be classified in the ECMA standard list of programming languages. The ECMA was created in 1961 and was created to standardize computers in Europe. It is now a widely accept organization that develops standards for companies to meet when they create their computer systems. C Sharp got its name from the musical notation. This is somewhat similar to where C++ got its name from the incrementation syntax of “++”. The sharp symbol also looks like if you were to add four “+” symbols in a grid shape. With the name C Sharp, a sharp symbol is typically not on most keyboards, so the sharp is represented with the pound sign in its name. Since C#, a lot of other languages have taken after its name such as J# and F#.

None of our group member have previously used C sharp. However, C sharp is very similar to other languages we are familiar. The syntax that C sharp is similar includes C, C++ and Java. Some of the syntax similarities include semicolons at the end of a statement, curly brackets, they were equal signs are used with variables and using square brackets with arrays. C sharp even has a drag and drop feature for the graphic user interface which will make programming the aesthetics a lot easier for someone who is not familiar with the language. So, despite our whole group having a lack of knowledge on the language, we decided that the learning curve wouldn’t be too bad to learn.

There are a few specific design goals for C Sharp that make it stand out as its key features. C Sharp is an object-oriented language that is simple and general purpose. This is very similar to many other commonly used languages such as Java and C++. C Sharp also provides strong type checking, automatic garbage collection, and array bounds checking. These things are common software engineering principles that all newer languages should have. These also provided make the software robust, and durable. Another big feature supported is portability which we will go further into detail later in this section. Another big key feature of C Sharp is that it is suitable for different means such as hosted and embedded systems. This makes it so C Sharp can be used for very large projects that use extremely complicated operating systems but can also be used for the very small dedicated functions. The feature of being able to work with very large and complicated operating systems, although very cool, is not very useful in this project. However, the features of being able to do the very small dedicated embedded programming is very important for this project. Although C Sharp, or almost any high-level language, will never be able to compete with C or an assembly language on the efficiency of memory or processing power requirements, it is notably efficient in those areas. For obvious reasons, assembly language will be better on memory and processing power requirements, for the what C Sharp provides and being a high-level language, it is very efficient in those areas.

Although C Sharp is very similar to a lot of commonly used languages, there are some distinguishing features that make it different than these other languages. One of the main differences of C sharp opposed to other languages is its portability. C sharp is portable in the sense that it is one of the languages that most directly reflects the underlying Common Language Infrastructure. The Common Language Infrastructure is a technical standard that was created by Microsoft. This standard describes code for high-level languages that can be used on different types of platforms without having to rewrite or change any of the code. Despite this, the compilers for C Sharp don't require any type of requirements or standards which means the compilers for C Sharp could potentially compile different languages such as C++. Also unlike Java, C Sharp doesn't allow multiple inheritance. C Sharp not allowing multiple inheritance means that certain classes cannot inherit characteristics from other classes. The decision to not allow multiple inheritance was decided by lead architects on the team creating C Sharp in attempts to avoid complications in the language. Although it won't play an effect on our project another cool thing about C Sharp is that it offers support for function programming. Although C Sharp is still an imperative language, the never versions allow support for function programming techniques such as the syntax for lambda expressions.

C Sharp also supports serial input and output. This is especially important for our project. If we want our code to be able to communicate to our device there has to be some serial communication. For C Sharp to be able to have serial inputs and outputs you have to import a library at the top of the code "import System.IO.Ports;" the Library to do this is called SerialPort. For this library to work the computer must have at least Microsoft .NET 3.5 on it. To code for C Sharp and serial input/output, despite there being other compilers, must be compiled through Visual Studios.

Another very important part to C Sharp is that it is a very widely and often used programming language. Although this may not seem like a very big deal, it is always nice to use a programming language that is used by many other people. For example, if we ever reach a bump in the road we can't seem to get past, it will be very easy to find someone that will be able to help us get up and over the bump. Along with that there is a lot of support online on many different websites from and where from Stack Overflow to Wikipedia. This comes into handy because if we run into a problem, it is very likely that someone else has as well and provided solutions to the problem online somewhere. Also, along with this since C Sharp was created by developers at Microsoft, there is a lot of documentation and tutorials that are provided by Microsoft. This makes it a lot easier to learn and continue gaining knowledge of the language. This is important because as stated earlier, no one in our group as any experience programming with C Sharp.

C Sharp has a unified type system that is called Common Type System. The Common Type System (CTS) is simply just a standard on how the computer uses memory for type definitions and specific values of types. The Common Type System makes it easy to share information among different programs that use different programming languages. With the Common Type System, C Sharp separates all the data types into two different categories. The two categories that C Sharp separates their data types into to accommodate the Common Type System standard is reference types and value types. Value types always come with a default value and they can always be copied or created. Value types do not have any type of reference identity. Some limitations of value types is that they cannot have default constructors ands they cannot derive from one another. Some examples of value types include ints, and floats. On the contrary, reference types have a referential identity. This means every instance of reference type is distinct from every other instance. Opposed to value types, for reference types, it is not always possible to create an instance of a reference type or copy an existing one. Some examples of reference types are objects, strings and arrays.

5.4.2 Microcontroller Software

The microcontroller code is made in the Arduino language. Since it is an imperative language, there are no classes or other similar code structures. It will handle all the device functionality including receiving messages, displaying messages, handling control inputs, screen brightness, and battery level detection. The software will primarily function using the interrupt functionality of the microcontroller, and will otherwise proceed through a main display loop showing the currently selected message.

The purpose of using the Arduino language to build the project was to take advantage of some of the built-in functionality available. Most notably, the language provides a shift out function that allows shift registers to be filled in a single line of code, although it still takes the same amount of clock cycles as it would to manually control a shift register. This does, however, free up some of the coding complexity and debugging time that would normally be required and allows for more development in other areas, such as parallel shift register filling.

5.4.2.1 Setup Function

The setup function configures the ATmega2560 microcontroller’s initial settings. Here, each port is set to be an output or input based on its usage on the board. The function will also initialize all the data values that need to be set to some initial value. This includes but is not limited to the PWM byte, the string and bit counters, etc. The setup function will also load the stored strings from the EEPROM memory so that it can continue using the same strings that were in memory when the device was off. It will then call the message select function using the parameter of whatever message number was being displayed when the device was on, a value that was also stored in EEPROM. If this value is not set, it will default to the first message.

5.4.2.2 Message Select Function

The message select function occurs when the microcontroller detects that a user has pressed one of the message preset buttons. The microcontroller will then call the message select function using an interrupt, passing in the message number of the button that was pressed. Initially, the message select function will disable interrupts so that no other button can be received until it has finished execution. The message select function will then set the “current bit” counter to zero and then fill the display array with the LED byte representation. It will set the “current character” counter to whatever the index of the character in the string was the first to not fit in the display array. It will also check whether the user selected that message to be one line or two, and if the message does not fit entirely in the display, it will set the “scroll” Boolean to true.

The message select function will load the first set of characters into the display array. It will accomplish this using a hard-coded font array of bytes, correlating to the LED representation of each character. For each character in the message string, it will check the byte representation and place that representation in the display array with the proper spacing. The display array has enough extra memory to hold one extra character that will not be displayed but will be shifted in to the display area one bit at a time by the main loop function. After the array is full, the pin that controls the ring counter will reset the counter and put the first bit into the first shift counter. At this point interrupts will be reenabled and the function will end, returning control to the main loop.

5.4.2.3 Display Function

The display function of the software is the loop function. The loop begins by checking the “scroll” Boolean, the “line select” Boolean, and the “current bit” counter byte to see what settings have been set by message select function. These will tell the display function what character in the string is next to be loaded in, and what bit of the current character is currently in the display. If there is still a character in the “to be loaded in” section of the array, it will shift the display array one bit to the left with bitwise logic, feeding in zeroes from the right. The function will increment the "current bit" counter here to show that the array has been shifted once character.

Once the bit counter has exceeded the size of a character, it will grab a new character from the message string that is selected, using the "current character" byte to find the corrector character and incrementing the current character. The function will find the corresponding character in the hard-coded array of character representation and place it into the "to be loaded in" section of the display array. This will also reset the "current bit" counter to zero. If the current character counter exceeds the size of the string, the display function will call the message select function again with the same message that is currently set, which will clear the display, reload the current message, reset all the values, and then return to loop function.

After the display array has been managed and set to the next display iteration, the loop beings executing the code that will update the LEDs on the display. This functionality is a loop that executes exactly 32 times, once for each row of the display. It begins by decomposing a line of the display array into a serial output and feeding that to the shift registers on the board. This requires using a bit mask to single out whether a bit from a byte in the display array is a 1 or a 0. To save on clock cycles, this is done across four bytes, each two bytes apart, and puts the resulting bits into a single byte of data which includes a bit for the clock and a bit for the latch. This byte is then pushed to an entire PORT of the microcontroller using the built in Arduino port function. This way, four shift registers can be loaded at once. After doing this sixteen times, all the data of a line is now loaded into the 8 shift registers, corresponding to the 64 separate LEDs.

At this point the ring counter needs to be clocked to increment which row is being displayed. This needs to happen at the same time as the latch is hit on the individual LED control shift registers, otherwise the previous row data will be displayed on the next row for the amount of time it takes to hit the latch on the LED control shift registers. For this reason, the latch control and the clock control of the ring counter are going to be placed as bits in a byte and pushed to its own PORT on the microcontroller. This will not be the same port as the ones that control the shift register serial inputs, but the latch will be connected to those shift registers. This way, by pushing a byte to the port with the pins that correspond to the clock and latch of the ring counter and shift registers, these actions can be done simultaneously and eliminate any bleed effects that would otherwise occur in the display. While the effects of a single clock cycle may not be noticeable to the naked eye, it is worth eliminating any efficiency wherever possible.

It is here that one of the 32 loops that corresponds to updating a row of the display ends. The loop function will then move on to the next row, continuing to decompose the array into serial outputs and pushing the update commands to the board. Once it has finished executing the display update loop, the main display function will loop in full and begin once again with updating the display array, this time starting with the newly updated "current bit" counter value.

5.4.2.4 Data Request Communication Function

When a PC or Bluetooth enabled devices wants to transmit a message to be stored in the display, the USB or the Bluetooth LE chip will send a signal to the microcontroller. The first signal that will be sent upon connection will be a data request signal. The activated pin will inform the microcontroller of whether the data was requested through USB or Bluetooth, and will throw an interrupt that calls the data request communication function with the correct Boolean parameter that represents the transmission line. The function will then disable interrupts so that transmission is not broken before the entire message is sent.

The function will then change the currently displayed message to a hard-coded message stored in the microcontroller that says connection established. It will do this by calling the message select function and passing the number that represents the hard-coded message. The message will simply display a message that states that connection has been established. The message select function will be called with a Boolean parameter that states not to reenable interrupts, so that they will remain off as control is returned to the data request communication function. This message will not display until the data request is complete as control will not be returned to the main display function.

If the request was through a USB connected device, the data request will be serviced over the USB connection, otherwise it will send the data to the Bluetooth LE chip to be transmitted to the Bluetooth connected device. The data being sent will be the status of the 4 preset strings in storage on the microcontroller. This is to allow the GUI to show the four strings that are already loaded to the user, as well as to allow the user to submit an empty string as a method of clearing the data stored on the microcontroller in a later function. To accomplish this, both the USB device and Bluetooth chip will be wired to serial output pins on the microcontroller. Once a transmit request has been sent, the microcontroller will first send a transmit confirmation and then begin sending each string, one bit at a time. There will be a termination character at the end of each string, and after the fourth termination character the microcontroller will send a transmit complete signal. The GUI will reply with a transmission received or an error signal.

If the microcontroller receives a transmission received signal, then interrupts are reenabled and control is returned to the main display function, which will now show the connection established message. If the microcontroller receives an error signal, it will attempt to retransmit the data in the same method as before. After five failures, the microcontroller will call the message select function with the Boolean for interrupts set to not reenable, and this time it will pass the parameter for the hard-coded communication failed function. Interrupts will then be reenabled and control returned to the display loop function. The PC or Android device will also cease sending requests at that time and will display an error message to its user.

5.4.2.5 Incoming Data Communication Function

At some point after having called the data request communication function, the system is likely to receive a signal from either USB or Bluetooth that the connected device would like to transmit new data to the display. This signal will be unique from the signal to call the data request communication function. Once called, the incoming data communication function will first disable interrupts. It will then transmit a signal over the appropriate communication channel to signal the PC or Android device that it is prepared to receive information.

The connected device will then begin sending four strings to the microcontroller, cut off at a length of 120 characters. Each string will end with a termination signal, indicating that the next string is about to begin transmission. Each string is read in and stored in the preset strings one bit at a time, filling a byte using the Shift In functionality of the Arduino language. That byte is then added to the string array that is being filled until it is full or complete. Once all four strings have been transmitted, the PC or Android device will send a transmission complete signal.

If the microcontroller has received four strings and the transmission complete signal, it will reply with a transmission accepted signal, reenable interrupts and then return control back to the main display function. If something was wrong with the transmission, the microcontroller will send a transmission error signal. This will prompt the PC or Android device to begin the process again, starting over with a transmission request signal. The process will continue from there, and if it is subsequently successful, it will reply with a transmission accepted signal, reenable interrupts, and return control back to the main display function. Otherwise, this process will repeat for a total of five iterations. After the fifth attempt, the process will call the message select function, passing in the communication fail hard-coded message. It will then reenable interrupts and return control to the main display function. The GUI will also output a communication fail to its user.

5.4.2.6 Bluetooth Connect Function

If a user presses the Bluetooth button, the system will call the Bluetooth connect function. This function will first disable interrupts, then will signal the Bluetooth chip to establish a connection. It will then reenable interrupts and return control to the main display function. The Bluetooth chip will wait until a device connects to it or the search duration times out. This will only take a few seconds.

If a device is not found, the Bluetooth chip will do nothing and return to a dormant state. If a device is found, it will establish the connection, but it will not signal the microcontroller. To execute commands, the Bluetooth chip can signal the microcontroller with any of the signals that call the functions present in microcontroller code. This will likely happen immediately after a connection is established, as the connected device will likely need to have data sent to it regarding the state of the four preset strings stored in the microcontroller.

5.4.2.7 Brightness Control Function

When one of the brightness control buttons are pressed, the system will call the brightness control function. This function will begin by disabling interrupts. It will then call the pulse width modulation (PWM) built-in Arduino function and pass the PWM byte stored in the code after incrementing or decrementing it. This byte, originally set to 127, will be incremented or decremented by sixteen depending on the brightness button pressed.

If the increase brightness button was pressed, the Boolean for increase will be passed to the function as true. If the decrease button was pressed, the Boolean for increase will be passed to the function as false. The PWM value cannot go below 16, and can also not go above 255. In changing this value, the microcontroller signal that controls the power to the LEDs will be output at a different frequency.

PWM functionality is built into the Arduino control software. It allows certain pins of the microcontroller to output a signal that is on only for a certain percentage of time defined by the value passed into the function divided by 255. For example, in a period of 1 second, with a value of 127 passed to PWM, the signal would by HIGH for 50% of the time and LOW for the other 50%, at a refresh rate of about 500 Hz. This allows for the microcontroller to control the brightness of the LEDs with “time spent on” rather than doing so by changing the voltage level provided to the LEDs. This also happens asynchronously to the rest of the code execution, therefore not affecting the rest of the device functionality.

5.4.2.8 Button Debouncing Logic

Each button on the display is susceptible to a condition known as a bounce. A button bounce occurs when a button is detected to have been pressed multiple times due to physical and mechanical issues with the reality of a button press. The fix for this is a button debouncing solution implemented into the main display loop. The main display loop will then go on to call the appropriate function corresponding to the button press.

The way the logic works is by checking if a button press has occurred twice over a time interval. This is accomplished with a series of if statements for each button inside the beginning of the main display loop. For each if statement, there is a data value unique to the button corresponding to the if statement. This value is an unsigned long that will hold the timer value at the time it is entered. The if statement will also set a “reading” Boolean associated with its button that states that its button is currently being read.

The next time it goes through the loop, it will check if the reading Boolean is set, and if so, it will check if the difference between the current time and the time that was stored on the last loop is greater than the desired latency between push button detections. This latency should be appropriate to the amount of time a human would normally push a button, which for this code a value of 50 milliseconds is being used. If both are true, it will then check button and see if it still registers as depressed. If the button is still pressed, the system will call the function related to the button press. If not, the function will not be called. In both cases, the state of the button “reading” Boolean back to false.

5.4.3 Data Structures and Memory

The data in the microcontroller is stored in several ways. Some if it is meant to be hardcoded and never changed, while some needs to be modifiable by the user at some point after the code is loaded onto the board. This data inputted by the user also needs to be retained after the system is powered off, so it cannot be stored in standard RAM memory. For this purposed, the system will utilize its EEPROM memory.

The EEPROM memory available on the ATmega2560 is 512 bytes. For that reason, the character input strings are limited to 120 characters each. This allows for maximum utilization of the EEPROM memory while leaving some bytes for other storage reasons. Namely, storing thing like the previous character the device had scrolled to in a string, the value of the bit counter that it was scrolled through, whether the display was set to “scroll” or single line display, and any other data that the display would need to resume operation from the point before it was powered off.

The code itself is loaded into its own section of memory that totals about 256KB worth of space. This is much greater than the needs of the software and will likely only end up using as much as 5% of this space for the final build of the project. However, only 8KB of data is allotted for global variables. While this may seem like a lot, due to the size of the hard-coded arrays, a lot of this space will be used. A full 6% will be devoted solely to the four 120 character strings that comprise the messages. Then a huge amount of the data beyond that will be dedicated to fonts, therefore, space must be considered when developing the code.

The fonts are hard-coded byte arrays. For each style of character output on the LED display, a new look for each character must be hard-coded into the system memory. For the single line characters, each character consists of 8 bytes. This means that for a full alphabet of upper and lowercase letters, a total of 416 bytes is needed. There are also symbols, which will also consist of 8 bytes each such as punctuation. If another 20 characters are provided as symbols, that leads to a total of 496 bytes simply to hold data for single line characters.

The double line characters would need their own array of values to be put into the display array. Each of these characters would be about 48 bytes large, as they would take up 24 rows of LEDs and about 16 LEDs across. Creating a full upper and lowercase alphabet for these single line characters, as well as around 20 symbols as characters, would require 3456 bytes of data, quickly putting the global variables as large as almost 4KB.

The actual display array must also be stored as a global variable. This array is the same size as the LED matrix itself, with 1 bit representing one LED. However, the display array also has extra space at the end for a character that will be slowly shifted into the display when the “scroll” selection is set for a message. This extra space must be as large as a single line character to accommodate the possibility that the current mode is set to single line. Taking all of this into account, the base size of the array without the extra character is 256 bytes, and the extra character adds another 64 bytes for a total of 320 bytes. While the single line characters being added to the array are only 24 rows high, the display array must still have the full 32 rows represented in its data structure or the program will not function.

The total amount of data is now 4.5KB, over half the available memory for global variables. After considering several other variables used for control of the program and storage of temporary data, the total size of global variable usage is 5KB out of the available 8KB. This leaves 3KB available for other optional features. These features could include a different font for single line characters that is only 2/3 as large but fits more characters to the display in single line format. Implementing this would use up most of the remaining 3KB, leaving only about 512 bytes for other storage and totaling up to 7.5KB of used space on the microcontroller.

5.5 Control Flow Diagrams

Before writing any code, it is always important to write our flow diagrams to show how you are going to end up writing the code. It does not matter at what level the code is, whether it be high level code or assembly code it, it is imperative to draw the diagrams before diving into the code itself. This helps writing the code in many ways. Firstly, it makes it really easy to write the code once you start. It’s much easier to look at a flow chart and translate it to code than it is to try and write it off the top of your head. Drawing the flow charts also help to make sure your logic is correct before you dive into coding. It’s much easier to realize your logic is wrong before you start writing your code than once you’ve written code and you get a plethora of errors when you try to compile your code. Worst case while making the flow diagrams, if you realize your code is wrong, you can simply just start the flow diagram over. Since creating flow diagrams takes significantly less time than writing code, it’s also much easier to edit it if anything were to be wrong. Dr. Young has explained to us the importance of using these flow diagrams and how to properly do them. Whenever there is a break statement such as an if statement or switch statement, the two outcomes should meet at the bottom. It should be very clear to anyone who looks at the diagrams what exactly the code is supposed to do. The purpose of this is so that the electrical engineers on the project can tell exactly what’s happening even if they themselves can’t code it up. Also, the lines that connect different blocks should not overlap. This creates confusion and typically means there some type of logic error within the diagram.

5.5.1 Main Code Diagram

The main code is what is run on the device when it first starts. Once the power button is hit and the device gets turned on, the code starts running. The first thing the code does is initialize everything. This begins with the refresh circuit. So that nothing gets left from the last time the device was used and so the code doesn’t get jumbled up we need to initialize the refresh circuit. To do this we push a 1 into the shift registers which clears them. Next, we need to initialize the pin configuration. The code makes sure each that each pin knows whether it is going to be an output pin or an input pin. This is essential for a serial input/outputs the code communicates to the device. After the code initializes everything, it waits for an input from one of the four preset buttons. Once one of the buttons are clicked it enters the Preset Buttons interrupt. This interrupt will be explained below in its respective section. After the interrupt, the main code hits an if statement about the size of the string passed in. It asks if the input is larger than the allotted size on the LED screen. If the string input isn’t too large for the screen the code is instructed to place the text in the center of the screen. If the text is too large for the screen then the code is instructed to scroll the text across the screen. After either case in the if statement, the code reaches its last interrupt which is the check battery level. This interrupt, like the others, will be explained later in their respective sections. While the main code is running, it is always waiting for an interrupt to interrupt it. These interrupts make it possible for the code to flow normally and still be able to get functions done when they are triggered.

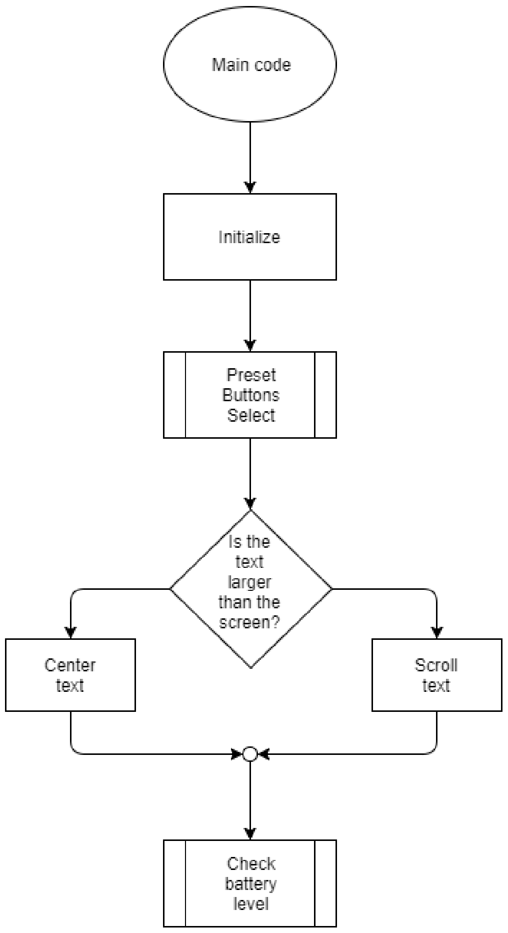


Figure 5.5.1 The control flow of the code that will run when a button is selected.

There are many interrupts that will be in the code that need flow diagrams to show exactly how they will used within the code. The interrupts although aren't in the main code with interrupt, as their name suggests, if their trigger is activated. Our code has many interrupts including brightness, power, Bluetooth, battery level, USB, preset buttons and check battery. The only two of those interrupts that will be in the main code are the preset buttons and check battery. By looking at the flow diagram for the main code, it'll be easier to understand why that's so. Below is each interrupt's flow diagram and an explanation as to how it works and why they're needed.

5.5.2 Battery Level Interrupt:

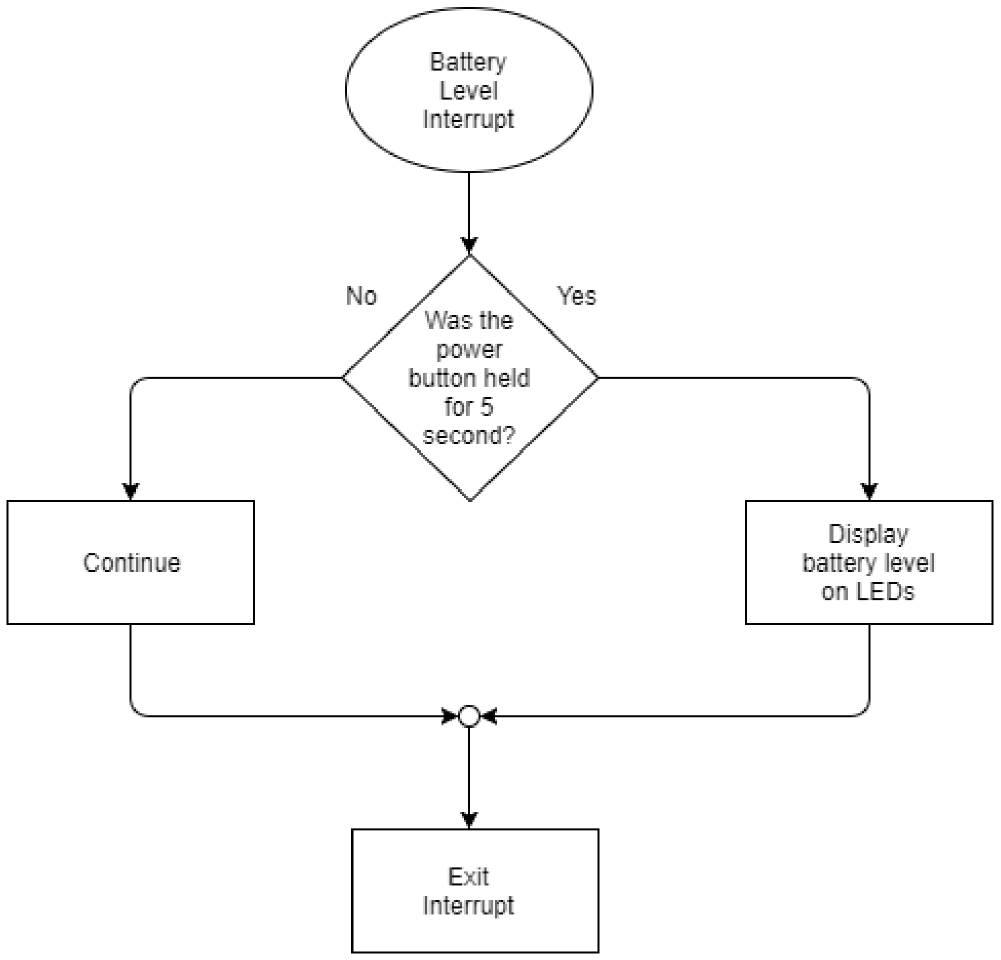

Battery Level Interrupt
No
Yes
Was the power button held for 5 second?
Continue
Display battery level on LEDs
Exit Interrupt

Figure 5.5.2 The control flow diagram for the battery check functionality, which occurs from holding the power button.

The battery level interrupt makes it possible for the user to check how much battery is left in the device. This interrupt is triggered by having the power button held down for 5 second. The first line of code in the interrupt asks if the power button has been held down for 5 second. If it hasn't been held down for that long then the interrupt should go ahead and be exited. If the power button has been held down for five seconds then the code is instructed to display the battery level on the LEDs. This is done by overriding whatever is currently being displayed on the LEDs and blinking a percentage representing the battery level.

5.5.3 Power Interrupt:

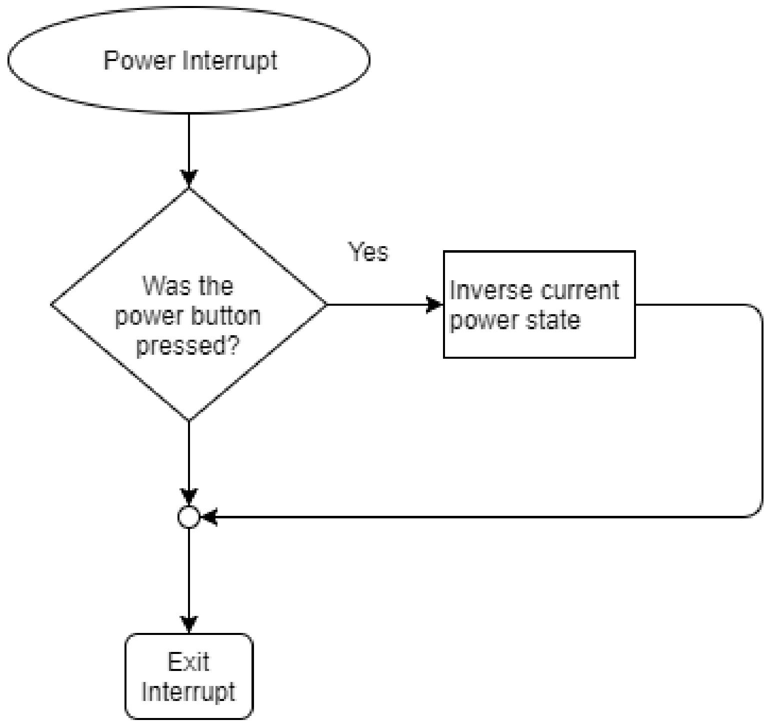


Figure 5.5.3 A control flow diagram describing the power button turning off the device.

Turning on the power via interrupt despite seeming very simple is arguably the most important part of the whole project. When the interrupt is triggered the first thing in the code is an if statement to determine if the power button has been pressed. If the button was not pressed, then the code will go ahead and exit the interrupt. However, If the power button was pressed then the code is instructed to inverse the current power state of the device. For example, if the power is already on then the code will turn it off. If the power is off, the code will turn the device on. This will be done using the XOR bitwise operator. C Language however gives us a useful tool of the "!" operator which can inverse Boolean variables.

5.5.4 USB Interrupt:

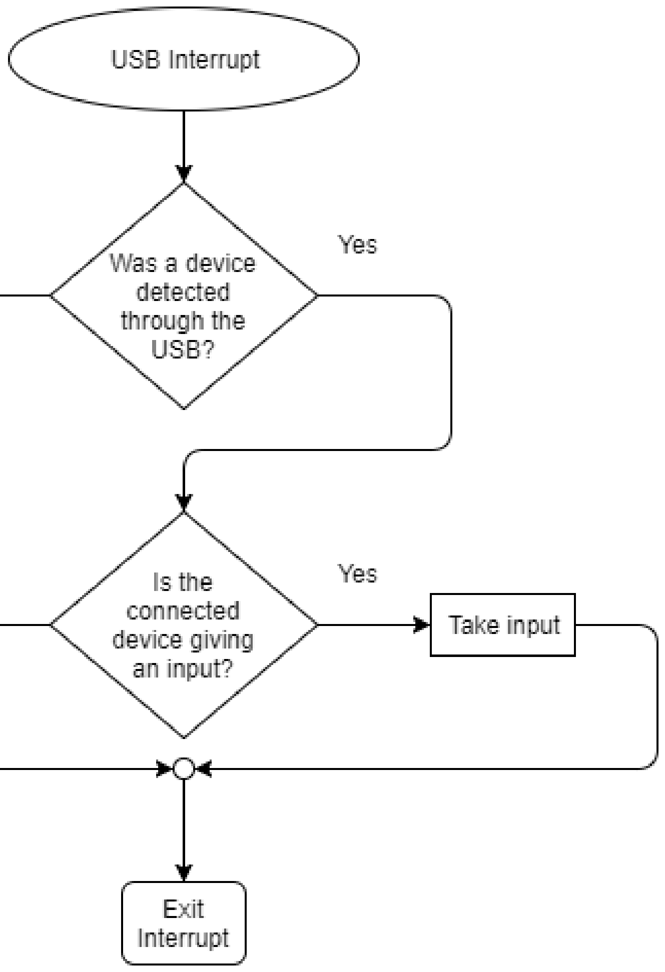

USB Interrupt
Was a device detected through the USB?
Yes
Is the connected device giving an input?
Yes
Take input
Exit Interrupt

Figure 5.5.4 A control flow diagram showing a device being connected by USB.

The purpose of this interrupt is for the USB (Universal Serial Bus) connect to the device. When the interrupt is enacted, it first looks to see if there is anything connect to the USB port on the device. This not only checks to see if there is something plugged in such as a wire but also that the wire has something on the other end, specifically a computer. If the answer to that if statement is no, then the code exits the interrupt. If the if statement is read as true, then it reads the next if statement which is asking if the device is giving an input. Again, if the answer to that is no, the code goes ahead and exits the interrupt. If the if statement is true then the interrupt takes in the input and then exits the interrupt.

5.5.5 Brightness Interrupt:

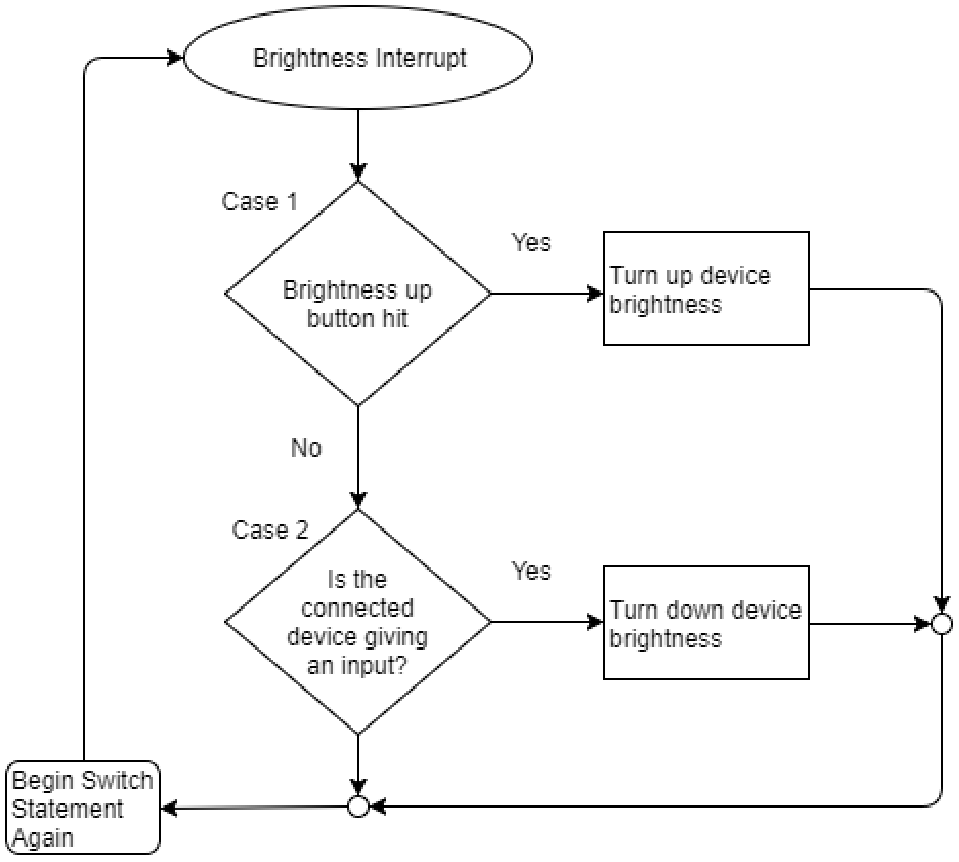

Brightness Interrupt
Case 1
Brightness up button hit
Yes
Turn up device brightness
No
Case 2
Is the connected device giving an input?
Yes
Turn down device brightness
Begin Switch Statement Again

Figure 5.5.5 A control flow diagram for the brightness button interrupts.

The brightness interrupt is much different than the others in the fact it is the only one that uses switch statements. This doesn't create too much of a problem but is definitely something to consider while putting everything together. There are two cases the conde considers while in this interrupt. The first one is 'did the button to turn up the brightness get pressed'. If it did then the code is instructed to turn up the brightness and go to the end of the switch statement. The second case is if the button to turn down the brightness got pushed. If it did then the code is instructed to turn down the brightness of the device and leave the switch case. If neither case is hit, then the code will leave the switch statement. Once the code is out of the switch statement it is instructed to wait again for another signal from the brightness buttons that would tell it to either turn the brightness up or down.

5.5.6 Bluetooth Interrupt:

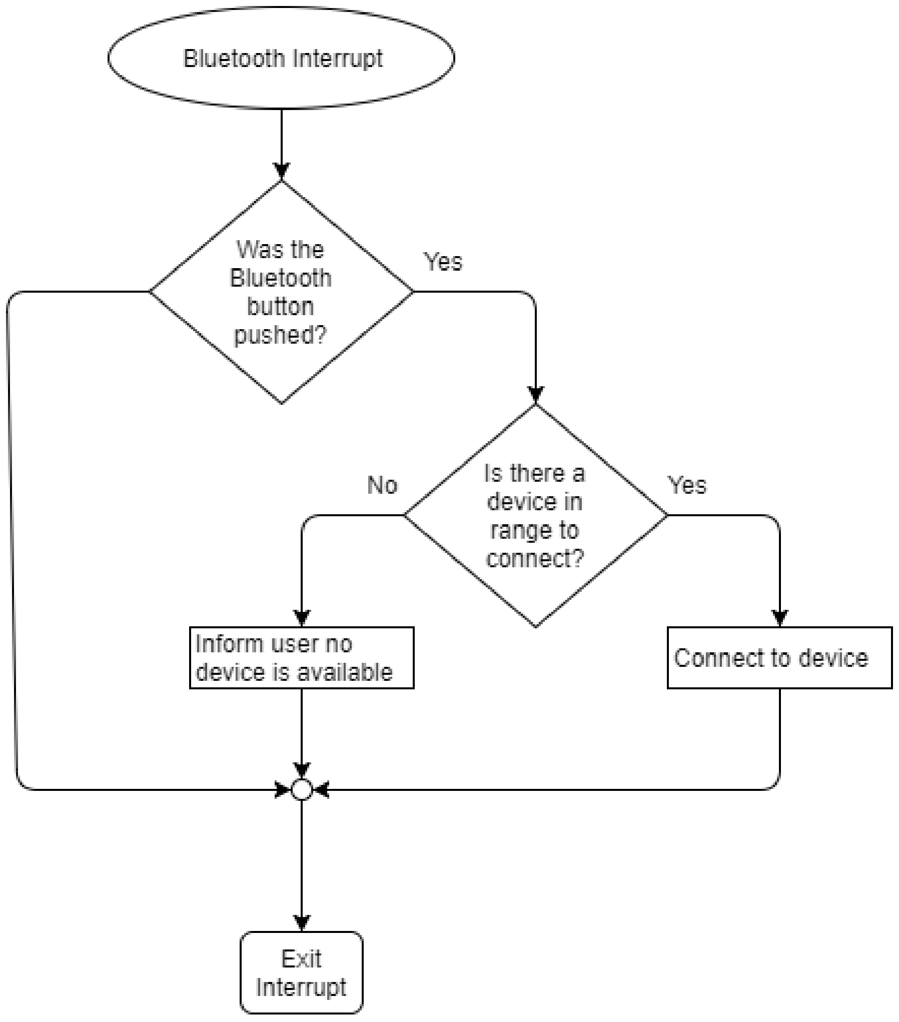

Bluetooth Interrupt
Was the Bluetooth button pushed?
Yes
No
Is there a device in range to connect?
Yes
Inform user no device is available
Connect to device
Exit Interrupt

Figure 5.5.6 A control flow diagram for the Bluetooth functionality of the device.

The Bluetooth portion of this project may arguably be the hardest, so we've tried our best to keep the programming part simple. With this, if the Bluetooth interrupt is triggered the first if statement asks if the Bluetooth button was pressed. If the button was not pressed, then the code goes ahead an exits the interrupt. If the Bluetooth button was pressed, then the code is instructed to search for a device to connect to. The next if statement in this interrupt is asking if there are any devices in range to connect to. If the answer to this if statement is no the code informs the users, there is no available devices and exits the interrupt. If there are devices in range, then the device will connect to that device. Once it connects to the device it will then exit the interrupt.

5.5.7 Check Battery Interrupt:

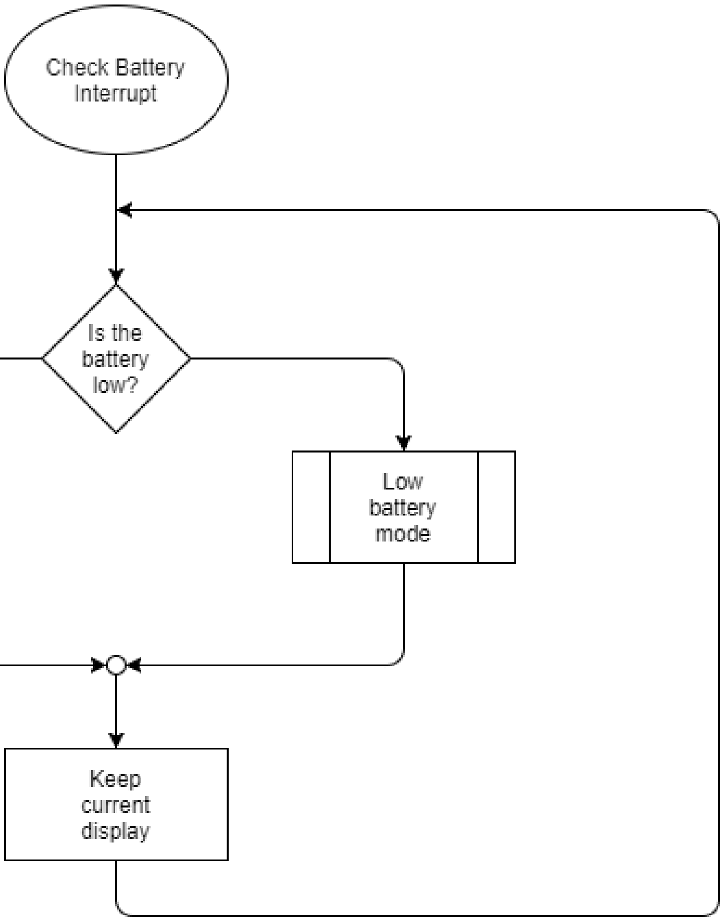


Figure 5.5.7 A control flow diagram that shows the device function of checking the battery level to warn the user.

The check battery interrupt is a very important interrupt. The interrupt is one of two interrupts called in the main code. The purpose of this interrupt is to make sure that the battery never gets too low on the device. When the code is first run it is asked if the battery level is low. If the answer is no, then then code just continues to run. If the answer is yes, then the code goes into low battery mode. Low battery mode will be a function that is called. The block diagram for the low battery mode will be elsewhere in the documentation. Once either direction of the if statement is and continues to the next piece of code. From here the code is instructed to keep the current display (unless the Button Pressed interrupt is triggered). After this the code loops back to the top to check if the battery is low again. From here the code is stuck in an infinite loop until another interrupt is triggered. This part of the code is in an infinite loop because this is the state we ideally want the device to be in the whole time it is on. In this loop the device will not do anything unless it is instructed to do so or the battery level becomes too low.

5.5.8 Preset Button Interrupt:

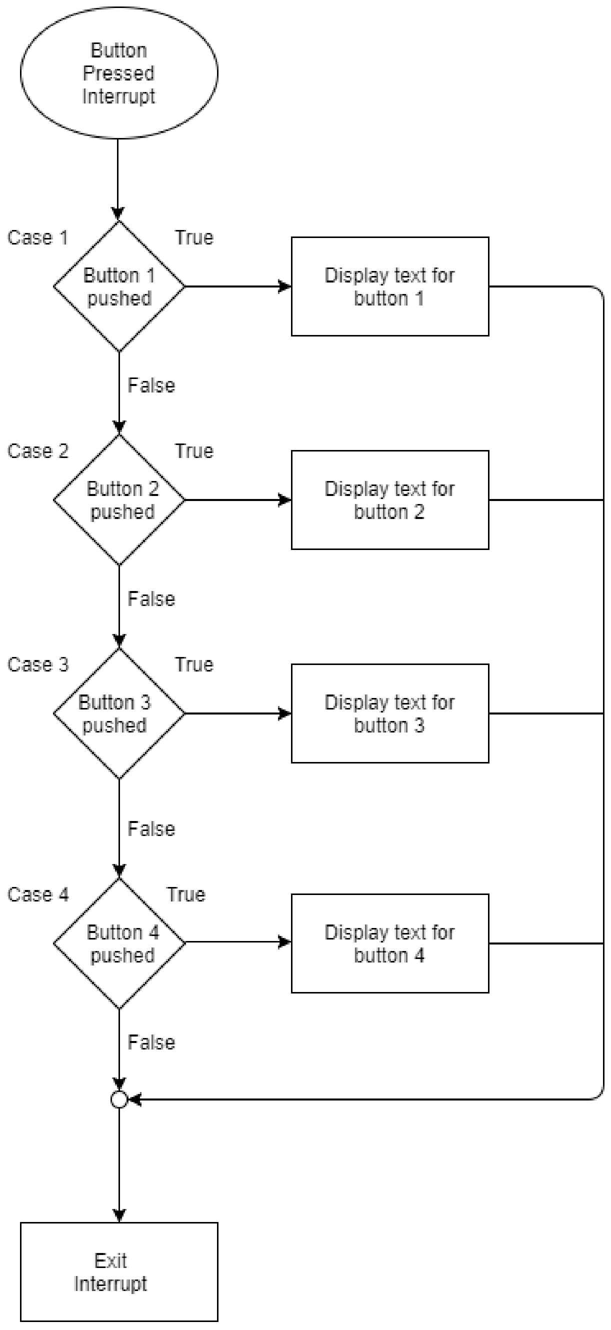


Figure 5.5.8 A control flow diagram that shows what functions the device will execute when a preset button is pushed.

The button pressed interrupt is made simply to display the text on the led corresponding to the button that was pressed. For example, if button 1 was pressed, display button 1's text, if button 2 is pressed, display button 2's text and so on. The was this will be coded is by a switch statement. This switch statement will have 4 cases. Each case number will correspond to its respective button. Once one of the cases are met or not met, the code takes us to the end of the interrupt and exits it. This interrupt is one of two that are actually in the main block of code. It’s the first code that is read right after the code initializes everything. Besides being in the main code it still runs as an interrupt. This means that if a button hit elsewhere in the code, it will jump back to this interrupt to execute it.

5.5.9 Low battery mode:

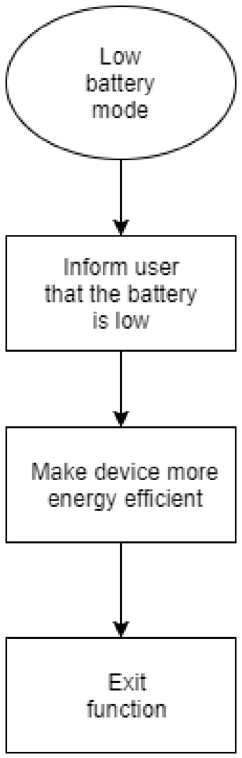

Low battery mode
Inform user that the battery is low
Make device more energy efficient
Exit function

Figure 5.5.9 A control flow diagram that shows the steps the device will take when low battery is detected.

Unlike every other flow diagram in this section, this diagram is for a function rather than an interrupt. This is the only function besides the main function that will be in the code. This is a function rather than an interrupt because it is called by an interrupt rather than being started by a trigger. This function is to put the device in low power mode. This function is called when the device realizes it is running low on power. The first thing this function does is inform the user on the LED string that the battery is coming closer to running out. One it has informed the user it beings to make the device more energy efficient. This can be achieved by dimming the lights and slowing down the blinking speed of the LEDs. Once this is done the function ends and the code goes back to the interrupt that originally called the function.

5.5.10 State Machine

Rather than sequential code there is always an option to do a state machine. Dr. Young recommended our group take a look at different state machines and as a whole decide if this is something that would be beneficial for our project. In a state machine, the code is always in one current state. One a trigger is acted upon; the code goes ahead and changes the current state to another. State machines get complicated because they're always in one state and each different trigger will take them to another state. There is a lot of examples of state machines in the real world. Some examples include a vending machine, elevators, combination locks, turnstiles, and even traffic lights.

You can implement a state machine on hardware by using a programmable logic device and controller, logic gates, flip flops and relays. Using these would change up our current schematics we have created for the LED boards.

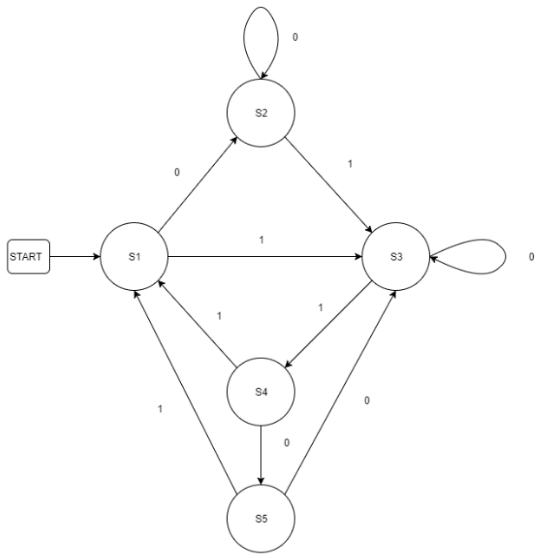


Figure 5.5.10: An example of a state machine.

The picture above shows a flow diagram for a state machine. IF you cross reference this to the flow charts of sequential code it makes it much easier to see the difference between the two styles of code. Here it shows a state for each situation in the code. Once an input, in the case of this diagram, of 1 or 0 is enacted the machine will change states, or keep the same state depending on the current state. It's almost as if once the code is started and the first state is reached, it's in a loop that the code won't leave until the device is shut off.

After comparing the pros and cons of state machines and sequential code, our group decided to go ahead with the sequential code rather than the state machines. Our group is a lot more familiar with sequential code and haven't had much experience with state machines. Along with that, creating a state machine seems a little bit too complicated and overkill for the LED display project we are doing.

5.6 Breadboard Testing

Once the design of the circuits and software have been sufficiently conceptualized, it is time to see if the theoretical designs hold up in a physical environment.

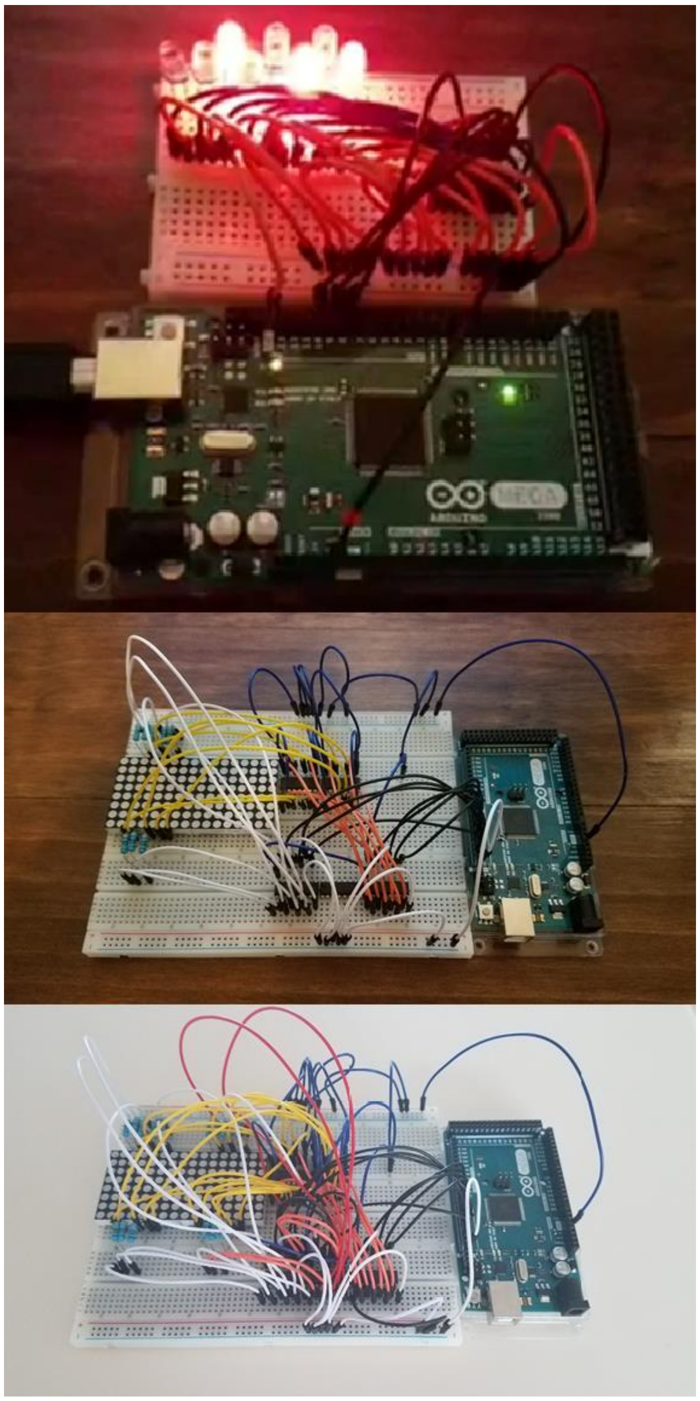


Figure 5.6: Three breadboard iterations testing the main functionality of the display. The first tested the electrical values of the circuit, the MCU and shift registers. The second tested the functionality of the transistors, the ring counter, and the high-speed refreshing of the display. The third tested parallel feeding of the shift registers.

The purpose of testing on the breadboard was to make sure that the values we calculated were matched relatively closely in the real world. Some of the circuits were omitted from these three tests, as they were considered trivial or unnecessary. These circuits include button testing, as they are simple connections to the voltage collector current and a digital input pin, as well as the power switch. Other circuits, such as the Bluetooth functionality, are tested with their own development kits, and will be included in the printed circuit board design. The last circuit that was not included is the power circuit.

The power circuit, while an important circuit, is holistically separate from the overall design of the main circuit. This is one of the reasons it was not included in the main circuit testing. The other reason is that the only way to feed power to this design is to use either a USB or DC adapter input to the Arduino development kit. This would require use to make an adapter for the battery that would not exist for any other purpose than to verify that it could output between 7V to 12V consistently. This is something that can be tested in its own circuit with a simply multimeter or oscilloscope.

It is worth noting that the power circuit will not be a simple battery, although the use of replaceable 9V batteries would significantly simplify the circuit design. Instead, it was decided that the device needed to use a rechargeable battery. The only widely available rechargeable batteries for a reasonable price are lithium-ion batteries that output 3.4V. This is well below the threshold needed to safely operate the microcontroller and the other components in the circuit, so the voltage needs to be boosted with an amplifier circuit. This can effectively double the output 6.8V, which is about the lower threshold needed to operate the microcontroller.

During this breadboard testing, it was discovered that the refresh rate of the microcontroller was sufficient to make an LED array that appeared to be constantly on despite only having a single row on at one time. Certain software functionality was also fine tuned in this phase, improving efficiency to smooth the refreshing functionality to keep the display appearing constant. There were noticeable delays during message swapping, such as changing the output message from “AB” to “BA”, but that was due to loading the new message into memory and once done, continued to display correctly.

Parallel loading of the shift registers was functionally successful, but due to the software additions necessary to accomplish this, it is hard to say whether this was an improvement to the efficiency of the device. It is for this reason that we will likely remove the parallel loading from the final design, as it increases complexity and likelihood of device failure without providing a tangible benefit.

6. Integration, PCB, and Testing

Both the breadboard prototypes will need to undergo significant design, testing, and redesign processes. The breadboard prototypes will occur in several stages, starting with a basic circuit without any attachments to understand the requirements of the LEDs and the microcontroller. The second iteration will show the LED array connected by rows to transistors and the by columns to the voltage dividing resistors and the voltage sinking shift registers. The third breadboard iteration will show the parallel functionality of the microcontroller ports, as well as the button control for the various interrupt commands.

The printed circuit board will begin without the parallel functionality as a proof of final design concept, to see if it is necessary to improve the refresh speed. If the design proves to refresh too slowly, allowing the naked eye to perceive the blinking of the LEDs, then the second printed circuit board design will have the LED matrix broken up into four sections controlled by two shift registers each, all filled simultaneously by a single port. Any further PCB design iterations will be made to fix unforeseen errors in the original design.

6.1 Prototype

To show our basic logic and electrical schematics work, we created a breadboard prototype to show that. The bread board didn't have any batteries, software, buttons etc. It only consisted of a 6 by 18 LED display and a shift register line select. We could display one letter at a time on this LED display which shows that our logic and schematic work. Having this confirmation, we can confidently continue to work on this project in the direction we originally planned on.

6.1.1 Initial Construction and Testing

While creating this product, just like any other engineering project, there must be a way to test the product to make sure it'll be able to do everything you want it to do. There are different parts and different stages as to what we must test. We first are going to test our prototype to make sure all our logic and electrical schematics are correct. After that we will test piece by piece as we finish our final product.

Starting with the prototype, we will first begin by making sure all our parts are working. We can connect the LEDs to different power sources which we shouldn't run into any problems with. After this we will set up the breadboard with multiple LEDs connects to the Arduino board. We will make sure we can output each LED we wish one at a time. It will be considered to successfully passed this test if we can go one by one and turn on each individual LED on the board. This test will not pass if there is any LEDs that do not turn on when they are intended to. If we run into a problem that a certain LED doesn't turn on when we want it to, there are multiple possible things we can do to start testing. First, we will begin to make sure each wire connected for this LED is properly attached. If all the wires look good, we will test the LED itself to make sure the LED isn't faulty. If the LED isn't faulty we will have to go ahead and check our logic given to the Arduino board and the electrical schematic getting the data to the LED. The prototype will look like the photo below, being able to show us how the basics work.

On the breadboard prototype, we should also check to make sure all the voltages are at the right amounts throughout the circuit. Each LED should have a voltage of 1.8V - 2.2V and 20-50 amps. We will use a voltmeter throughout the circuit to make sure each LED is receiving the correct amount of voltage. We will also use an ammeter to check that the LEDs are receiving the correct amount of current. As well as the LEDs, the microcontroller unit (MCU) will need to be receive 9 volts. We will use the voltmeter again to measure this.

If the prototype passes all of its it tests, we will begin to construct the actual product itself. The product itself is going to have a lot of different testing that is going to need to be done. This includes anything from the hardware, to the electrical circuits to the software, to connecting all the above.

The third breadboard prototype was made to test the functionality of the parallel input software. The purpose of this software was to reduce the number of clock cycles necessary to refresh the screen, thereby increasing the refresh rate of the LED matrix. This is only necessary if it turns out that serially feeding the shift registers is too slow and allows the human eye to see the screen refreshing. The first printed circuit board design will be built using a design like the second breadboard. After completing that design, tests will be performed to see if it is necessary to use a design like what is used in the third breadboard iteration.

These tests will include a visual test and a camera test. Text will be loaded on the device software with the intention of displaying a scrolling screen of text. If the refresh rate of the screen is too low, it should be possible to see tearing of the message between refreshing. It may also be possible to see the individual rows lighting if the refresh rate is slow enough.

If a visual test does not seem sufficient to discern a problem with the refresh rate, then a camera test could help see if the refresh rate is near to what the human eye can discern. If the LED matrix does not refresh fast enough, it may desync with the refresh rate of the camera and allow for odd patterns to appear through a video. While this is not necessarily a problem, it could point towards future issues should new features cause the software to become less efficient.

6.1.2 Battery Charger Circuit Testing

In order to test this 3.7-volt lithium ion battery recharge circuit, we will do it in a safe testing environment, like the Texas Instruments Innovation Lab on campus, or somewhere else where there is nobody around that could be harmed from our testing.

We will test the functionality of this circuit by connecting a 3.7-volt lithium ion battery to the battery terminal pins of the TP4056 integrated circuit, and monitor the voltage of the battery with a digital multimeter to see if the battery is in fact being recharged correctly, along with watching the red and green light emitting diodes that are pre-installed in the prototype module.

Before we do that however, we will measure the voltage of the battery before we connect it to the circuit, to see what it started at, to see if it is in fact being recharged. We will set our charging current output at one ampere since we are recharging a 3.7-volt lithium ion battery, and the battery has a capacity of 3,000 milliampere hours.

We will also monitor the charging current being drawn by the battery along with the battery voltage, once again being measured with a digital multimeter. We will then see how long it takes to recharge the battery, and see if it did in fact recharge the battery to its maximum nominal voltage of 4.2 volts.

6.1.3 Bluetooth Circuit Testing

The Bluetooth circuit will be tested mainly in two major ways. The first major component that will be tested for accuracy and reliability is to make sure that the Bluetooth interface can correctly and consistently connect an Android smartphone and our own handheld light-emitting diode message device together. The second major characteristic that will be tested for is to make sure that once the Android smartphone and our handheld light-emitting handheld message device are connected, that the input commands that the user gives to the device are executed promptly and correctly.

In order to test for the first major characteristic of making sure that the Android smartphone and our device can connect together via Bluetooth correctly and consistently, we will first gather multiple Android smartphones. Only one person in our group has an Android smartphone, so we will have to use other means to obtain a few more of them, to make sure that our device can handle connecting multiple different devices, and to make sure that a weird scenario doesn’t happen, like the off chance that our group member’s phone connects fine, but then no other phones are able to connect after that one is connected.

As for testing the second major component of the Bluetooth interface, we will be using those several Android smartphones that were connected to our device during the testing of the first component. Once we have verified that all of the Android smartphones can connect to our device without any issues, we will use those smartphones to issue commands to our device. We will make sure that the commands that the smartphones send are read and executed correctly by the module, and for testing purposes, our environment will just be one light emitting diode for a simple connection test.

6.2 Final Design Testing

The first and easiest test will be to make sure the frame is structurally sound. To do this once the frame is build we will place it in each group members hands and have them walk around with it and move it around as they would the actual product. If the frame or any pieces don't break it will have passed this test. If anything breaks or becomes loose, this test will be considered failed and we will have to rebuild a studier frame.

Once everything is put together the first test to verify the electrical side works is to turn on the device. If the device turns on and the LED display, then this test is considered passed. If nothing happens when we try to turn the device on we will have to look further into this situation. The most likely situation here is that one of the wires is either faulty or not correctly attached to the rest of the circuit. We could also have another problem of the battery itself either being faulty or not fully charged. To make sure it's not the battery we can simply replace it with a new fully charged battery. One of the previous solutions should be the fix to the problem. Considering the prototype worked there should be no problem with the logic or electrical circuit.

Another test that we will need to run for this product is to make sure the brightness can change. We want the brightness to be able to turn up to 100% all the way down to 0% with multiple various levels in between. To test this, once the device is on, we will hit the brightness buttons to change the display on it from all the way up to all the way down. If we can notice multiple different brightness levels on the LEDs, this test will have passed. Otherwise this test is considered failed and we will have to look further into it. The first thing we will look into is to make sure the brightness buttons are properly connected to the circuit. Next, we will make sure the PWM pins are working correctly. The PWM pins are what give the device the ability to change the brightness settings. They change the time the voltage is on or off to make the LED show different levels of brightness. If there is still a problem, we will have to check that there is 5 volts properly going through the circuit. If there is not, it could create problems for the PWM pins.

The next test that should be done it to make sure that each button outputs their respective message. To run this test, we will hit each individual button and make sure it outputs what we want it to. If it does correctly output what we want this test passes and we are done. However, there are multiple ways for this test to not pass:

1. The LEDs could display nothing. Considering all the other test we've already run, we know there's not a problem with the LEDs or power and we should start looking into the software. If this is the case it is most likely that either the device is not properly reading the serial input, or the software is not

properly giving a serial output. Another possible problem, however being much less likely, could be that the button itself is faulty and that we need to replace the button and/or the wires connecting it to the rest of the circuit.

2. The LEDs could be outputting the wrong message. For example, you could hit button 1 and the message that's supposed to be associated with button 3 is displayed. The first thing to check is the software code. We will have to make sure that each string stays consistent throughout the code and that it outputs the same string as what it intakes in the GUI for each respective string number. If that's not the problem, the next thing to check is that the buttons are wired correctly so that whatever number that button displays, it is also wired to is respective port.
3. The messages could be displayed backwards. In the English language, everything is read from left to write. It is possible for the strings to be displayed from right to left. If this is the case, there is probably something on the software side that got reversed which would be an easy fix.
4. Lastly, the LEDs could display the correct message and everything but the letters themselves could be messed up a little. There's multiple reasons this could be a problem. The LEDs themselves could be wired incorrectly and not placed properly on the grid. This would be the most tedious problem to fix. Another potential problem to this could be that the character array isn't updating properly. This will be fixed by reexamining the code and making sure the right LEDs are triggered for each letter. Lastly if the letters are being elongated, the ring counter input is probably holding two values at once.

Testing the Bluetooth is also another important part to the testing process. To test this, we will try to connect the device to Android Bluetooth phone. This test will be considered passed if the Android phone can recognize and connect to our device. If it can't do that then it does not pass this test. If we run into this problem, we will first make sure the Android phone isn't the problem just to make sure it's our device that is faulty. If it's our device that is faulty, we will first have to make sure the Bluetooth is correctly installed in the device. Once that is checked will have to make sure all the software to connect the Bluetooth is also properly in place.

Another part that will need to be tested is the USB port. This has two purposes a) to charge the device and b) to connect the device to a computer to program the device. We will test this by connecting the device to a computer via USB and making sure the computer recognizes the device and that the device begins to charge. If either one doesn't happen, this test is considered a failure. If the test fails, the first thing to check is that the USB wire isn't faulty, and if so, try another USB wire. If the same thing happens again, start looking at the USB connection. It is possible the USB input isn't wired correctly to the device.

6.3 Device Components

Below, in Figure 6.3, the main components are shown that will create the circuit for the device electronics. On the left, a white LED is shown that was used for prototyping, but also exhibits the same characteristics as the ones ordered for the final complete device. To the right of the LED, one of the resistors for the matrix current control is shown. This resistor is rated at 150 Ohms, which was selected based on calculations of 2.2 forward LED voltage and a desired forward current of 20 mA. In the upper right of the photo is the TI SN74HC595 shift registers, with a total of 16 pins for the 8 outputs and 8 data control/power supply ports. Next to that, in the middle at the top, is the NPN transistor array. This array has 14 pins but only 4 transistors internally, with 4 pins for each of the gate, source, and drain ports, and the source and ground pins. At the bottom in the center, although hard to see, is the voltage boosting chip. This chip will convert 3.7V to 5V at a maximum of 400 mA, which is plenty for this application. The chip is encased in a tape package with a protective film. Lastly, to the right on the bottom row, is a recharging module. This module is great for prototyping as connections are simply made with bread-boarding wires and charging is done through the universal mini-USB port. Depending on the space considerations of the final PCB, the chip may be extracted from the module, and our own circuit will be constructed with more space efficiency.

The MCU is not pictured here, but is shown on the Arduino development board in the previous prototyping section. The rest of the parts used are passive, which are easily obtained and applied to these important base circuits. Also, through-hole components will not be used on the PCB, so SMD packaged alternatives of the through-hole components will be acquired.

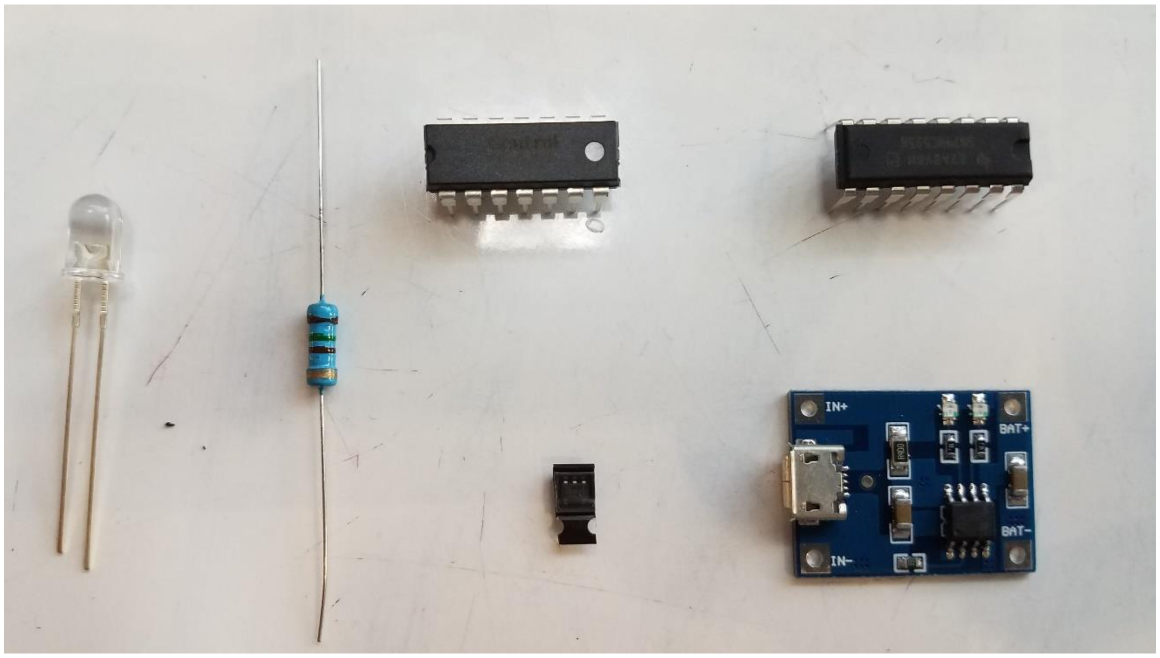


Figure 6.3: Main Components for Circuitry

6.4 Device Operation (Owner's Manual)

This product is meant for recreation uses and can be used to display messages on LED lights. It has many parts that are working together. The software GUI is used to program the hardware output. All is user friendly and quick to learn. This user manual is used to help user successfully use this product. Below you will learn how to use this device and output personalized messages on the LED output as well as troubleshooting directions and safety tips.

Safety Tips:

1. Keep device out of hands of children.
2. Keep the device away from water and other liquids that could have a reaction to the electrical circuits.
3. Do not put any other types of cords in the USB slot besides the designated charger.
4. Do not use the device if there are any exposed wires or internal parts.
5. Do not use the device while driving or operating any other type of vehicles.
6. Make sure you know how to properly use the device and read the Owner's manual before using the device.
7. Do not take the device apart or try to change anything internally.
8. Turn of the device when it is not in use.
9. Although LED bulbs run at a much cooler temperature than incandescent bulbs, do not touch the LEDs.

To begin there is a USB charging system. To power on the device, simply start by giving the device power. To do so plug in the USB charger to an outlet and the USB slot on the device. There will be a LED indicating when the device is done charging. If the LED is red, the device is currently charging and if its green then it is done charging. Once the LED is green you can unplug the device from the charger. Now you can turn on the device with the power button.

The device is preset with messages already programmed to it. However, you'll likely want to change these messages and there is a software included to make this easy to do. Plug in the device to a computer with the USB connector or connect the device to an Android phone via Bluetooth. The software will give confirmation once it has recognized the device. Once this is done you can write messages with a length of 150 characters in any of the text boxes. Once you hit the button at the bottom of the GUI "program hardware" the messages will be programmed to their respective buttons on the device. You can also clear all the buttons by hitting "clear presets". If you did this through your computer safely eject the device and remove the USB cord. The device is now ready to use.

One of the key features of this product is being able to wireless connect the device to the GUI on an Android phone via Bluetooth. To do so on the back of the device there is a Bluetooth button. After you set up the settings on your phone to look for a Bluetooth device, click the Bluetooth button on the back of the device. Once they two are connected, your phone should indicate the connection. From here you can open the app on your Android phone and see that the device is connected to the phone. To write messages to the device via the Bluetooth system you use the software the exact same way you would for the desktop version. You can write whichever messages in each of the four texts boxes. Once you are ready to program the device click the “Program Hardware” button. If you want to reset everything, just like the computer software, you can hit the “Clear Presets”. Once you’ve done everything you want with the messages, you can disconnect the device and use it.

Turn on the device by hitting the power button on the back of the device. From here you can hit any of the four buttons you programmed to output their respective messages. You can also edit the brightness with buttons on the back. Once the device is low on battery the red LED on the back will turn on indicating you should charge the device soon.

Troubleshooting	
Problem	Solutions
The device won’t turn on.	• Make sure the device is charged. You can do so by plugging in the USB charger and looking at the LED light. • Make sure there are no broken parts on the device.
The software won’t recognize the device.	• If using a computer, make sure the wire is securely inserted in the device and the computer. • Check to see if the USB wires are faulty or not. • If using Bluetooth make sure both the device and Android phone have their Bluetooth turned on.
The device is not outputting the correct messages.	• Make sure you are hitting the right button for the respective message. • Double check the messages were loaded onto the device.

7. Administrative

Execution of this project requires certain administrative oversight to keep the team focused with realistic goals and proper funding. This section will illustrate how the device will be paid for, as well as discussing how the device can be marketed to consumers. It will also provide a timetable for when tasks should be completed to keep progress on the device on track.

7.1 Budget

The funding for the project is, generally, derived from the pockets of the team members. The sponsor has promised a stipend upon project completion, but the stipend will not be near covering the cost of the final product as well as research and development of the product. This section will cover the investments that are expected to be necessary to fund both the initial prototype and the final build of the product, as well as pricing should the product enter mass production.

7.1.1 Prototype Budget

The prototype will require a decently large number of parts, some of which will not be present in the final build. These include the development board, breadboards, through-hole LEDs, etc. A few parts were purchased in advance that were found to be unneeded, but they will not be included here as this is intended to show what the actual cost of the prototype would be. Below is a table that illustrates the parts needed, their price, their quantity, and the total cost per part.

Part	Cost	Quantity	Total
Arduino MEGA Development Board	$38.50	1	$38.50
Breadboards	$10.50	2	$21.00
Red LEDs	$0.12	128	$15.36
Connector Wires	$7.50	1	$7.50
USB A-to-B Cable	$6.00	1	$6.00
Shift Registers	$0.45	3	$1.35
Transistors	$0.21	8	$1.68
Shipping	$20.00	1	$20.00
Total			$109.92

The total cost of the prototype build, without any mistakes, is estimated to be around $109.92. However due to the high likelihood that mistakes will be made, along with the necessity to order new parts and the possibility that the wrong parts may be ordered, a dollar value of closer to $200.00 is a safe estimate of the prototype cost. With group 21 being a four-person team, it is therefore proposed that each member contribute $50.00 to the construction of the prototype to spread the effects of the costs throughout the team.

7.1.2 Final Product Budget

The final product will be using significantly more parts than the prototype. These parts included the printed circuit board, the incredibly large number of LEDs, many more shift registers, buttons, a Bluetooth LE chip, a switch, the housing, the handle, etc. Furthermore, these parts are purchased through relatively expensive means by going through domestic outlets. It would be much more cost effective to purchase these parts directly from foreign markets, where the costs of production are cheaper, namely Chinese markets. This only works for mass production, which will be discussed later.

7.2 Bill of Materials

A bill of materials (commonly known as BOM) is a list of raw materials, sub-assemblies, intermediate assemblies, sub-components, parts, and the quantities of each needed to manufacture an end product. Our bill of materials will mainly include a varying array of different electronic parts ranging from transistors to light-emitting-diodes to shift registers. A bill of materials is good for communicating between members of a team that are working together to design and build a device, just like our team is doing for our handheld electronic message display.

Below is a table that illustrates the parts needed, their price, their quantity, and the total cost per part for the final product budget.

Part	Cost	Quantity	Total
Atmel MEGA 2560 MCU Chip	$11.89	1	$11.89
Printed Circuit Board	$34.99	1	$34.99
1206 Red LEDs	$0.0033	2048	$6.83
Bluetooth LE Chip	$2.48	1	$2.48
Shift Registers	$0.45	12	$5.40
Transistors	$0.21	32	$6.72
Buttons	$0.42	7	$2.94
Housing	$27.22	1	$27.22
Handle	$4.22	1	$4.22
USB Port	$2.52	1	$2.52
Switch	$0.84	1	$0.84
Battery	$32.99	1	$32.99
Shipping	$45.00	1	$45.00
Total			$184.04

The total cost of the final build is estimated to be around $184.04. This puts the cost at around twice the estimated total cost of the prototype. The printed circuit board also costs a large amount, and the shipping from the many individual sources of the part runs up to a large amount as well. The final large cost is the battery, as we will be using a Lithium Polymer flatpack battery of at least 7 volts, which is relatively expensive. Altogether, this puts our expected cost to be around at least the $250.00 mark. This will also be split among the team members, putting each team member's investment at around $125.00 for both the initial prototype and the final product combined.

7.2.1 Mass Production

The interesting fact about mass production is that it severely reduces cost. Not only is this true for just about every part, it also allows for a streamlined shipping process due to ordering most parts from a single vendor. For example, take the difference in cost between the prototype LEDs and the final product LEDs. The final product LEDs are 36 times cheaper than the prototype LEDs. Most of the other parts are significantly cheap when order parts by thousands as well. None are quite as impressive as 36 times cheaper, but they can reach up to as much as 8 times cheaper than if they bought in smaller quantities.

7.3 Marketing

Although there are similar devices out there, this product truly has a market that would be willing to buy it as a product. This device has many real-world applications and would be successful with the right business team to support it.

The main application for this device is a fun tool to use to communicate to friends. You can write any message wanted up to 150 characters in length. With four programmable buttons, you can use the same four messages over and over or you could put new messages on all the time.

Another application that this can be used for is to communicate between drivers on the road. With the four different messages that can be stored, every driver can say what they're really thinking to other drivers you can't normally communicate to. For example, many people get angry with the way people drive and don't use a blinker. If a car cuts you off you can use your LED device to display "USE YOUR BLINKER NEXT TIME!"

At almost every sporting event there's always supporters that bring signs to support their team. With this device, it's like you have four signs all in one with the four messages you can program to this device. Throughout the game you can keep changing what your device says. Imagine being at the next football game and being able to display "CHARGE ON!" to the whole stadium and national tv.

With this device, you can also remove the handle and have a permanent LED display mounted in your room. You can have positive messages scrolling to help motivate you get through your homework assignments or even funny quotes to make you laugh every time you see it. You'll feel like you're on wall street with the scrolling LED display in you room.

This device also has a very diverse market. Anyone from either gender or any age would love to have a device like this. Someone younger may use this as a toy or a game while an older person would use this to display messages potentially while driving. With the device being very easy and self-explanatory to use, there's not learning curve to get to use it. It's a common thought that a lot of people aren't good with technology, however, this device could easily be used by someone that's never used a computer before.

This product could also be sold in many different types of stores. Amazon and other online stores are always on the cutting edge to selling innovative technology. A product like this would thrive on an online store. Also, local stores such as Walmart would have a lot of success with selling this product. This is the type of product that catches your eye when you walk past it in the store. With how inexpensive it is, it'll be impossible for anyone to pass it up once they look at it.

This product will also be easy to manufacture and remake once the schematics and basics software is created. The hardest part about putting this project together is making sure everything is engineered correctly. Drawing the circuit, making sure the correct voltages and current reach each part, creating the software, then connecting the software to the hardware are all hard to do. However once all that is done once, nothing new must be done. The software can be copied repeatedly and redistributed easily. Once the schematic is drawn and proven to work, it easy to print out the PCB over and over. The LEDs being connected to the circuit also would be easy to recreate after the first time. Considering all this, this would be a good product to manufacture and try to sell on the marketplace.

7.4 Project Contributors and Assistance

This project is both guided and sponsored by Michael F. Young. Mr. Young is a professor at UCF and founder of Young Engineering Services, LLC. Being an experienced professional in the field of electrical engineering, Young provides consultation for the group as necessary, as well as consistent guidance concerning project progress and quality. Also, Young has obtained competitive products for comparison with design and functionality. As for funding, Young will provide a stipend for a completed, working final product.

For professional fabrication of the PCB, the manufacturing company Seeed will be utilized when schematics and Gerber files are completed. Seeed offers incredible flexibility with service options and accommodations, making for a valuable resource in terms of constructing the exact PCB that the project needs. In addition to PCB fabrication Seeed also offers crafting of solder paste templates for the sake of quick and manageable soldering of small SMD components to the PCB. The template is used to expose only the areas of the PCB where solder paste needs to be applied, and leaves all other parts of the PCB clean. This will make assembly of the PCB and components extremely efficient and precise, especially considering the number of LEDs and components that will need to be soldered.

Another company willing to provide aid in the manufacturing process is Quality Manufacturing Services (QMS), a PCB assembly facility that mounts SMD parts that are normally too small or complex to solder by hand. The facility was observed through a tour to understand the system for the “pick-and-place” operation. The representative from QMS made the message clear that any parts too difficult to place ourselves should be left for the facility to mount on the board. This, however, does not account for all parts, which was advised the group take responsibility for. The aid from this company is immensely appreciated, as assembly of possibly multiple revisions along with many components coming in many shapes and sizes is a cumbersome task for any group to accomplish.

7.5 Development Methodology

When code is being written by a team there must be a plan for the software development the team will use to work together. There are many different types of software development types and strategies to use. In fact, there are so many different types that there are classes dedicated to teaching the different types. UCF has a class called Process of Object Oriented Programming which is required for Computer Engineers to take. This class taught the different types of software development such as Agile or Waterfall and different strategies for software engineering.

The type of software development our group decided to go with is Agile software development. This decision mostly effects the computer engineers in the group so the two of them were the ones who came to that decision. Agile development has a lot of advantages compared to other software development styles and is one of the most popular types currently. Along with this agile development is one of the newer types of software development.

Agile development can be referenced back to 1957 however in 2001, *Manifesto for Agile Software Development* was published by seventeen software developers. From then there has been many different standards, guides and books written to further Agile development.

The purpose of Agile Software Development is a way of developing software that has requirements and solutions that are quickly changing. This is done by creating cross-functional teams that are collaborative and self-organizing. The Agile Alliance does a good job at describing the advantages of Agile Software Development by saying “It advocates adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change” (Agile Alliance, 2015). The main principles of Agile Software Development are that everyone is self-organizing and self-motivated, they prefer working software over loads of documentation, the customer must be active with the coloration and that its quick on responding to change.

In the *Manifesto for Agile Software Development* there are twelve principles of Agile Software Development. These twelve principles show what makes agile development different than other types of development styles. The main idea for the twelve principles is that the code is very flexible and that you frequently work with the customer to make sure the software is what they want. These principles work together well because as you talk with the customer more, the software itself will likely need to be changed often. This works out because the principles also call for flexible work within the code.

One of the main points of agile development is breaking up the development of the software in to small portions. This way you don’t have to worry as much of the up-front planning and get to work on the specific portions. Sprints indicate one these small portions that typically last from 1 to 4 weeks. In the sprint, each developer plans on finishing their part of the small portion. At the end of each sprint or portion, the should be a product that is able to be displayed to stakeholders. Breaking the development into smaller portions makes it easier for the product to be adapted or change quickly.

Another big part of agile development is that everything is efficient and that there is face-to-face communication. In agile development, there is always at least one person on the team that is the main contact for the stakeholders. This person is referred to as the Product Owner. This persona is responsible to staying in contact with the developers and available to answer any questions they may have about the product. At the end of each sprint, the Product owner gets in contact with the other stakeholders to make sure the product is still in line with everything they originally intended and that is worth their time.

Another software development technique is the waterfall model. The waterfall method is older than Agile Software Development. The main idea of the Waterfall method is that it is a linear sequential design. The waterfall model being linear makes it a lot less flexible than the other software development styles.

The waterfall model was first introduced in 1970. When the waterfall model was first introduced, it was introduced as an example of a flawed, non-working model rather than something that should be used. However, this software development model is still in use and has six stages in a linear order which are original design, more in-depth design, coding and unit testing, integrating and then its testing. In this model, a person or team should not move to the next stage until it is confirmed that the current stage is done and done correctly. However later in the life of this model, it’s been accepted to move back a stage or all the way to the beginning if there is a flaw or something went wrong elsewhere inside the project.

Bugs or problems that are caught in the code earlier in the process will cost a lot less money than any bugs or problems caught later in the process. The waterfall model really helps with this by helping catch problems earlier on in the process. With the waterfall method, a considerable amount of time is spent just on planning and getting the structure for the process in place. With this everything about the project will be very strict. Especially with larger projects, every single step along the way to creating the project is regulated. Another big thing that makes the waterfall method a desirable one is the emphasis on documentation and source code. Documentation and source code are very important especially for big projects. Having good documentation and source code makes it so other people can easily pick up where someone left off. Without any types of documentation or source code, if someone else where to try to pick up the project or code, that person would spend more time trying to understand the current code and processes than they would actually coding or working on the project. Also with the waterfall method it is very easy to understand. Everything is linear and there’s no cross-referencing relationships throughout the process. Everything is straight forward and the people on the project don’t have to spend much time thinking about the process itself.

So, although it may seem like there’s a lot of positives to the waterfall method, there’s also a lot of negative things and criticism that it receives. Although everything is very upfront and straightforward from a developer’s standpoint, it can sometimes be hard to keep transparency with the clients. This becomes a problem because typically the client wants to be able to see what going on with the project and the progress that’s been made. In our situation, its Dr. Young who we need to consistently keep updated. A lot of time, people working on the project cannot foresee the problems that will arise while working on the project. The waterfall method is not very good at solving unforeseen problems or changing up the current process of the project.

After discussing with the team, we have decided that Agile development would be the best process for our team to follow. There’s a lot of factors that helped us make this decision. A big one is that multiple team members already have experience with Agile development while no one in the group has had previous experience with the waterfall method. Another big factor is that the waterfall method seemed to be more beneficial on really big projects while Agile development seemed to be more beneficial for smaller projects. On our team, we have the four team members, our sponsor Dr. Young and the two professors for the class. This is a pretty small group for software development so in this sense Agile development would be more beneficial. Along with these previous things, Agile development is a lot more flexible. This is very beneficial to us because most of our team members are not too familiar on a lot of the things were doing for this project. For example, no one has much experience with C# or Unity. Since there’s a lot of things were not completely confident with, we assumed that we will run into unforeseen problems. Agile development is much better when it comes to running into unforeseen problems. The last factor that we looked into while decided which software development style to use was the transparency. Agile is much more transparent with the client. Dr. Young is our client but also someone who guides as a lot throughout the process of creating this device. Being transparent and being in close communication with him is very important and helped us choose the Agile Software Development.

7.6 Milestones

Similar to a task list, the milestones provide a timeline of objectives to be completed, including the date and a brief description of the task necessary. By referring to the list and dividing the work among team members, progress can continue in a trackable, efficient manner. Below is the completed list of goals with desired completion dates. Detail is left out of the task description to provide only a quick, simple view of the requirements, to verify the work completed, and work that needs to be done. Documentation is separately listed because this documentation does not physically provide changes or design to the overall product. Only project tasks are included, so the direct progress of the project itself can be visualized. Also note, trivial details in terms of tasks are also left out. The reason for this is that the purpose of ‘milestones’ is to show the more prominent objectives that are desired to be accomplished, not the numerous actions that are needed to reach that. In this case, the goals are set, and the actions are sequentially, and separately, determined that help reach that goal. The dates set for each goal also provide insight as the progress overview can be understood as the approximated completion time is kept in accordance with the final date of submission.

As seen in the Milestones Chart, prototyping is an integral part of the initial design. Once the design is finalized, after many revisions, continuous testing has to be accomplished to verify all of the needed functionality, and making any changes that are needed. This includes hardware and software, especially knowing that any changes in either sense usually affects the other, requiring even more changes. Dates are estimated based on the necessary action needed to reach the goal, as determined by the member(s) responsible for achieving that goal, and agreed upon by the group. As prototyping continues, testing is intermittently conducted to analyze the functionality every step of the process. Acquiring this information is vital to keep the group from back-tracking the process and, thereby, delaying the entire timeline. Once designs are completed for the prototype and tested, the focus immediately shifts to the final project layout, expanding the prototype design to what the final circuit needs to look like. All software at this point, towards the end of the electronics design process, should be established. Such software includes a scaled-down variation of the final project code that operates that LEDs, as well as a working GUI, with functionality only for basic input/output and device recognition necessary. After the completion of all, and the prototype is complete and successful, the code will be further built up to support the final product, with all features present for both the GUI and the code that operates the matrix. The electronics design is also expected to be complete as soon as possible for PCB manufacturing, since PCB delivery is a very time-consuming operation. All of this should be ready for immediate assembly and implementation in the spring term.

In the spring, the final project should be constructed based on full confidence from prototyping and design from the fall term. The first PCB should be immediately ordered at the start of the term, in anticipation that the return time of the PCB will take as long as 2-3 weeks. During this time, software should be completed with iterations of final testing, debugging, commenting, and optimizing, if not already done. Also during that time, the device encasing should be assembled to take care of the remaining hardware that is required for the final product. When the PCB finally arrives, Rev. A, all of the components must promptly be soldered onto the designated locations. PCB issues will likely be faced, or at least expected, which will be tested for in the most efficient manner possible to make the best use of manufacturing time. Assuming any problems are found with the PCB, a newly corrected PCB, Rev. B, is quickly ordered. Any additional tasks that are incomplete should receive full attention at this time, while the Rev. B PCB is being delivered. Any issues in hardware and software are solved again, which should finalize debugging and testing for the product. The encasing should be finalized by the end of this period. Then when Rev. B arrives, the components are soldered, and the testing process is conducted again. Depending on the time remaining and conditions at hand, issues are solved in a timely manner to make the device fully operational. All features of the software and hardware should be fully implemented. Both wireless and USB connected communication should be successful and reliable. Final operating tests are conducted and any issues found are resolved. Once full confidence from the entire team is agreed upon, the project is deemed complete and submitted for review. Given the expected completion time, flexibility for any last-minute problem solving is accounted for in the timeline. Plenty of room for errors and delays should be accurately accounted for in the chart, as long as goals are progressively completed on schedule. Following this chart, however, the project should progress smoothly and with plenty of space for adjustment.

7.6.1 Milestones Chart:

Fall Semester:	9/3/17	Project selected
	9/26/17	Prototype components selected and ordered
	10/11/17	Prototype circuit fully designed
	10/12/17	All parts ordered
	10/20/17	Prototype circuit assembled and tested
	10/22/17	Prototype code designed for selected MCU
	10/27/17	Software GUI basis established
	11/22/17	Code tested and debugged with prototype circuit
	11/25/17	Final electrical schematics complete
	11/28/17	Prototype finalized and presented for review
Spring Semester:	1/12/18	PCB Rev. A designed for manufacturing and ordered
	11/15/18	Device encasing and handle designed
	1/19/18	Initial encasing constructed
	1/23/18	Prototype code expanded to final matrix and components
	1/26/18	PCB Rev. A tested and programmed
	1/31/18	PCB re-designed and reordered as necessary (Rev. B)
	2/7/18	Device encasing and handle constructed
	2/12/18	PCB Rev. B tested and programmed

	2/16/18	LED matrix placed on PCB
	2/21/18	PCB components, battery, and buttons assembled and tested
	2/28/18	Device assembled within the encasing, adjusted as necessary
	3/9/18	GUI completed and tested
	3/21/17	Interface and connectivity issues solved
	3/28/17	Complete device tested and analyzed
	4/20/18	Final revisions completed
	4/26/18	Project completion

7.6.2 Document Deliverables:

8/25/17	Idea page submission
9/15/17	Specification Diagram
9/22/17	Initial specification document submission
10/2/17	Mock user manual submitted
11/3/17	Remaining document rough draft
11/17/17	Full assembled document rough draft
12/4/17	Final complete document

8. Conclusion

This report shows the design and considerations that went into the creation of a handheld LED sign. The primary concerns of its production were that it be consumer friendly, lightweight, and easily accessible. To that end, more specific design criteria were created such as the max limitation of 2 lbs., the control scheme, and the battery requirements. Other requirements came from the general necessities of making electronic devices that exist in industry standards and institutional constraints.

After the main list of requirements was compiled, the main design phase began. The first and most important aspect was the LED array itself, and it’s required voltages and control systems. The control systems went through many iterations, from the use of decoders and an up-counter with a set of inverters to the final design of using shift registers as a ring counter to control the rows and simply connected shift registers to control the columns. Transistors were eventually added to the rows to create a third floating point outside of just high and low voltage. This allowed for only a single row to have a complete circuit at one time.

Once the LED matrix design was complete, with values selected for the resistance and LEDs selected of the proper type, the support circuits began being designed. The battery needed to be rechargeable, and the only Lithium batteries of a reasonable price were 3.7V. That led to the inclusion of a voltage amplifier circuit. Other support circuits include the button control circuits, USB circuits, and Bluetooth circuit.

With all the components designed and purchased, and enough research put into the product to make sure that we are not breaking and governmental or environmental constraints, we are confident that the product will function as intended. We hope that it provides a viable alternative to the non-consumer focused products that are currently available on the market. Should the design be a success, we would like to expand its production using slightly different production methods than what went into this design, such as using a die-cast aluminum housing instead of a 3D printed housing.

9. Bibliography

[1. “Electronic Component Distributor Sales Terms and Conditions.” Mouser Electronics, www.mouser.com/saleterms/.

[2. “User Code of Conduct | Bluetooth Technology Website.” Bluetooth.com, www.bluetooth.com/user-code-of-conduct.

[3. Liang, Oscar. “BJT vs MOSFET.” Share Knowledge and Ideas, 12 Oct. 2013, oscarliang.com/bjt-vs-mosfet/.

[4. Poole, Ian. “Bluetooth Technology Tutorial.” Radio-Electronics.com | Resources and analysis for electronicsengineers, www.radio-electronics.com/info/wireless/bluetooth.

[5. “LED Lights - How it Works - History.” Edison Tech Center, Copyright, 2013, www.edisontechcenter.org/LED.html.

[6. “What LED Color is the Brightest? What Every Installer Should Know.” Feniex Blog, www.feniex.com/blog/2016/02/22/83/.

[7. “World's Largest Selection of Electronic Components Available for Immediate Shipment.” DigiKey Electronics - Electronic Components Distributor, www.digikey.com/.

[8. “Battery Types: Rechargeable and Non-Rechargeable.” Battery Solutions, www.batterysolutions.com.

[9. “32x16 and 32x32 RGB LED Matrix.” How the Matrix Works | 32x16 and 32x32 RGB LED Matrix | Adafruit Learning System, learn.adafruit.com/32x16-32x32-rgb-led-matrix/how-the-matrix-works.

[10] “How to Dim an LED Without Compromising Light Quality.” Digikey Electronics, Digi-Key's North American Editors, 11 Oct. 2016, www.digikey.com.

[11] Kumar, Barun. “What Is the Difference between a Clear LED and a Diffused LED?” Quora: A Place to Share Knowledge and Better Understand the World, 6 Sept. 2015, www.quora.com/What-is-the-difference-between-a-clear-LED-and-a-diffused-LED.

10. Appendix

Lumex Image Use Authorized:

Peter Guarner <pguarner123@gmail.com> 10:24 AM (2 hours ago)

to Jack

Jack,

Of course, thank you so much.

Peter

On Fri, Dec 1, 2017 at 3:54 PM Gaon, Jack <jgaon@itwecs.com> wrote:
Hi Peter,

Approved on the conditions that image is used for school project purposes only and the source is noted as ITW Lumex.

Kind regards,
Jack Gaon, E.E.
Eastern Regional Sales Manager
***ITW* Lumex- Electronic Component Solutions**
ITW Formex | ITW Linx | ITW Lumex
Mobile: (516) 319-0404
Email: jgaon@itwecs.com

425 N. Gary Ave. | Carol Stream, IL 60188 | www.itwecs.com

From: Peter Guarner [mailto:pguarner123@gmail.com]
Sent: Wednesday, November 29, 2017 7:12 PM
To: Gaon, Jack <jgaon@itwecs.com>
Subject: Re: LMX / U of C Florida / Contact Us Submission

Hi Jack,

So I am working on a project and needed this particular image for research and referencing purposes to cite in our project paper:

Which I found on the Lumex website. I wanted to see if I could receive permission to use this image, while citing and providing due credit.

Thanks,

Peter
On Wed, Nov 29, 2017 at 10:47 AM, Gaon, Jack <jgaon@itwecs.com> wrote:
Hi Peter,

How can I help you?

Regards,
Jack Gaon, E.E.
Eastern Regional Sales Manager
***ITW* Lumex- Electronic Component Solutions**
ITW Formex | ITW Linx | ITW Lumex
Mobile: (516) 319-0404
Email: jgaon@itwecs.com

425 N. Gary Ave. | Carol Stream, IL 60188 | www.itwecs.com

From: Oliveros, Jeffrey
Sent: Thursday, November 16, 2017 9:10 AM
To: Gaon, Jack <jgaon@itwecs.com>
Subject: FW: Contact Us Submission

From: Lumex Inc. [mailto:lmxsales@lumex.com]
Sent: Wednesday, November 15, 2017 8:26 PM
To: Lumex - For Leads <lmxsales@lumex.com>
Subject: Contact Us Submission

Comments Hi, my name is Peter Guarner and I am an engineering student at the University of Central Florida. I would like to talk with a someone about using informational material provided on the Lumex website for the purpose of conducting a project. Please reach me at the email provided. Thank you.
Email pguarner123@gmail.com

FirstName Peter
LastName Guarner
Company N/A
Title N/A
CountryId United States
Department Engineering
Industry Electronic Signage
StreetAddress 12215 Kings Knight Way
ZipPostalCode 33024
City Orlando
StateProvinceId Florida
Newsletter False
Phone 9546681255
Website
LeadSource Lumex.com: Contact Us Form