Wilderness Information Link Device

WILD

University of Central Florida

Department of Electrical and Computer Engineering

Dr. Samuel Richie & Dr. Lei Wei

Senior Design 1 - Spring 2019

Group 10

Final Documentation

Brandyn Brinston

bbrinston@knights.ucf.edu   Computer Engineering

Caleb Phillips

calebphillips@knights.ucf.edu   Computer Engineering

Michael Sedlack

m.sedlack@knights.ucf.edu   Electrical Engineering

Justin Tuggle

jtuggle@knights.ucf.edu   Electrical Engineering

Table of Contents

List of Figures

List of Tables

1. Executive Summary

Our project tasks us with the designing and testing of a communication device capable of transmitting a texted ASCII message between another of the same device. We intend to use long-range wireless transceivers, with capability of communication distances up to a few miles outside, and a few hundred feet inside. The device should be relatively small whilst containing an LCD for the display screen, a basic keyboard consisting of all alphabetical letters including spacing, and status LCD's relating the current whereabouts of the message being sent. For the device, we have considered the use of a li-ion battery for operations, having an overall lifespan of approximately four hours, which can also be recharged fully in around two hours via a cable. Other thoughts and decisions have also been made regarding other aspects of the device. This includes technical specs of the device, such as the power ratings, frequency spectrum range, weight, size, appearance, etc.

Fortunately, we are in a good position for the time being. The various modules that will be used for our device are being researched and considered for implementation. As of now, we have currently decided on a few of these said modules. The microcontroller that we have interest in using for our project will be the MSP432P401R. The decision to choose this MCU was ultimately made based on a preference of embedded programming for the MSP family. For the I/O section, the keyboard is a tad trickier and has required more attention. This was due mainly to concerns toward its interfacing and sizing. After research, it appears as though the optimal choice was to find a small enough keyboard that still offered an ergonomically acceptable feel for the user, that is low-voltage and draws a relatively small amount of current. Further research helped us identify a good candidate for this module, the BlackBerry Q10's keypad. The only issue regarding the use of the Q10 is that interfacing it has proven to be a rather difficult process. The LCD display will most likely consist of a 20-char x 2 lines dimension with a length of approximately 2.93 inches. This seems like the most optimal choice as the size of this screen will be only slightly larger than the keyboard, and so the casing of the device will most likely leave just the right amount of spacing on all sides. Although other modules are still being examined, we will also be implementing solar panels as the means to recharging the battery that our device will operate off. As far as the programming aspect is concerned, we will most likely be incorporating embedded C for the I/O of the system as well as mesh networking for the stability and consistency of the transceiver's communication(s).

For the device, we have considered a multitude of different reasons for its practicality in the real-world. It is understood that this creation serves the purpose of acting as a simplistic communication device that can be used for emergency situations. It would also act as a device for messaging capability in areas where cellular communication or networks where internet access is not readily available.

Fortunately, the main priority of this project is to create functional transceivers capable of transmission of text-based messages, and incorporation of these devices into real-world situations would be acknowledged after functionality is properly working.

2. Project Description

For our project, we expressed an interest in making a device capable of texting communication. This device would consist of a microcontroller that would utilize a transceiver for means of communication. The input to this would be a keyboard most likely, although there is also a potential for use of cursors as our main form of entry. An output screen such as OLED or LCD could be used to portray the inputs associated by the user. The casing for this device would be designed to withstand environmental factors, such as temperature, rough terrain, etc.

Our microcontroller will handle the task of handling most of the processes involving functionality. This chip will be programmed such that all the hardware modules will communicate efficiently and accurately with their relevant components. For instance, the MCU will be acknowledging an input made by the user from a keyboard, and then make an output to the display screen.

The transceiver module has capabilities that allow it to be the optimal choice of data communication between multiple devices. Essentially, messages can be sent and received from any transceiver that is in range of one another. This component will directly talk to the microcontroller, in a bi-directional path. Incorporating mesh-networking will help disperse communications more effectively with transceivers, which may be used once basic functionality of the transceivers has been established.

Inputs from the user may be in the form of tangible buttons. A miniature keyboard will be the main form of input, with a width of 2.9 inches. Once this keyboard has been interfaced properly, connected and soldered to the MCU, character entry will prove to be successful. Any key pressed will result in its equivalent ASCII code being interpreted by the program written for the MCU, which should then be outputted to the display screen.

The output to the display screen will be in the form of the ASCII characters from the key inputs made by the user. The screen will be a liquid crystal display and will consist of a width size of approximately 2.9 inches. The characters per line will be 16x2 respectfully, and the decision to use 16 characters comes from the caution of using hardware lines. For sixteen characters, we use four data lines and four control lines, which results in eight total hardware lines. A total of 32 characters may be displayed using this module.

Status indicators will be implemented into casing of the device as well. The illumination of the light emitting diodes will help clarify about the communication

regarding a message being sent. This will include notifying a user about the whereabouts of the data transfer, such as when it is idle, in the process of transferring, being sent or received, etc.

2.1 Motivation

In today's technologically-driven world we are more connected than ever. Communication has long been at the center of many emerging technologies. As we continue to advance our ability to communicate with innovations that bring connectivity to the farthest reaches of our big blue planet, there are still some areas that are, quite frankly, hard to reach.

With modern cellular and satellite communication we are able to establish wireless connections on opposite sides of the Earth. These solutions, however, are not always ideal. With cellular communication we are able to send vast amounts of data at very high rates, with very low latency. However, cellular networks are only as good as the quality and distribution of their towers. There still exist significant swaths of the United States that do not have cellular service coverage; rural areas, parks, and nature reserves, to name a few. As such, it is desirable to have a method of electronic communication that does not rely on cellular services while in these locations. In these areas, one of the technologies that has seen fairly extensive use thus far is satellite communication. While often slower than cellular networks, satellite provides true coverage of almost the entire surface of the Earth. Unfortunately, this comes at a high price, and often requires much larger equipment; in addition, it is susceptible to atmospheric interference.

We are seeking to create a device that will facilitate communication for those who are in these very areas or situations that do not allow for easy and accessible cellular communication. We are seeking to create a reliable way for individuals to remain connected without any need for outside networks such as cellular towers or satellite connections. The device we plan to create will also be capable of sustaining operability for long periods of time by collecting its own power via solar panels. Another goal of our project is that it will be able to maintain a connection over a long range.

While defining our project goals we considered several different scenarios in which such a device would be useful. One major contributor in this area was for outdoorsmen in rural areas. Whether it be a long section of the Appalachian Trail, the untamed wilderness of Alaska, or even the vast expanse of the Sahara Dessert, there are oftentimes when some form of communication is either vital or at a minimal very useful. Maintaining a line of communication also provides an inherent security. Without communication a minor injury in the wilderness could become a major problem, for example.

A couple of methods of communication to this end were considered, and the conclusion was reached that text rather than voice was going to be the most reasonable communication method – the reason for which is bandwidth. The reason CB and HAM radio suffer so badly in requiring large, non-portable equipment is due to their high power and bandwidth capabilities; if we remove the ability to transmit audio – a relatively bandwidth-intensive feat – we can have a greater range using far less power, all while using equipment that is significantly more portable.

As such, the purpose of this project is to develop a wireless communications device capable of transmitting point-to-point (P2P) text messages of sixteen or more characters across a relatively long distance. With this aim in mind, there are a few key points we would like to address in order to express our expectations of the design and functionality of the device.

2.2 Goals and Objectives (to be expanded)

It is the aim of this project to create a device capable of high performance (as determined by the criteria set out in the test plan to be developed later) and of a high build quality. It is the hope of the team that we will each further our technical and interpersonal skills over the course of this project, and that we will have a better understanding of and appreciation for the formal design process. We will better our skills in the creation and consumption of documentation. It is the sincere desire of the team that we will develop professionally over the course of this project, leaving us prepared to practice engineering in a real world environment.

2.3 Requirements Specifications

Our team has considered a vast number of issues that may be potentially concerning about the specific objectives and approaches that we intend to achieve. In regard to the electrical design, we have determined some of these concerns. For instance, the range that the transceivers could communicate between is one of our priorities, as we have a set specification that needs to be met. Fortunately, a simple remedy to this concern has been met, as we determined an adequate chipset that far exceeded this constraint. Moving onwards, there are other electrical constraints that we had listed previously, such as a limited amount of wattage rated per module. This is necessary so that when we build our PSU, we will know what wattage is necessary to sufficiently power our transceiver device(s). Still, other constraints are present, such as interfacing a small keyboard to the microcontroller, implementing solar panels capable of charging an upwards of a watt, and further.

Other constraints include the software aspects. Embedded programming is an imperative necessity to the functionality of our transceivers. The constraints that are related to this are numerous. A character limitation of twenty-per line is most likely our limitation as the OLED module contains this exact specification. We will also need to be able to program the microcontroller such that it can operate based on low-power usage. The messages that we plan to type out will also need to be in the ASCII standard. This means we must be familiar with its table involving the octal and hexadecimal codes that relate to each character. Considering we are using the MSP family by Texas Instruments, the embedded programming will be specifically the language C, which is another constraint imposed by the project. Still, other constraints are also present. We will need the programmed transceivers to efficiently communicate with one another, and at adequate distances to make them a relevant design. This could present a situation in which mesh networking must be implemented, although we intend to keep the system simplified for the moment unless otherwise required. The code written will need to adhere to the space limitations of the RAM which for our considered MCUs is 64 KB of memory and higher.

From the list of constraints seen in the above sections, it is understood by our team that proper research and design must directly correlate with these limitations. Fortunately, many of these restrictions have been considered. This enables us to make an approach towards the device that becomes much narrower and more specified, leaving room for error and disregard to these constraints minimized. Although it is reasonable to assume that there are many "hidden" constraints still unseen, we feel that there is enough flexibility currently. The research and design stages of the project are now currently underway and any problematic or necessary limitations to the hardware or software components will be updated within the document.

2.3.1 Engineering Requirements Specifications

2.3.1.1 Hardware

1.0	Battery operation of device capable of lasting for approximately 4 hours
1.1	Time required to recharge battery shall be approximately 2 hours
1.2	Transceiver frequency spectrum should be in the range of 900-930 MHz
1.3	Power rating of the modules of use should be within 0.5-3 Watts
1.4	Solar panels utilized may generate an upwards of roughly a watt
1.5	Keyboard should be fully functional & capable of inputting the desired characters
1.6	Display screen will output the ASCII characters as input from the key pressed
1.7	Status LEDs respond in accordance to the message being sent/received

Table 1: Hardware Requirements

2.3.1.2 Software

2.0	Software will conform to spacing limitations of code posed by MCU
2.1	Embedded software shall be able to communicate efficiently to all relevant hardware
2.2	Embedded software should be designed to support low-power use
2.3	High-power performance solutions shall be emphasized
2.4	Transmitted/received messages shall not exceed ~ 16 character limit
2.5	Communicated messages will consist of the ASCII encoding standard

Table 2: Software Requirements

2.3.1.3 Mechanical

3.0	The weight of each produced device shall not exceed ~ 10 lb
3.1	Each device shall be easily transportable
3.2	Device shall provide adequate ergonomics
3.3	Entirety of the hardware should withstand a drop/impact of ~ 3 ft
3.4	The cost of any singular module shall not exceed a cost of $250

Table 3: Mechanical Requirements

2.4 House of Quality

The House of Quality diagram seen in Figure 2: House of Quality, lists and compares the relationships between the engineering requirements and the expected customer requirements for the Wilderness Information Link Device. Once relationships between the different requirements are established, the most important engineering requirements are calculated based on the weights of the customer requirements. Each of the five most important customer requirements are listed on the left portion of the diagram and ranked based on importance, with the most important requirements—starting with five being the most important—being placed in a higher position than the requirements with a lower importance. A weighted percentage was then assigned to each customer requirement based on the importance of each requirement to the customer. For instance, the most important customer requirement, which explains that the Wilderness Information Link Device must be able to run on battery power for at least four hours, was given a relative weight of 33%. Alternatively, the customer requirement which states that the cost of the device must not exceed $250 was given an importance rating to the customer of one, which resulted in a relative weight of 7%. Listed in the vertical direction in opposition to the customer requirements, are the engineering the requirements. These ten most important engineering requirements each have a direction of improvement assigned to each in the form of an upwards bolded triangle, which signifies that this requirement shall be maximized, a down facing bolded triangle, which signifies that this requirement shall be minimized, or an outline of a square, which signifies that this requirement shall be reached, but neither maximized nor minimized. For instance, the requirement which states that the transceiver frequency shall fall within the range of 900-930 MHz is designated with an upwards bolded arrow. This implies that although the listed range is 900-930 MHz, the transceiver frequency shall be maximized such that it is as high as reasonably able, within reason of what is both capable and allowed by established standards and within the constraints of allocated funding and what is considered possible for the Wilderness Information Link Device. Alternatively, the engineering requirement which states that the keyboard on the device must be functioning has been given a target designator, which shows that fulfilling the terms of the requirement involves producing a totally functioning keyboard—maximizing or minimizing this requirement would not be reasonable and scenarios where the keyboard is not functioning as desired would not satisfy the requirement.

After assigning directions of improvement to each of the engineering requirements, relationships of correlation were assigned to each of the engineering requirements. In the triangular matrix above the engineering requirements section, symbols denoting either a positive, negative, strong positive, strong negative, or no correlation were assigned to each of the possible relationships between the ten engineering requirements. A positive relationship is shown by assigning a cross symbol. A negative relationship is shown by a small dash. Similarly, the symbols for a strong positive correlation and a strong negative correlation were shown by double crosses and double dashes, respectively. A relationship where there is no correlation is shown by a blank space. To illustrate, the engineering requirements

that state that the device shall be able to withstand a drop from three feet and that the solar panel shall be able to generate one watt were given a negative correlation designator. This is due to the relationship where the larger amount of power generated, the larger and more expensive the solar panel would likely need to be. As the size of the solar panel is increased, the likelihood that the device as a whole would be able to continue to withstand a three-foot drop would decrease, signifying that there is an inverse relationship between the two requirements; therefore, there is a negative correlation.

For the comparison between the customer requirements and the engineering requirements, three symbols are used to denote a relationship. For a strong relationship, a filled in circle was used. A medium relationship was given a hollow circle. Weak relationships were given a hollow downwards facing arrow. In order to make the final calculation for the most important engineering requirements based on the customer requirements, each of the relationships are also assigned a weight. Strong relationships have a weight of nine, medium relationships have a weight of 3, and weak relationships have a weight of one. To explain the procedure, take the customer requirement of making sure to cost of the device stays below $250, and the engineering requirement that the frequency range of the transceiver is between 900 MHz and 930 MHz. The relationship between these two requirements can be considered strong, as the functionality of the transceiver frequency range is in most cases, directly influenced by the cost of the transceiver and antenna. Therefore, should the frequency range of the transceiver increase, the final cost of the device would most likely increase in proportion, which would signify that there is a strong relationship between the two requirements. Conversely, the customer requirement that the Wilderness Information Link Device be transportable and the engineering requirement that there be status LEDs based on the current state of the device have a weak relationship as there is not direct link between the two requirements.

The final weight of importance percentage is assigned based on the weight given to each of the relationship symbols and the relative weight assigned to each of the customer requirements. We can see from the House of Quality diagram that the engineering requirements with the least weighted percentage at 3% are the requirement that the software will conform to the microcontroller code space and that the device will have status LEDs to signify a transmission as occurred as well as the state of the device. The engineering requirement with the highest weighted percentage at 23% is the requirement that the solar panel shall generate at least 1 Watt. The second more important requirements at 19% are that the power rating shall be within 0.5W and 3W, and that the software shall be designed with low-power use and performance in mind.

To aid in the understanding of Figure 2: House of Quality, Figure 1: House of Quality Legends is shown to display each of the symbols that are used in the House of Quality diagram. The symbols for correlation, direction of improvement, and relationships as well as the weight assigned to each are shown in the corresponding tables.

Correlations	
Positive	+
Negative	-
Strong Positive	++
Strong Negative	- -
No Correlation	

Relationships		Weight
Strong	●	9
Medium	○	3
Weak	▽	1

Direction of Improvement	
Maximize	▲
Target	□
Minimize	▼

Figure 1: House of Quality Legends

Figure 2: House of Quality [4]

Relative Weight	Customer Importance	Customer Requirements	Transceiver Frequency Range 900-930 MHz	Power Rating within 0.5W-3W	Functioning Keyboard	Display ASCII Characters from Keyboard Input	Status LEDs	Software Design for Low-Power Use	16 Character Limit for Send/Received Messages	Withstand Drop/Impact of 3 ft	1W Generation from Solar Panel	Software will conform to MCU code space
		Direction of Improvement	▲	▼	□	□	□	▲	▲	▲	▲	▼
33%	5	Battery Operation of at least 4 Hours	▽	●	▽	▽	▽	●	▽	▽	●	▽
27%	4	Recharge time at maximum, 2 Hours	○	●	▽	▽	▽	●	▽	▽	●	▽
20%	3	Easily Transportable	▽	▽	●	▽	▽	▽	▽	●	●	▽
13%	2	Ergonomic	▽	▽	●	○	▽	▽	○	○	○	▽
7%	1	Cost Should not exceed $250	●	●	○	▽	▽	●	▽	○	○	▽
		Importance Weight	6%	19%	11%	4%	3%	19%	4%	9%	23%	3%

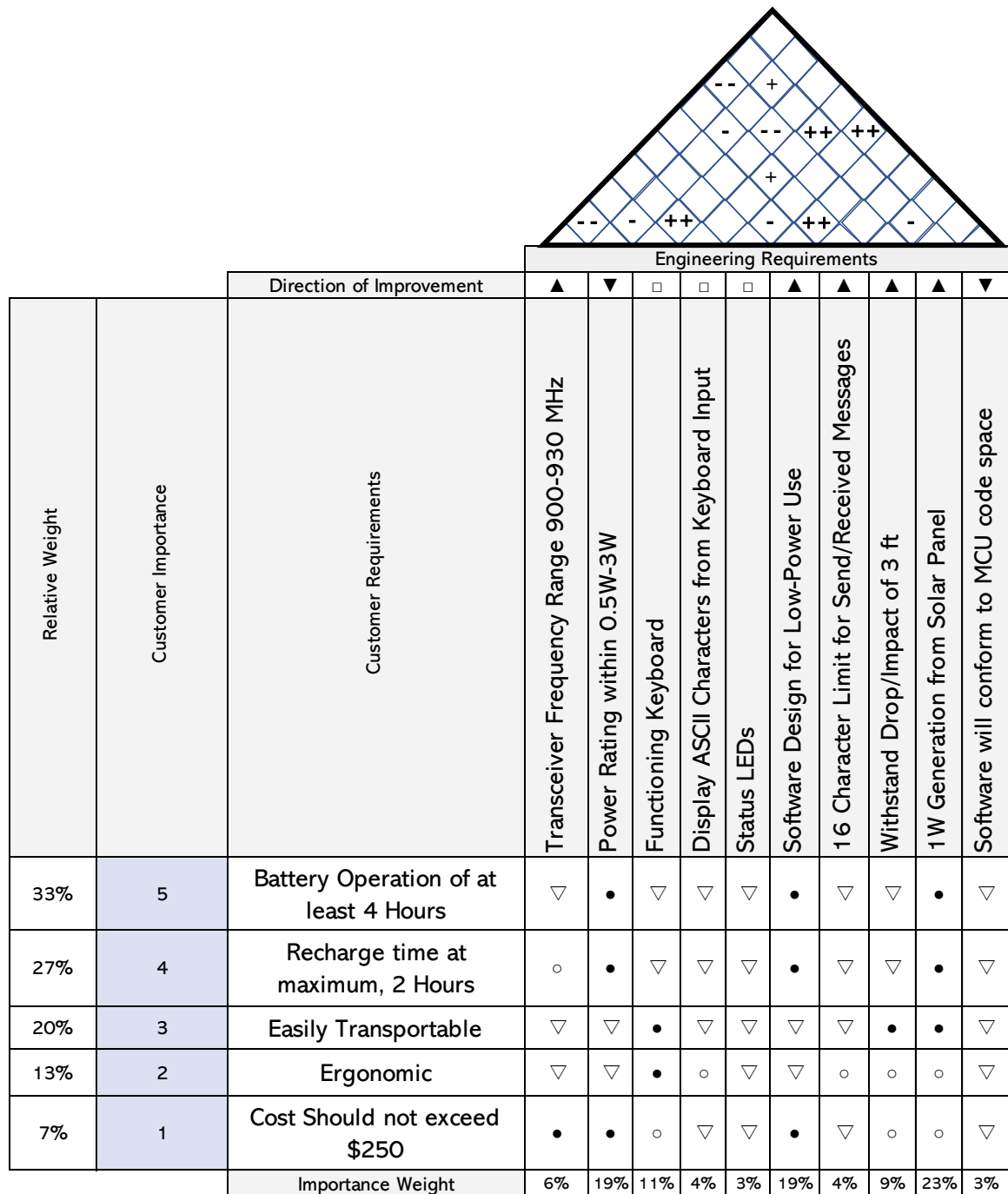## 2.5 Requirement Justifications

The Wilderness Information Link Device (WILD) is a device founded based on the need for a device capable of providing a means of emergency communication

between a sender and receiver. Considering the primary use environment of WILD, the device should be able to function using as few required reliable and self-sustaining resources as possibly capable. The creation of the device is a product of each of the requirements and constraints needed to ensure that the device operates as intended and will meet the needs of the customer in a way to guarantee the functionality required of it, based on WILD's initial purpose. Therefore, it can be said that the justification provided for each of the posed engineering requirements and customer requirements is both an explanation of why each requirement is established as important, as well as a fundamental validation of the overarching purpose and objective of the Wilderness Information Link Device. The requirements chosen to represent the most relevant and important to the core tenets of WILD are listed in the vertical and horizontal columns and rows as seen in Figure 2: House of Quality—However, it should be noted that the requirements listed do not encompass all objectives of WILD, and simply cover the most relevant and impactful requirements established that hold the most weight in the overall design and construction of the device.

2.5.1 Engineering Requirements

The engineering requirements most relevant to the Wilderness Information Link Device involve the requirements pertaining to the internal design and functionality of the device. These types of requirements usually do not encompass the requirements relating to the primary customer interaction of the device or the visual aesthetic of the device. These three types of requirements may overlap—however, the engineering requirements at their core mostly deal with how the device will be designed, regardless of usability and visual appeal.

Transceiver Frequency Range Between 900-930MHz: Considering the regulations surrounding frequency band usage as well as what is feasibly available to use for the production of WILD, the frequency band selected for WILD was 915 MHz which falls within the approved bandwidth of 902 MHz and 928 MHz. A frequency of this magnitude would ensure that the bitrate of the device would be as high as reasonably capable. As seen in section 3.2.2.1 Technology, a frequency of 915MHz provides reasonable obstacle penetration, which would allow for the usage of the Wilderness Information Link Device in non-rural areas, which are outside of the main scope of the device expectations.

Power Rating within 0.5W-3W: This power range selected for the Wilderness Information Link Device was chosen with all relevant materials needed during the production and prototyping of the device, including the solar panel, microcontroller, transceiver, keyboard, display, and other devices used, considered and accounted for. This range of 0.5W to 3W is a key factor in the core functionality of WILD as it provides a set limitation to fall within for all devices used to create WILD. This importance is based on the consideration that there are two main sources of power for the device: solar charging power and USB charging power. Due to the limitations posed by both methods—particularly the means by which the charging methods may be used as well as when—a power rating range must be established

and followed to make sure that WILD, and all relevant parts that make up the final device, are able to function and run as expected.

Functioning Keyboard: This requirement exists so that the user will be able to properly interface with WILD and each advertised function as seen through the display. As this engineering requirement only states that the keyboard must function, there are no limitations posed by this specific requirement that state exactly how this requirement must be met. At minimum, the user must be able to interface with the display using provided keys in order to select their desired function, such as to send a message or to view messages that have been received. Therefore, the only keys that are necessary to ensure that the needs of this requirement are met, are to have a key that allows selection of an option, and navigational keys—such as keys that travel left and right. A full-fledged keyboard is not required as the means to type in a message can be accomplished using two keys that cycle between each letter of the alphabet and certain special characters to provide a means of punctuation. This requirement also does not specify exactly how the keyboard must be designed, which therefore implies that the keyboard may either be constructed entirely using a unique design or purchase a proprietary part or design from an outside distributer.

Display ASCII Characters from Keyboard Input: Following along with the previous requirement to have a keyboard that is deemed functional, an additional requirement relating to the keyboard is to make sure that input provided from the user by way of the keyboard appears on the display of WILD in the form of ASCII characters. ASCII is a means to display text characters using computers, and—for the purposes of the Wilderness Information Link Device—includes the alphanumeric characters most commonly seen in the North American English language. These ASCII characters include other characters as well, including but not limited to '.', and ' ', or the full stop symbol and whitespace symbol. Although alternative configuration can easily be reconfigured depending on the desired target audience, it should be mentioned that displaying the alphanumeric characters most commonly seen in North American English is also a limitation on who may reasonable use WILD for wireless and self-sufficient communication.

Status LEDs: In order to properly communication to the user the current status of the Wilderness Information Link Device, it is a requirement that status LEDS be used to ensure that the user may at all times know the current actions being performed by the device. An LED will be used to show that the device is charging, in the form of a blinking yellow LED that performs its blinking action during the times when the device is either charging through USB-C or through solar charging. Should the device ever be in a state where it is not charging and idle, the yellow LED will instead illuminate solidly without blinking to show the user that no current actions are taking place. A green LED will illuminate solidly to show that a message has been successfully sent. Moreover, a red LED will illuminate solidly to show that a message has been received and is ready to be viewed by the user. It is important that the user be able to effectively and efficiently understand the current state of the device in order to limit the customers need to view the display to see the state and the quickly ascertain the status of WILD by using a simple glance instead of

continuous physical effort which may result in increased battery use in the long run.

Software Design for Low-Power Use: Ensuring that the Wilderness Information Link Device is equip with software designed for low-power use is a requirement which poses limitations and expectations on primarily the software side. By using low-power mode states available to the chosen microcontroller as well as creating efficient code that allows the device to run and perform all tasks without expending any unnecessary power, the device will be able to fulfill this requirement as needed to ensure that WILD is able to efficiently and consistently perform in the target environment. Making sure that the internal software is properly optimized for low-power usages would allow the device to both perform efficiently as well as fulfill the alternative customer requirement that the device must be able to function while running on battery power for a minimum of four hours.

16 Character Limit for Sent and Received Messages: The requirement for WILD that the device must be able to send and receive a message that is limited to sixteen characters must be clarified. The initial limit set in place for the device is limited to sixteen characters is the standard target value. Should it be deemed feasible to institute a design that exceeds a sixteen-character limit, this alternative route shall be taken as long as no other requirements for WILD are infringed upon, such as limitations posed to space/storage requirements and other related requirements. This limit of sixteen characters ensures that the message created by the user as well as the message received is able to appear fully visible on the character display used for the device. There are two possible options that may impact the limit on the length of the message: the number of characters that are able to appear on the character display are once, as well as if the device display and software will allow for scrolling capabilities. If the character display is able to display more than sixteen characters at once, then the limitation on the length of the message may be increased. Additionally, if the display is able to scroll the message such as only a portion of the message is able to appear on the display at once, then the sixteen character limit requirement may also be relaxed as desired as long as no other relevant requirements are altered and negatively affected in the process.

Withstand Drop or Impact Present at 3ft: As the Wilderness Information Link Device has a target usage environment of outdoor, mostly rural locations and would most likely be used in a handheld fashion, there is a requirement established that ensures that WILD should be able to withstand drops or impacts that would occur after being released from a height of three feet. This type of drop resistance would provide considerations for the device being dropped from a height of three feet onto varying surfaces, including concrete and grassy terrain. This requirement would have a direct impact on the configuration of the core devices present within WILD, such as the display, the solar panel, and the keyboard. Those three components would also be affected by the material and manufacture quality of the encasement chosen to hold the main elements of WILD in a safe and physically manageable vessel. Therefore, this requirement altogether states that the device will maintain expected functionality and usability after being dropped from a height

of at maximum three feet onto varying surfaces that will be reasonably encountered, such as concrete and grassy terrain.

1W Generation from Solar Panel: This requirement states the minimum power generation expected from the solar panel seen on the Wilderness Information Link Device in order to ensure that all functionality of the device is adequately powered and maintained for continued and sustained use when using solar power as the source of charge. Core elements of WILD that are directly influenced by this requirement include the display, the keyboard, status LEDs present on the device, as well as the battery used to provide power to all components connected. This is a requirement that can be expanded upon in order to potentially provide more power to the device; however, any alterations to the capabilities of the solar panel desired for use must not infringe upon any of the other requirements in a way that would further restrict or lesson the overall capabilities of WILD.

Software Conforms to Microcontroller Code Space: This engineering requirement for WILD pertains mostly to the capabilities of the software. All software used to enable the functionality of the core components of WILD, including the display, status LEDs, and keyboard must be thoroughly optimized and perform as expected within the established confines of the chosen microcontroller code space. The created code must not exceed what is allowed and able of the microcontroller in order to endure that all required specifications relevant to the Wilderness Information Link Device performs as expected and as presented.

2.5.2 Customer Requirements

The customer requirements pertaining to the Wilderness Information Link Device are the necessities most relating to how the customer interacts with the device at the surface level. Therefore, needs that involve the inner workings of the device, such as how the software must be coded, or what the mechanical requirements of the device must conform to, remain separate to the customer and are therefore excluded.

Battery Operation of At Least 4 Hours: As the Wilderness Information Link Device has an intended and targeted use environment of areas where cellular communication may be difficult to guarantee and the use of satellite communication may be met with atmospheric limitations, it is recognized that these particular environments may also be ones in which access to reliable charging may also be restricted. From a full charge scenario to complete battery depletion, WILD should stay functional for a minimum of four hours. Considering the solar charging state and capabilities available to the device, this charge depletion time may be over the expected four hours minimum. This limitation would be communicated to the user and would ensure that the following of two scenarios would occur: that this allotted time would give the customer enough time to travel from a location with no standard means of communication to a location where either these common communication standards would exist or that non-solar powered charging methods would be available for use. The second scenario would be that the four

hours of minimum non-charged use would be enough time to outlast any weather conditions that would limit the ability to charge the device by using solar power.

Recharge Time of At Maximum 2 Hours: WILD is capable of two distinct charging methods: solar power charging by way of a solar panel and charging via a cable. As charging the device using solar energy will mostly be a constant source of sustained charge, given the correct conditions for solar charging, the requirement ensuring a recharge time of at maximum two hours, applies mostly to the charging that occurs when the supplied USB-C cable is used. This recharge time references the period of time required to charge the device from a state of complete battery depletion to full charge.

Easily Transportable: The Wilderness Information Link Device has an expected usage environment of mostly rural or underdeveloped locations where standard communication standards, including cellular and satellite communication, are either unavailable or inefficient. Furthermore, it would not be illogical to assume that the target consumer audience for the device would include travelers, hikers, climbers, or other individuals who find themselves in environments where WILD would thrive and be the preferable means of communication. Considering this target group, it is established that the customer would be most suited to a device that is easily transportable. In other words, the size and transportability of the device is a detail that would be of direct concern to the user. The definition of 'easily transportable' encompasses the ability to move and carry the device within standard wilderness holding devices, including backpacks, satchels, and in rarer cases, pockets available on the clothes of the user. Other than the size of the device, the weight of the device would also be included in deciding whether the device is easily transportable or not. This weight would be a target value less than ten pounds, which is considered a feasible weight to transport for most average healthy adult humans.

Ergonomic: Considering the ergonomics of the Wilderness Information Link Device involves mostly the consideration of how comfortable the customer should feel while interacting with the device. This requirement involves the overall size of WILD, the size and positioning of the keyboard, the positioning of the screen, and the location and size of the solar panel. The device should be constructed with the goal to ensure that the device is handheld. The initial plan is that WILD will be able to be operated and held within a single hand; however, the specifications can be relaxed to a point where the device is able to only be held with a minimum of two hands. With ergonomics in mind, the keyboard that will be used for the Wilderness Information Link Device is a blackberry inspired keyboard that is able to be used with either one hand or both, depending on the desires of the user. Therefore, all necessary keys will be in reach and able to be accessed and used using the thumb of a single hand. Should these capabilities be relaxed to a twohanded configuration, the user will be able to use both thumbs to select all keys needed for the use of WILD. The size and orientation of the keyboards, display, and solar panel shall result in an overall configuration that is comfortable to hold and successfully use by the customer.

Cost Should not Exceed $250: The requirement that the cost of WILD shall not exceed $250 ensures that the target consumer base for the device will be able to feasibly afford to purchase WILD, as well as to provide a reasonable fund limitation to be used during the research, design, and prototyping phase that will allow all necessary parts and tools to guarantee the functionality of the device are able to be acquired. As seen in Table 12: Budget, the total cost of each individual module is estimated at being near $198.82. This value is below the maximum value of WILD of $250, which would also confirm that a profit would be made from the distribution of the device. However, should the cost of production of WILD exceed the set maximum of $250, it should be noted that the most expensive component of the device is the transceiver. If the maximum cost of production cannot be adjusted, the type of transceiver used would likely be the first component altered to fit the economic needs of the user as well as of production and prototyping.

2.6 Project Block Diagrams

Anticipated Hardware Block Diagram
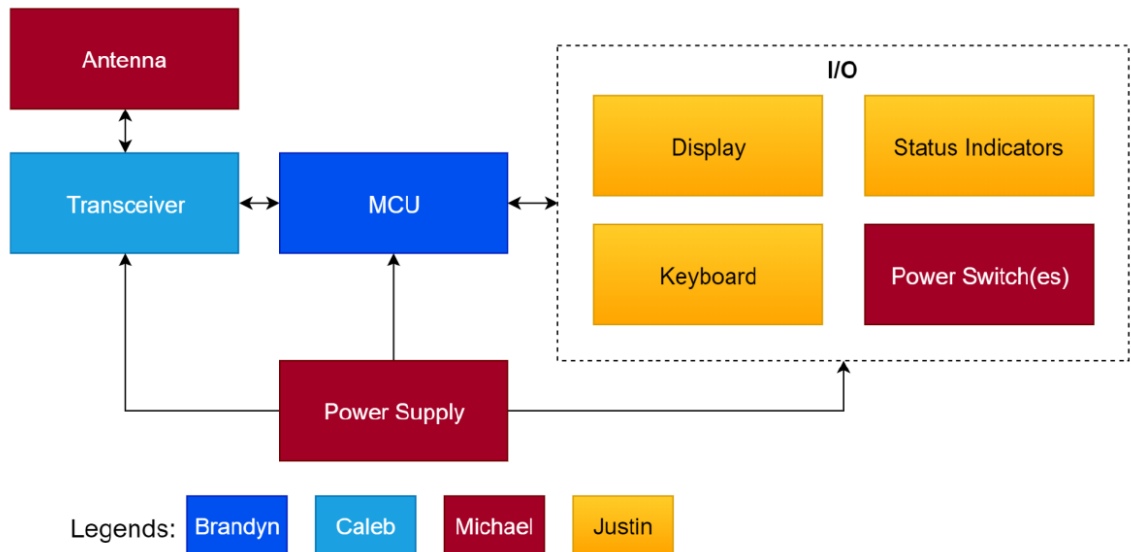
Figure 3: Hardware block diagram

Considering the diagrams shown above, a list of labeled blocks concerning the modules of our project are linked together. We have currently assigned the appropriate parts of the device to each of our members in accordance to their respectable role and position. As to be expected, most of the hardware modules will be handled by the members who study electrical engineering. The hardware that specifically relates to software aspects will be overseen by the members who study computer engineering.

Expanding upon the hardware illustration, the microcontroller is the backbone of our device, as the integrated circuit will be making all the necessary processing and handling of our device. A power supply unit will be researched and designed for powering on the entirety of the system. The input and output as well as the transceiver will be communicating to the MCU bi-directionally in that data will be transmitted between both, however not simultaneously. For the I/O, we intend to have three main sections: keyboard interfacing, a display screen, and status indicators. As of now, research is still currently being conducted to help attain the most feasible implementation of the PSU and I/O modules.

Anticipated Software Block Diagram
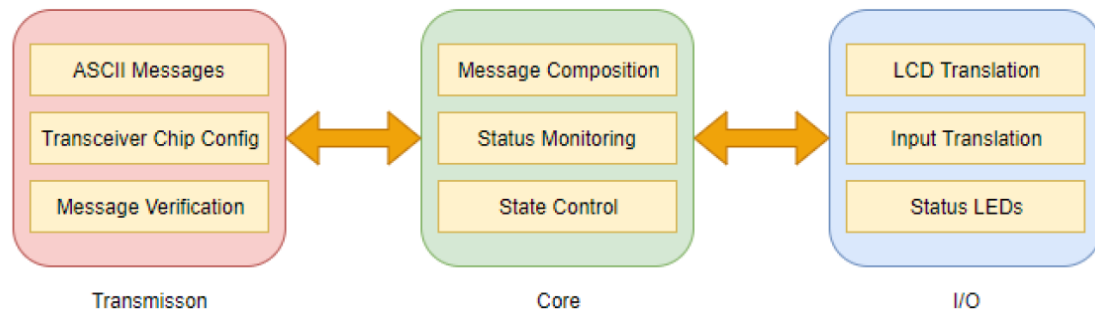
Figure 4: Software block diagram

Transitioning focus toward the computer block diagram, there are three main sections we have composed. They are the transmission, core, and input/output. The core will consider the message's composition, as well as status whilst a state control will be provided. Making an input towards the MCU will act as the translation between physically pressing the key button and then generating an output to the display screen. Status LED's will also be incorporated to help acknowledge the current position of the message of interest. After proper configurations to the transceiver have been made, transmissions of a message will then be promptly verified by the MCU which in turn will communicate to our status indicators.

2.7 Project Milestones

The broad milestones encompassing the length of the Wilderness Information Link Device project include four main phases: Initial Ideation and concept phase, the Research phase, the Design phase, and the Prototyping, Testing, and Finalization phase. Figure 5: Milestone Timeline, shows a graphical representation of each of the phases and describes in succinct terms the main goals within each of the four stages. The diagram shows the beginning term in Spring 2019 and the ending term as Summer 2019.
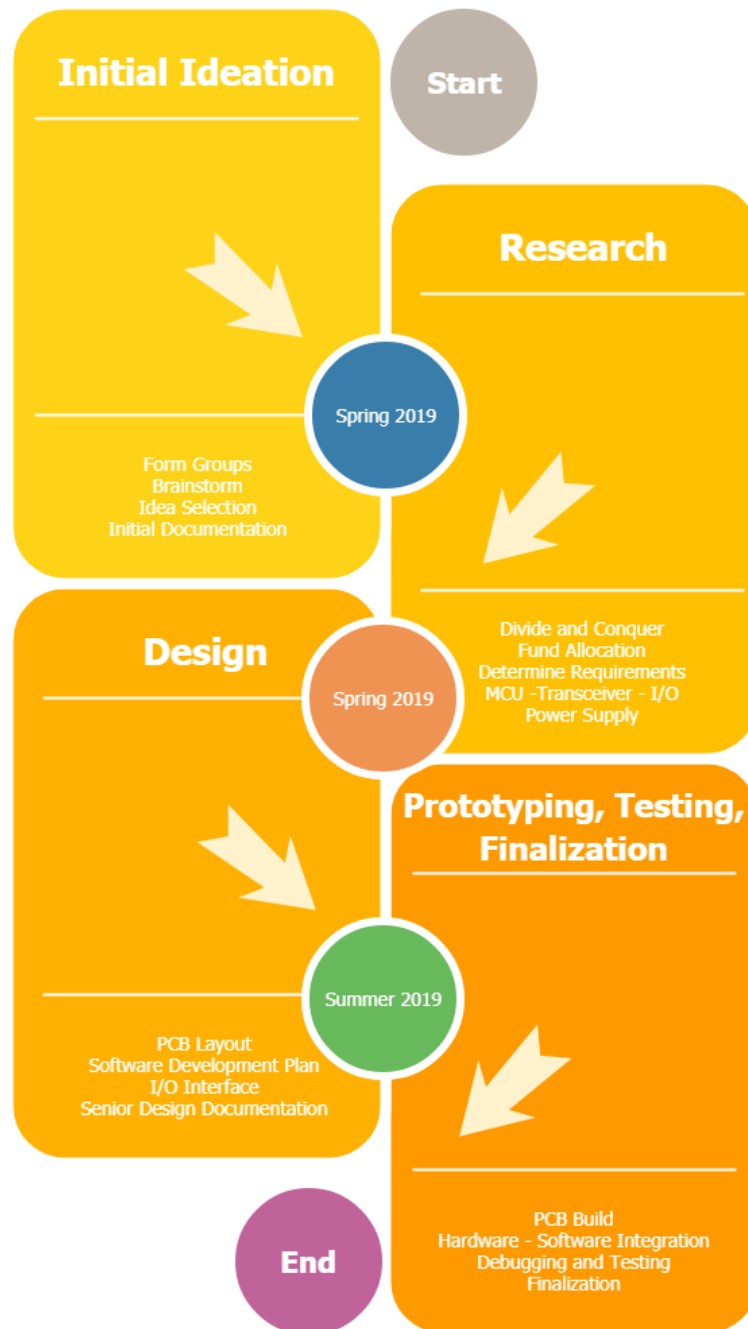
Figure 5: Milestone Timeline [3]

As the ideation, development, and construction period for the Wilderness Information Link Device will span the course of over six months, thorough planning for each stage was made a priority. The complete timeline was split into each of the four main sections: the initial idea creation process, the research phase, the design phase, and the stage encompassing all physical creation, such as prototyping, testing, and the final wrapping up of all previous efforts. For clarity, each of the four main sections have been emphasized by showing separate complete plans of each stage, as it pertains to the relevant milestones of the project. It should be noted that the research and design phases are predicted to overlap each other's stages as the two go hand-in-hand—research will also likely be done during the final stages of the project as well, as a result of all troubleshooting and difficulties that are likely to be encountered.

The beginning of the Wilderness Information Link Device, WILD, began with the initial ideation process. This process is mapped in Table 4: Initial Ideation Process Milestones, which covers the formation of the group, provides a period of brainstorming lasting multiple weeks, and leaves off with the process of singling out a specific and well-constructed proposal for the final choice of the project to be researched, designed, and constructed.

Deliverables	Term	Date Begin	Date End	Responsible	Status
Initial Ideation Process					
Form Groups	Spring 2019	1/8/2019	1/8/2019	Team	Complete
Project Idea Brainstorming	Spring 2019	1/8/2019	1/22/2019	Team	Complete
Pick final choices for consideration	Spring 2019	1/22/2019	1/25/2019	Team	Complete
Idea Selection	Spring 2019	1/24/2019	1/28/2019	Team	Complete

Table 4: Initial Ideation Process Milestones

Following the stages encompassing the initial brainstorming and selection of the project, is the research phase. This phase is stepped through in Table 5: Research Milestones, which shows topics to be covered, as well as essential decisions to be made with reference to the administrative portion of the project. These administrative procedures include documentation covering the initial stages and brainstorming portion of the project, mostly seen in the initial ideation stage. Main

concerns for research for the project were decided upon assigned to the most relevant individuals or teams. These topics include microcontroller research, research pertaining to the transceiver and antenna, as well as research on the power supply to be designed and used. Meetings involving the administrative documentation were scheduled as well.

Deliverables	Term	Date Begin	Date End	Responsible	Status
Research					
Divide and Conquer Document (Initial Project and Group Identification Document)	Spring 2019	1/22/2019	2/1/2019	Team	Complete
Decide general budget and source of funds	Spring 2019	1/15/2019	1/29/2019	Team	Complete
Assign roles	Spring 2019	1/22/2019	1/29/2019	Team	Complete
Determine Official Requirements	Spring 2019	1/31/2019	2/1/2019	Team	Complete
MCU research	Spring 2019	1/29/2019	2/14/2019	Brandyn	Complete
Transceiver and Antenna research	Spring 2019	1/29/2019	2/14/2019	Caleb	Complete
I/O research	Spring 2019	1/29/2019	2/14/2019	Justin	Complete
Power Supply research	Spring 2019	1/29/2019	2/14/2019	Michael	Complete
Meet with Professor (Divide & Conquer)	Spring 2019	2/5/2019	2/5/2019	Team	Complete

Table 5: Research Milestones

The research phase both leads into and precedes the design phase, as seen in Table 6: Design Milestones. In order to most efficiently facilitate the design phase, research involving each of the main areas to be covered—such as microcontroller research, transceiver and antenna research, power supply research, and I/O research—will be continued and expanded upon in a more material aspect. Both

the construction and the early stages of realization of concept will be in the design phase. Instead of individuals researching their topics of study and choice, team members will group up in this stage to accomplish the tasks outlined in Table 6.

Deliverables	Term	Date Begin	Date End	Responsible	Status
Design					
PCB Layout and Design	Spring 2019	2/7/2019	3/28/2019	Michael/Justin	In Progress
Embedded Software Package Plan	Spring 2019	2/7/2019	2/28/2019	Brandyn/Caleb	In Progress
Design of Power Supply	Spring 2019	2/7/2019	3/28/2019	Michael/Justin	In Progress
I/O Interface Design and Planning	Spring 2019	2/7/2019	3/28/2019	Michael/Justin	In Progress
Embedded Software Development	Spring 2019	2/28/2019	4/23/2019	Brandyn/Caleb	In Progress
60 Page Documentation Draft	Spring 2019	2/7/2019	3/29/2019	Team	Complete
100 Page Documentation Submission	Spring 2019	3/29/2019	4/12/2019	Team	In Progress
Final SD1 Document	Spring 2019	4/22/2019	4/22/2019	Team	In Progress

Table 6: Design Milestones

After all necessary elements of the Wilderness Information Link Device have been researched and designed, the prototype for WILD will be assembled. Based on the results of integration, changes may be made and previous stages (most notably the research and design phases) may have to be revisited to ensure complacency with all established requirements and expected behavior. The hardware and software side of the Wilderness Information Link Device—sides which in previous stages will have been mostly separate—will be integrated in the prototyping, testing, and finalization phase. All steps for the final completion of WILD, following the initial ideation phase, research phase, and design phase, are referenced in Table 7: Prototyping, Testing, and Finalization Milestones.

Deliverables	Term	Date Begin	Date End	Responsible	Status
Prototyping, Testing, and Finalization					
PCB Build	Spring 2019	3/28/2019	4/23/2019	Michael/Justin	To Be Completed
Hardware/Software Integration	Spring 2019	4/23/2019	5/6/2019	Brandyn/Caleb	To Be Completed
Debugging and Testing	Summer 2019	4/23/2019	7/19/2019	Team	To Be Completed
Revisions	Summer 2019	4/23/2019	7/19/2019	Team	To Be Completed
Final Report	Summer 2019	5/13/2019	7/26/2019	Team	To Be Completed
Final Presentation	Summer 2019	5/13/2019	7/26/2019	Team	To Be Completed

Table 7: Prototyping, Testing, and Finalization Milestones

2.8 Project Stretch Goals

There are several items that we would like to add to this project before its completion, but that have been determined as of yet to be beyond what will likely be completed by the end of the project. While these items would add functionality or usability to the device that has been deemed desirable, they are not essential for the achieving the basic functions of the device as laid out in the requirements specifications.

First, we would like to implement *mesh networking;* this is as opposed to the basic goal of the project to implement *peer-to-peer networking*. In a peer-to-peer networking scheme, each device communicates directly with each other by establishing a direct connection between the two of them. The scheme may or may not support establishing the identity of the device(s) with whom it is communicating, but must be able to send and receive information between each device with which it intends to communicate.

As such, it is limited by the range of its wireless hardware – any devices not within its own radius of wireless range are entirely inaccessible. A possible alternative to this arrangement is the use of mesh networking, where a "mesh" would be established by several devices in an area.

Essentially, each device in an area becomes a node in a network, allowing the devices to "play a game of telephone" with any messages. For example, if there are three devices in an area, and two of them are out of reach of each other but the third device is in reach of them both (between the two far devices), one device can send a message intended for the third. Because the third is not found, it is passed on to the next device in the area who is told to pass it on to the third. The second device will then send on the message the same way that the first device did, asking the recipient. Due to the third device being within range of the second, the transmission will be successful, and the second device will let the first know that it managed to find the third device and relay to it the message.

Naturally, the complexity of programming involved in this system is significant, and while it would improve the utility of the device by expanding range in areas where other devices exist, it is not necessary for the satisfaction of the project requirements specifications.

Our second stretch goal would be the implementation of a power back-feeding system that would allow the user to use the internal battery of the device as a power supply for an external device via the USB port. Since a relatively large-capacity battery is planned for use in the device, the ability to access some of this capacity for other devices in field situations is of great utility. However, this is not a requirement for the project, and an added complexity to the already-complex power supply.

Our final stretch goal is the addition of a battery monitoring system so that the user would not have to guess how much charge is left in the battery. This could be implemented either via software or hardware; an implementation rooted in software would require both hardware and software changes that would come to be more significant than would be something that can be added purely for ergonomic-type considerations; an implementation based in hardware might be simpler in several regards, but has also been thus-far disregarded on account of avoiding adding complexity to the already-complex power supply, as with our second stretch goal.

3 Research

In the following pages, thorough research surrounding the project will be found. It is imperative that this be a slow and rigorous process to ensure that proper data and approaches are met. This will specifically be aimed to ensure that our specifications will be met. Our method of research, although not limited to, will consist primarily of using internet, textbooks, and human resources. We intend to utilize as many videos, articles, and hobbyist-relevant projects that will give us the substantial edge in the designing phase, with proper citation included. Starting off the research stage, we will first go into the depths of the hardware aspects, and then software. Particularly, the I/O section of this project will be the very first thing covered as it serves as the main physical functionality that the user will interact with, and thus would make logical sense to be covered first. It is also necessary as all modules used within the project must be determined before further design such as with the PSU can be made. This is so that specifics such as power and min/max voltage and current ratings can be accurately established.

3.1 Existing Product Summary

Looking into already-existing products on the market will help us ascertain certain attributes that may be necessary for implementation of our own design. It is understood that these products are sold and must take into account the many standards and qualities that must be upheld for customer demand. Such qualities may include the ergonomics, ease of use, functionality, costs etc. It is important to account for these factors as it may potentially reflect off of our device. One such existing product that is relevant is a dual band radio with keypad functionality. A specific model is the "BaoFeng UV-5R", a two-way radio. This specific product contains a variety of features, such as the keypad, LED indicator, LCD display, frequency switches, antenna etc.

3.2 Hardware

It is necessary to establish the relevant hardware technologies that will potentially be used in our design stage. Our project is geared towards the user being able to comfortably type characters using a keyboard, and ascii characters being output to a display screen. This message should then be accurately and efficiently transmitted across a medium between two or more transceivers. Thus, it's imperative that the hardware modules of interest must be thoroughly researched. In doing so, this will help achieve our goals whilst adhere to the specifications listed earlier. These hardware modules will vary, although the main components are: a

microcontroller, keyboard, display screen, status LEDs, power supply, printed circuit board, transceiver units, power switches, solar panels, and relevant circuit components. We intend to interface a keyboard with ergonomics that make typing a pleasant experience, whilst also promptly generating the correct ASCII characters entered. The screen display is most likely going to be an OLED, and although this does use slightly more power than that of a LED screen, the quality will be worth it. As far as the interfacing for the display screen, I2C will be used, a type of integrated circuit communication system that will be discussed later. Status LEDs will be utilized in order to show the current standing of the message being sent, and they will most likely be RGB to help further signify the whereabouts. Transceiver units will be the main components used for communication between MCU's, and the power supply will of course be the main source of powering the entirety of the device.

3.2.1 Microcontroller

Choosing the Microcontroller to be used for the Wilderness Information Link Device was a straightforward decision, as the parameters considered were fulfilled by multiple MCU options. The MCU chosen must have an adequate number of I/O pins needed to provide support for the keyboard interface, display, and other features integrated on the device. UART, SPI, and I^{2}C support would be needed to establish communication between the I/O devices, transceiver, and antenna. The MCU would also have to provide enough code-space to contain all code necessary to configure and run the final working module. Likely one of the most important factors in deciding which MCU to choose would involve how friendly to development the MCU will be. This would entail the existence of a native development board to do initial configuration, testing, and integration.

3.2.1.1 ATmega328P

The ATmega328P is an 8-bit AVR microcontroller developed by Atmel. 32KB of ISP flash memory is available for use—wherein ISP (In-system Programming) memory is classified as storage able to be configured and programmed after the microcontroller has already been integrated within the system. Additional memory includes 2KB of SRAM and 1KB of nonvolatile EEPROM. There are three total timers available for configuration: two 8-bit timers and one 16-bit. The clock for the MCU is 16 MHz. For communication, there is 1 UART (Universal Synchronous and Asynchronous Receiver-Transmitter, or USART more specifically), 2 SPI serial interfaces, and a two-wire serial interface for I2C communication. With 32 total pins, 27 are programmable I/O pins and there is a 6-channel 10-bit ADC (analog to digital converter) available. The MCU operates between 1.8 V-5.5 V and the current falls around 0.32 mA while in active mode, 0.2 uA in power-down mode, and 1.3 uA in power-save mode. Along with active mode, power-down mode, and power-save mode, there are 5 total possible power saving modes available for the ATmega328P. On average, the cost for the MCU is near $2.01 per unit [2].

The ATmega328P appears to have most capabilities needed, such as UART, SPI, I2C, and adequate I/O pins; however, the MCU was found to be lacking in almost all departments when considering if it would be able to efficiently and feasibly accomplish all tasks required.

Foremost, there was concern shown relating to whether the ATmega328P would be able to provide a large enough code-space to contain the necessary code. While 32KB of possible code would likely be enough for all functions, it would be best to have an MCU where available space would not have to be a concern during development. Although there were many possible implementations for the communication interface, using either a Bluetooth-inspired keyboard, a cursor configuration, or a T9 texting model, there would need to be an adequate number of I/O pins to accommodate the implementation chosen. 27 possible I/O pins would have been more than enough to implement the communication interface. The main deterrent from choosing the ATmega328P for use is the number of communication options. One UART, 2 SPI interfaces, and one I2C serial interface would provide a limitation to the number of possible additions to the project requiring communication to the MCU.

3.2.1.2 ATmega1284P

Similar to the ATmega328P is the ATmega1284P. The ATmega1284P is an 8-bit bit AVR microcontroller which has 128KB of ISP flash memory available for development. 16KB of SRAM and 4KB of of EEPROM are also provided. All other specifications seem to improve over the ATmega328P, other than the operating voltage which remains unchanged. The ATmega1284P has four total timers, two of which are 8-bit timers, while two are 16-bit timers. The clock for the MCU operates at up to 20 MHz. There are two UARTs, 3 Serial Peripheral Interfaces (SPIs), and 1 two-wire I2C interface. Along with an 8-channel 10-bit ADC, are 44 total pins where 32 are programmable I/O pins. Another improvement over the MTmega328P is the expected power consumption. While in active mode, there is a 0.4 mA current draw—in Power-down mode, 0.1 uA and in Power-save mode, 0.6 uA. The price per unit for the ATmega1284P is approximately $5.29 [1].

Had the choice between which MCU to use been between only the ATmega328P and the ATmega1284P, the clear winner would have been the ATmega1284P. With an increased code-space, more interfaces for communication between devices, additional programmable I/O pins, and a lower power consumption at normal operating conditions, the ATmega1284P seems to provide all that is necessary and desired for development. However, both Atmel MCUs are 8-bit microcontrollers. Developing and operating in the final stages with an MCU that is 16-bit or 32-bit would mean for much faster processing power than would be expected from an 8-bit MCU.

3.2.1.3 MSP430FR6989

Looking for alternatives to 8-bit MCUs led to the consideration of the MSP430 series of MCUs developed by Texas Instruments. TI's MSP430 MCUs are touted as being ultra-low-power, well-documented, 16-bit MCUs which are available in a large variety of configurations. The MSP430FR6989 is an MCU with a clock that operates at up to 16 MHz. There are 128 KB of nonvolatile FRAM and 2KB of SRAM available. With five total timers provided, all are 16-bit. For communication, there are two UARTs with automatic Baud-rate detection, four Serial Peripheral Interfaces, and two I2C interfaces with support for multiple slave addressing. Surpassing both the ATmega1284P and ATmega328P, the MSP430FR6989 has a 16-channel 12-bit Analog-to-Digital Converter. Like the previous MCUs, the MSP430FR6989 has an operating voltage range between 1.8 V – 3.6 V. The power draw for the ultra-low-power modes are 100 uA when in Active mode, 0.4 uA when on standby in low power mode 3, and 0.02 uA during shutdown. Likely more than could be needed are the number of general purpose I/O pins, which totals 83 pins for the MSP430FR6989. Per unit, the MSP430FR6989 costs $7.64 [5].

The MSP430FR6989 provides a wide array of functionality that could be used during and after development. The larger code-space of 128KB, number of communication interfaces for communication between the microcontroller and other devices, the presence of a unique development board, as well as the abundant number of I/O pins more than satisfy the needed development requirements. That the MSP430FR6989 is a 16-bit MCU rather than an 8-bit MCU also provides an advantage when it comes to processing power and use ability in comparison to the previous Atmel MCUs.

3.2.1.4 MSP430FR5964

Also within the MSP430 series is the MSP430FR5964—and 16-bit MCU with a 16 MHz clock, 256 KB of FRAM, and 8 KB of SRAM. The MSP430FR5964 is also within the MSP430FR599x line of microcontrollers, which comes with a Low-Energy Accelerator which dramatically increases processing speed, power, and efficiency when performing certain tasks. The MSP430FR5964 comes with six 16-bit timers, each with up to seven capture and compare registers. The Analog-to-Digital Converter component of the MCU has twenty input channels and a 12-bit SAR (Successive Approximation Register) ADC. Doubling the communication capabilities of the MSP430FR6989, the MSP430FR5964 has 4 UARTs, 8 Serial Peripheral Interfaces, and 4 I2C interfaces with multiple slave addressing. Moreover, there are 68 general purpose I/O pins for use. The operating range of the MSP430FR5964 is between 1.8 V and 3.6 V. While in Active mode, the current draw is 118 uA—500 nA while in standby using low power mode 3, 350 nA while in standby using low power mode 3.5, and 45 nA when in shutdown (or low power mode 4.5). The MSP430FR5964 is sold for slightly less than the MSP430FR6989, at $6.45 per unit [3].

The MSP430FR5964 is a clear contender for use as it lacks any drawbacks or failures that would prohibit its use in development. Its features are similar

to the MSP430FR6989, except that the MSP430FR5964 most-noticeably has twice the amount of code-space. More devices could also be linked to the MCU since all communication interfaces (UART, I2C, and SPI) were doubled in quantity. All pins on the MSP430FR5964, as also seen on the MSP430FR6989, have support for capacitive-touch.

3.2.1.5 MSP430FR5994

Present as an alternative to the MSP430FR5964, is the MSP430FR5994. The MSP430FR5994 also has the Low-Energy Accelerator seen in MCUs that are a part of the MSP430FR599x line of MCUs. This MCU was considered over the MSP430FR5964 due to the circumstances surrounding the development board. There is a development board available for the MSP430FR5964; however, the MCU used is the MSP430FR5994. Due to their similarities, it can be assumed that there would be no issue with using the development board in tandem with the MSP430FR5964 instead. Although, to maintain organization and simplicity, testing on a development board using a native MCU would be considered as preferable. The MSP430FR5994 is a 16-bit MCU with a 16 MHz clock. As with the MSP430FR5964, the MSP430FR5994 has 256 KB of nonvolatile FRAM and 8 KB of SRAM for use. There are twenty Analog-to-Digital Converter channels, with a 12-bit SAR. As with the MSP430FR5964, there are 68 general purpose I/O pins, each with support for capacitive-touch without the need for any external interface or configuration. There are a total of six 16-bit timers for use. For communication with the MCU, there are four UARTs, 4 I2C interfaces with multiple slave addressing abilities, and 8 Serial Peripheral Interfaces. The supply voltage range is between 1.8 V to 3.6 V, and for power consumption, there is an 118 uA current draw while in active mode, 500 nA in standby with low power mode 3, approximately 350 nA during standby using low power mode 3.5, and 45 nA during shutdown. The price for unit of the MSP430FR5994 is $7.46 [4].

The MSP430FR5994 has all the benefits of using the MSP430FR5964, apart from its own native support of the development kit. All differences would likely be in name only, between the MSP430FR5964 and MSP430FR5994, but would also make documentation of all MCU and development board usage much simpler.

3.2.1.6 MSP432P401R

The last MCU up for consideration is the MSP432P401R, which is a part of Texas Instrument's SimpleLink Microcontroller platform, instead of the former MSP430 series. The MSP432P401R is based around ARM (Advanced RISC Machine), which is an MCU/processor architecture that emphasizes lower power consumption, cost, and energy, while maintaining efficiency of use and capability. The MSP432P401R is a 32-bit MCU with a clock frequency that operates up to 48 MHz. 256 KB of FRAM and 64 KB of SRAM are available for use and storage. The SAR ADC has fourteen total channels and a 14-bit resolution. Considering communication, there are four UARTs, four I2C interfaces, and eight Serial

Peripheral Interfaces. Much more complex than the previous microcontrollers are the timers equipped with the MSP432P401R. There are a total of 6 timers—four are 16-bit timers and two are 32-bit timers. The supply voltage range is from 1.62 V to 3.7 V and there are multiple low power modes. Power consumption from each mode is 80 uA when active, 83 uA when in low-frequency active mode, 660 nA when using low power mode, 630 nA when using low power mode 3.5, 550 nA when using low power mode 4, and 25 nA when using low power mode 4.5. The price for a single unit is $7.46. The MSP432P401R is also supported by a development kit that is native to the MCU [6].

This MCU was chosen for consideration mainly due to its ARM architecture. As the Wilderness Information Link Device is one that is made using energy efficient, low-power, and low-cost means, a microcontroller using ARM would fit the expected needs and requirements due to the low-power and cost benefits that come from using the ARM architecture. It's 32-bit architecture is also a draw due to the vast benefits of 32-bit in comparison to both 16-bit and 8-bit. The processing capabilities gained from using 32-bit would far outweigh those gained from a smaller-bit architecture. Being able to test using the native development kit would also make integration into the final product a simpler and more organized endeavor.

3.2.1.7 Microcontroller Comparisons

Cost: The benefit of constructing and designing modern day embedded products is that the cost of most common individual parts is well within the means of the average consumer, designer, and entrepreneur. Based on the estimated budget allocation as seen in Table 12: Budget, the approximate amount of funds provided towards the purchase of a single microcontroller would be approximately ten dollars. This would bring the final price of all three microcontrollers needed for use at approximately thirty dollars, with leeway given to circumstances involving an excess need of microcontrollers.

Table 8: Cost Comparisons lists all six discussed microcontrollers under consideration as well as the price for each unit and the total cost should three of the microcontrollers be purchased for use. Logically, the simplest microcontroller, the ATmega328P, came out to be the cheapest in price at $2.01 per unit. This is likely due to the unsophisticated nature of the hardware in comparison to the other microcontrollers being considered. The ATmega1284P is the second cheapest, at $5.29 per unit. As purchasing three of these units would cover only half of the total allocated microcontroller budget, the ATmega1284P, as well as the ATmega328P, would be more than acceptable choices based on price alone. The remaining four microcontrollers form Texas Instruments are close in price, with the lowest (the MSP430FR5964) being $6.45 per unit, and the remaining three being either $7.64 or $7.46 per unit. All microcontrollers chosen for consideration would fall within the allocated thirty dollars limit for purchasing three identical models, so all choices would be acceptable when considering only cost. Ignoring the features and

qualities that would make the microcontroller a better fit for the Wilderness Information Link Device, we can see that the ATmega328P would win when it comes to being the cheapest microcontroller. The MSP432P401R and MSP430FR5994, however, are the most expensive at $7.46 per unit, and $22.38 for a purchase of three models.

	Cost (Per Unit)	Cost Total
ATmega328P [2]	$ 2.01	$ 6.03
ATmega1284P [1]	$ 5.29	$ 15.87
MSP430FR6989 [7]	$ 7.64	$ 22.92
MSP430FR5964 [5]	$ 6.45	$ 19.35
MSP430FR5994 [6]	$ 7.46	$ 22.38
MSP432P401R [8]	$ 7.46	$ 22.38

Table 8: Cost Comparisons

Operating Voltage: All of the microcontrollers under consideration follow a standard convention when it comes to the operating voltage range of the microcontroller. This convention allows the microcontroller to be used for most common peripherals that can be found in embedded development and allow for the designer and creator to easily and efficiently integrate the microcontroller in most of the desired applications without fear of device failure or design modification. As seen in Table 9: Operating Voltage Comparisons, all of the microcontrollers, apart from the MSP432P401R, have a operating voltage of 1.8V at their lower ranges. The ATmega328P and ATmega1284P both have a maximum operating voltage bound of 5.5V. The MSP430FR6989, MSP430FR5964, and MSP430FR5994 all have a maximum operating voltage bound of 3.6V. The only abnormality, by only a slight margin, is the MSP432P401R, with an operating voltage range of 3.7V to 1.62V, which is slightly higher in the maximum range and slightly lower in the minimum range from the other Texas Instruments microcontrollers.

The Wilderness Information Link Device would be suited towards devices with an operating voltage range of at least 3.6V to 1.8V. All of the microcontrollers under consideration either fall within the boundary or exceed the desired expectations. The Atmel microcontrollers (the ATmega328P and ATmega1284P) all have the largest operating voltage range which could allow for more flexibility in use, despite the fact that for the Wilderness Information Link Device, a maximum operating voltage of 3.6V would be sufficient. The operating voltage range of the MSP432P401R is the second largest and would also be more than adequate to use for WILD. Considering only the operating voltage range, the Atmel microcontrollers and the MSP432P401R would all be acceptable choices.

	Operating Voltage Range HI	Operating Voltage Range LO
ATmega328P [2]	5.5	1.8
ATmega1284P [1]	5.5	1.8
MSP430FR6989 [7]	3.6	1.8
MSP430FR5964 [5]	3.6	1.8
MSP430FR5994 [6]	3.6	1.8
MSP432P401R [8]	3.7	1.62

Table 9: Operating Voltage Comparisons

Power Consumption: One of the most important features and considerations of the Wilderness Information Link Device is how much power and energy is required for the device to function, as well as how long the device is able to stay active for. As the device relies on both rechargeable battery power and solar energy by way of a solar panel, design solution revolving around energy efficiency must be prioritized. When considering which microcontroller to use, the power consumption of each microcontroller under consideration must be evaluated.

Modern microcontrollers are usually equipped with various low power modes that help facilitate the designer/user's needs for an energy efficient microcontroller. These modes exist apart from the standard active mode of the device, and allow the designer to specific exactly how the microcontroller will run based on how the embedded product is operating in an instant of time and how the microcontroller must behave at a minimum in order to accomplish the task in the most efficient way possible. Table 10: Power Consumption comparisons lists each

microcontroller under consideration as well as the number of power saving modes and the current draw of the microcontroller—if the value is known—from using each operating state and low power mode available. Modes that are either not used by the microcontroller or have current values that are unknown are designated using the 'N/A' designator. Of all the considered microcontrollers, the MSP432P401R has the lowest active mode current draw, at 80 uA. Comparatively, the Atmega1284P has the highest active mode current draw at 400 uA. When viewing all possible low power/power-save modes, the ATmega328P has the highest power-save mode current draw at 1300 nA, while the MSP430FR6989 has the lowest low-power mode current draw, at 20 nA while in low-power mode 4.5. Considering both the low-power modes and the active mode current draws, the most viable microcontroller would be the MSP432P401R as it has the lowest current draw while in active mode, while only the second lowest low-power mode current draw at 25 nA. The MSP432P401R also has the most low-power mode options at six—similarly to the Atmega1284P, which also has six low-power modes.

	Total Power Saving Modes	Active Mode	Power-Down Mode	Low Frequency Active	Power-Save Mode	LPM 3	LPM 3.5	LPM 4	LPM 4.5
ATmega 328P [2]	5	320 uA	200 nA	N/A	1300 nA	N/A	N/A	N/A	N/A
Atmega1 284P [1]	6	400 uA	100 nA	N/A	600 nA	N/A	N/A	N/A	N/A
MSP430 FR6989 [7]	4	100 uA	N/A	N/A	N/A	400 nA	350 nA	N/A	20 nA
MSP430 FR5964 [5]	4	118 uA	N/A	N/A	N/A	500 nA	350 nA	N/A	45 nA
MSP430 FR5994 [6]	4	118 uA	N/A	N/A	N/A	500 nA	350 nA	N/A	45 nA
MSP432 P401R [8]	6	80 uA	N/A	83 uA	N/A	600 nA	630 nA	500 nA	25 nA

Table 10: Power Consumption Comparisons

Final Comparisons: Table 11: Microcontrollers Under Consideration lists the basic parametric of each of the six microcontrollers being considered for use with the Wilderness Information Link Device, Values being compared between the six

include the basic architecture (such as the length of the data bus in bits), the clock speed in MHz, the number of GPIO pins available for configuration, the number of communication options involving UART, SPI, and I2C, whether the microcontroller has native development board support for ease of testability, the space available within each microcontroller for design, as well as the number of ADC channels.

	Architecture	Clock	GPIO Pins	UART	SPI	I2C	Native Dev. Board Support	Storage / RAM	ADC Channels
ATmega328P [2]	8-bit	16 MHz	27	1	2	1	No	32 KB / 2KB	6
ATmega1284P [1]	8-bit	20 MHz	32	2	3	1	Yes	128 KB / 16 KB	8
MSP430FR6989 [7]	16-bit	16-MHz	83	2	4	2	Yes	128 KB / 2 KB	16
MSP430FR5964 [5]	16-bit	16-MHz	68	4	8	4	No	256 KB / 8 KB	20
MSP430FR5994 [6]	16-bit	16-MHz	68	4	8	4	Yes	256 KB / 6 KB	20
MSP432P401R [8]	32-bit	48-MHz	48/64/84	4	8	4	Yes	256 KB / 64 KB	24

Table 11: Microcontrollers Under Consideration

When thoroughly comparing each of the six microcontrollers with each other, the Atmel MCUs were quickly disregarded from the considerations. Both the ATmega328P and ATmega1284P MCUs are 8-bit, which is still useable within the scope of the project, but are both outshined by the other MCUs in processing capabilities which are either 16-bit or 32-bit. The number of GPIO pins for both Atmel MCUs was among the lowest as well. Additionally, the capabilities for communication between external devices and the MCU would be limited as well, given that there are few UARTs, Serial Peripheral Interfaces, and I2C interfaces for both MCUs when compared to the Texas Instruments MCUs that were considered.

The MSP430FR6989 is a 16-bit 16 MHz microcontroller that satisfies most of the desired MCU parameters. With the most GPIO pins out of the 4, as well as support for UART, SPI, and I^2C, the x6989 would likely support all desired capabilities; however, the code-space available in the x6989 was a concern, in addition to the low number of UART, SPI, and I^2C channels. The MSP430FR5964—also a 16-bit MCU—is similar to the MSP430FR6989, except for the larger amount of memory

and communication channels. This MCU would have been more desirable over the x6989, except for the lack of native x5964 development board support.

The most likely contender over the MSP430FR6989 and MSP430FR5964 was the MSP430FR5994. With the same benefits of the x5964 and the inclusion of a native development board, the x5994 provided an excess of features to the project. The only apparent downside would be the 118uA current consumption.

As the final Wilderness Information Link Device will be created with the purpose of being low-power, one of the most important considerations is which MCU would provide the lowest operating current. Assuming that the MCU is active and not operating in a low-power mode, the MCUs with the highest current are the ATmega328P and ATmega1284P, while the MCU with the lowest draw is the MSP432P401R. All options considered, there is favor shown towards the MSP432P401R, which has a current consumption of 80uA. Matching the specifications of the other MCUs in consideration, the MSP432P401R was also the MCU that would most fall in line with the project's expectations of efficient, low power operation.

3.2.2 Communications Device

Selection of the communication device was a lengthy process that involved 3 Steps. The first step was to determine the technology that would be used. This was important because we needed to find a technology that met all of the parameters of our project, while remaining in budget. Secondly, we needed to determine the specific brand of product within that technology. With any given technology, you will find a number of products that are produced inside that domain. We needed to find a product that not only worked for our application, but also was well suited for designing, developing and using the end product. Finally, the third stage was to select the specific revision of the chip for our product. In our search we noted that there were many products that came in a variety of versions. In order to ensure the optimal performance of our product, we needed to narrow down the different options to the one that suited our product the best.

3.2.2.1 Technology

While considering technologies, many options were considered. The process used to determine the correct choice was to refer to our requirement specifications. These requirements were seen as a bare minimum. If we could improve on a requirement to go above and beyond the minimum, we decided that we would, so long as such a decision did not have an adverse effect on the other requirements. Additionally, we needed to determine which of the requirements were most relevant to the communication technology used. Once we determined the requirements that were related to the device, we needed to define the general hierarchy for these requirements. This was necessary in the event that we had two competing options that had tradeoffs. By selecting the requirements that we thought would be the most important to emphasize we could ensure that we made the right choice in this category.

The most important category, we determined was the maximum range of the technology. Our reasoning for this was based on a number of observations. The first of these observations was that the range would be one of the most variable factors in the building and use of our device. While the power usage can be expressed as a clear maximum, for instance, the minimum expected range can vary greatly due the large number of environmental conditions. The best range for most devices is Attained in an area with clear line of sight and no interference. Line of sight is easy to define and thus easy to predict. With the range that we wanted to attain in this product however, there were a number of factors that affected the LOS (line of sight) range. These factors will be more deeply discussed in the requirements validation section of this document.

Interference was more difficult to predict. This is partially due to the fact that interference is defined differently for different technologies. The only way to plan

for this factor was to assume for the worst-case scenario as indicated in the documentations for the different technologies that we considered.

The second most important factor considered was the power rating of the device we chose. Because we wanted this device to function for long periods of time without an external power source, we needed to select a technology that was capable of functioning at the ranges that we desired without consuming inordinate amounts of power. While researching all the parts for this product, we found that in nearly all cases, the communications portion of the device would consume the most power when the device was at maximum power consumption.

The Third consideration was for the development environments supported. We found that most technologies supported a vast number of development environments, some of which would work for our project. Because of this, this was not a major factor in determining the best technology to use.

Finally, it should be noted that cost was also an important factor. With a development budget of one thousand dollars, we needed to ensure the device fit within that budget. To clarify the general budget for the communications device we performed the initial research for all the main components simultaneously, then allotted an appropriate and approximate cost for each part in our budget, leave some extra room in the event that some parts went over budget.
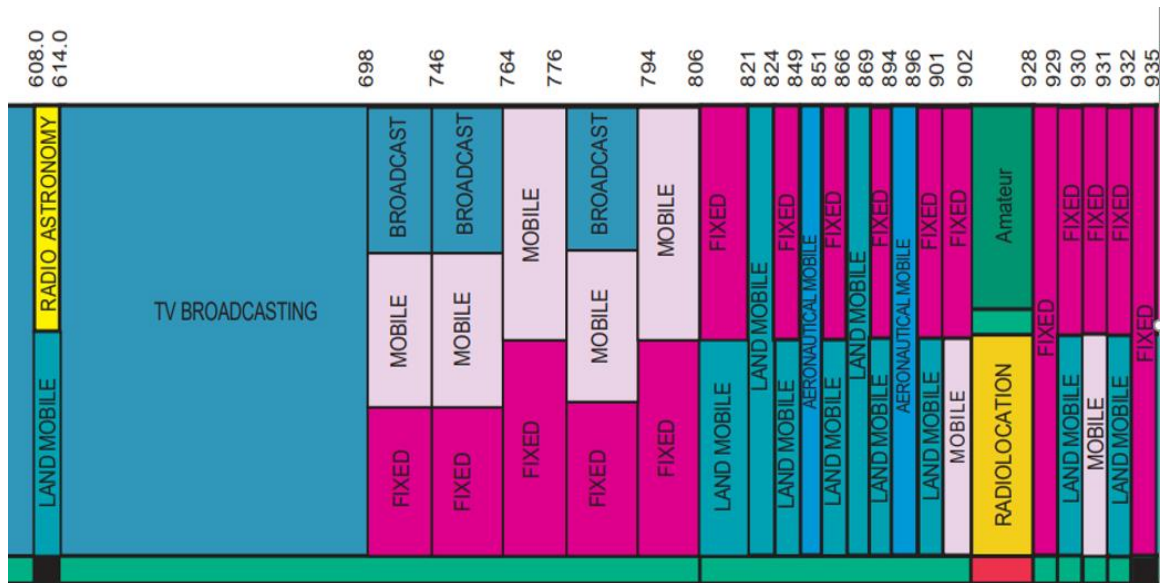
Figure 6: Usage of Frequency Band

While considering technologies that would work over the range that we required, it was established that the method of communication that best suited our needs was Radio Frequency. This family of communications devices was the only method of wireless connectivity that not only would allow connections to be made a very long ranges, but without the explicit need for line of sight. Given the possible use cases

for our finished product, we confirmed that this was the method that we wanted to use.

The next step was to determine which frequency to use. Radio frequencies cover a very large range, form about 3Hz to 3THz(Although not all frequencies in this range have well developed technologies). In most countries, including the United States these frequencies are highly regulated. Some frequencies are reserved for government use, while others are reserved for specific uses, such as cellular networks or broadcast television. It was extremely important for the success of our project to select a frequency that was open for our use. Additionally, there are regulations that specify the maximum peak amplitude for devices transmitting on these frequencies. This would greatly impact not only the range of our devices but also the power usage.

Range and power use, it so happens, are tied very closely to the physical properties of the frequency used. The major two trade-offs are range in open areas, and the ability to penetrate obstacles, such as buildings or terrain features like a hill or forest. While researching the different open RF bands, we determined that at a frequency as low as 2.4ghz(a common WiFi band), we would not achieve a range that was acceptable, even in an open space.

Another factor was the bitrate for the devices. As you lower the frequency of the transmission, the data bitrate drops accordingly. Because of this, we needed to ensure that the technology we chose would fit within our bit-rate requirements. The overall packet size we determined to be in the range of a few Kb. The data we intended to send on our devices was fairly small, consisting of a string of acsii characters and some provisions for transmission acknowledgements.

Based on these considerations we arrived on two frequencies that seemed to fit our requirements well. These two RF bands were the 435MHz and 915MHz bands. These two bands had a number of pros and cons for each. Ultimately, either of these bands would have worked for our project, however we still continued to determine the best option for our project.

For the 435MHz band, the biggest advantages were a slightly higher theoretical range, and a wider RF bad to work with. This band allows amateur use from 420MHz all the way up to 450MHz. The advantage of a wider frequency range is that it allows more concurrent users or devices in the same area while avoiding conflicts. There are ways to mitigate conflicting signals, however they are out of the scope of this project. Due to this, any conflicts would need to be dealt with either by utilizing the RF band given, or relying on built in technologies in the product that we choose. As far as the increased range for the 435MHz band, we determined that this was largely inconsequential, as both frequencies offered plenty of range.

The disadvantages of the 435MHz band can be summed up mainly by the lower obstacle penetration ability, as well as a significantly smaller pool of brands and products to choose from. Because the frequency range we are looking at is very low for obstacle penetration, it can be greatly affected by the slight change in frequency from 915MHz.

When Considering the 915MHz band, there were several advantages. There were a number of product lines that seemed to support what we intended to do with this project. The maximum range for this technology was well above the range specified for our project, and the cost was fairly attractive as well.

While the range would vary based on the actual product that we selected, the devices in this Frequency band seemed to all have ranges that either exceeded our requirements, or greatly exceeded them. Not only did this mean that selecting a product would be much simpler, it also meant that when it came to power usage, we could lower the power usage, which in turn would lower the maximum range, while staying inside our design specifications. This gave us a great amount of flexibility for later stages of the project. This flexibility was something that we valued greatly.

The 915MHz band also featured slightly better obstacle penetration than that of the 435MHz band. Because of this increase, we could also expect our finished device to function with the specified requirements, even in a worst-case scenario such as a busy urban area with greater amounts of interference and more obstacles.

One of the few disadvantages that we saw with the 915MHz band was that it did have a slightly smaller usable bandwidth. This band is approved for amateur use from 902MHz to 928MHz. This is significantly smaller than the 435MHz band. One more consideration is that the wider range of products available may suggest that this band is more popular. While this is not necessarily a bad thing, it does mean that there is a higher risk of running into interference from other devices using the same frequency band.

Ultimately, we did decide on using the 915MHz band, as this was the band that most closely fit our design requirements. Upon selecting this band, while still defining the product, we felt that it was necessary to add the frequency band to the requirements specifications. This was not simply because we had made the design decision to use this band, but also due the regulatory restrictions that using RF bands entails, as well as the fact that other parts, such as the antenna, would specifically relate to this frequency.

After selecting the 915 MHz band additional research was done to verify that our use was appropriate.

3.2.2.2 Product Brand

When selecting the product brand, we needed to repeat many of the steps from the first stage of the selection process. We once again referred to the requirements specifications and used these to create a prioritized list of features that we sought from the device that we chose. Because all of the devices that we would look at from this point on used the same frequency, there were some similarities in many features. This meant that the priority among features was significantly different.

The biggest variable in the differing products that we looked at was the software, as well as the hardware interface. These features are largely unhindered by the frequency used, therefore this is where we saw the largest variations.

On the software side of things, we wanted to find a device that would support the use of mesh networking, this would not only help with communications from one device to another, it would also allow the use of multiple devices to extend the overall range of our device. Generally, we found that there were two different types of networks used, A star type network, and the afore mentioned mesh network. Some products also featured peer to peer networks.
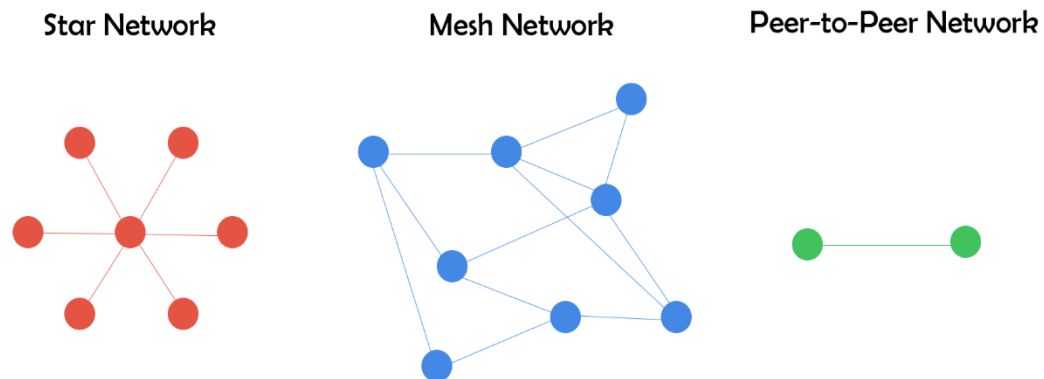

Figure 7: Star, Mesh, and Peer-to-Peer Network Examples

The following image shows a visual representation of typical configurations of star networks, mesh networks, and peer-to-peer networks.

Star networks operate off of a central server, or a number of servers, that connect end users. These networks make routing messages simple, however, there is minimal flexibility for mobile networks where all of the infrastructure is contained in the devices themselves. A good example of this is a cellular network. Your phone can connect the any cell tower within range (the "servers"). This works well because the towers are both high powered and stationary. Routing calls is easy because the networks servers stay in the same configuration. Our device is intended to function without the need for any stationary network of servers, meaning that it can be used in any area of the world, to some extent, with a minimum of 2 devices.

Digi XBee-PRO 900HP	900 MHz	Multipoint	Extended-range peer-to-peer mesh, sleeping routers	9 miles / 14.5 km	Through-Hole	XKB9-DMT-UHP (US/CA) XKB9-DMT-AHP (AU) XKB9-DMT-BHP (BR) XKB9-DMT-SHP (SGP)	10 Kbps or 200 Kbps	215 mA / 29 mA
Digi XBee SX		DigiMesh	20mW networking XBee module for mission critical applications	9 miles / 14 km		XK9X-DMS-0	250 Kbps	55 mA / 40 mA
Digi XBee-PRO SX			1-Watt networking XBee module for mission critical applications	65 miles / 105 km	Surface Mount			900 mA / 40 mA

Figure 8: A Selection of the Available Devices

Peer to peer networks were also something that we considered. A peer to peer network is very simple. One device connects to another directly. This method could be expanded upon to create a custom mesh network protocol; however, it is less than ideal. By selecting a peer to peer device, we would require ourselves to perform more development tasks.

The other main consideration while looking into these products was the method which we were going to interface with the MCU. While research was done on the technology to use, it was also determined that we would use a microcontroller from the Texas Instruments line of MSP430's. Because of this, we knew that the simplest method to implement the MCU connection would be through the UART connection.

Based on these determinations we proceeded to select two major contenders for our device selection. These two products were LoRa Technology chips, and the sub GHz line of XBee products from DIGI.

The LoRa Chips featured the all of the features that we needed however not all the features that we wanted. They were built for simple integration using UART transmissions. This meant that we would not need to write substantial code to translate the messages from the MCU to the transceiver unit. This chip also featured ranges up to 15km. This far exceeded the range required by our projects, such that we could be fairly certain that we would be capable of meeting or exceeding all requirements that we had created to that point.

The major drawback to this chip was that the chip only supported star type networks and peer to peer connections. The documentation for the chipsets in this product range was quite comprehensive however the major drawback of having to write our own code and store that code on the limited space supplied by the MCU meant that we needed to see if there were better alternatives.

That alternative was found in the XBee line of products. The Xbee products feature ranges going up to 105km. The data rates were anywhere from 10kb/s up to 250kb/s. This data rate was well within the needs for our device. Even in urban non-LOS applications, we saw that we might exceed the range requirement noted in our requirements specifications.

The software for this product was the most comprehensive that we had seen. Not only did XBee support configuration on the end device, it also had a development suite that allowed configuration of the chips through a USB development board with a pluggable socket. This meant that we could use the chips in a testing environment prior to installing them in the end device. We could also become more familiar with the commands used to control the chip and test the range before we fully developed the end product.

The software also contained discrete support for the mesh networking that we planned on possibly implementing. The inclusion of the networking protocol for mesh networking in the chipset itself meant that the software development could focus more on the core functionality that we required rather than getting caught up in the details of the backend.

This chipset also used the same UART communication standard that the pervious chipset considered. The only drawback that this line of products had, was that the prices seemed to be a bit higher than other chipsets. We determined that this price was mostly due to the much better documentation and addition features that were available. Additionally, there were a wide range of different versions for this product, which meant that we could select a device that more closely fit the use case for our project.

Because of these reasons, we decided that the XBee chipsets would be what we used and continued to select a specific chipset. At this point we also reached out to DIGI to establish a line of communication for product support and recommendations with implementation.

3.2.2.3 Chipset

During the final stage of the communications device selection we needed to select the variety of the XBee chip that best suited our needs. Disregarding the chips made for other frequencies, which mainly existed for other regions in the world, there were about a dozen options to choose from.

There were three main criteria that we used to compare the units. The first of these was price. The price for each unit varied. While the units that were more expensive certainly had more features, we did not want to pay for features that we would not use. All units, however, fit within the budget for our device. Because of this, preference was given to devices that offered better performance, so long as the improved performance would be perceived in the end product.

The next factor was the range of the units. Typically, the range was also closely tied with the power consumption. Units with a much greater range used a much greater amount of power. Considerations were made pertaining to the solar aspect of the device. The primary concern was that the device would consume more power than that which a portable solar panel could produce. Upon further investigation we found that a 2 watt panel could be found at a reasonable size and price. Based on this, we figured that a device that had no more than a 1 watt power consumption would fit our use cases. We wanted to be certain that the device could operate for a significant portion of the day, even without external power sources.

Based on the power limits, we determined that all of the units being considered were within the required specifications. This allowed us to consider other factors from a more exclusive point of view.

Another difference in the various units offered was the form factor. For our device to maintain its hand held and portable status, we needed to make sure that it would have a PCB that would fit inside such a package. There were only a handful of chips that were offered in a surface mount configuration. This was the configuration that we wanted as it allowed our PCB to maintain a low profile.

Of the surface mount chips there were enough to choose from that we felt that we could achieve want we wanted from that selection. One particularly interesting option was a chip that included an MCU on the board. While we had already

selected another type of MCU, we could not avoid considering this product as it could make the development much simpler.While researching the unit further, we determined that our I/O would require more pins than the unit supported however, and we continued on to the remaining options.

SPECIFICATIONS	Digi XBee® SX Module	Digi XBee-PRO® SX Module
PERFORMANCE		
FREQUENCY RANGE	ISM 902 to 928 MHz	ISM 902 to 928 MHz
TRANSMIT POWER (SOFTWARE SELECTABLE)	Up to 13 dBm	Up to 30 dBm*
CHANNELS	10 hopping sequences share 50 frequencies	10 hopping sequences share 50 frequencies
RF DATA RATE	Low data rate: 10 kb/s; Middle data rate: 110 kb/s; High data rate: 250 kb/s	
MAXIMUM DATA THROUGHPUT	High data rate: 120 kb/s	High data rate: 120 kb/s
AVAILABLE CHANNEL FREQUENCIES	Low and middle data rate: 101**; High data rate: 50	Low and middle data rate: 101**; High data rate: 50
RECEIVER SENSITIVITY	Low data rate: -113 dBm; Middle data rate: -106 dBm; High data rate: -103 dBm	
RECEIVER IF SELECTIVITY	Low data rate, +/- 250 kHz: 40 dB; Low data rate, +/- 500 kHz: 50 dB Middle data rate, +/- 250 kHz: 30 dB; Middle data rate, +/- 500 kHz: 40 dB High data rate, +/- 500 kHz: 30 dB; High data rate, +/- 1000 kHz: 45 dB	
RECEIVER RF SELECTIVITY	Below 900 MHz and above 930 MHz; > 50 dB	Below 900 MHz and above 930 MHz; > 50 dB
RURAL RANGE LINE OF SIGHT***	Low data rate: Up to 14.5 km (9 mi)*****	Low data rate: Up to 105 km (65 mi)*****
URBAN RANGE LINE OF SIGHT****	Low data rate: Up to 2.5 km (1.5 mi)*****	Low data rate: Up to 18 km (11 mi)*****
INDOOR RANGE*****	Low data rate: Up to 100 m (330 feet)	Low data rate: Up to 300 m (1,000 feet)

Figure 9: Digi XBee Specifications

The last two options we considered were the XBee SX and the XBee SX Pro. These two chips were essentially identical with the exception of the power ratings and maximum range. The Pro version features a range of over 100km while the base version only featured a range of up to 14.5km. The power difference between the two modules was explicitly in the transmit peak power, with the sleep power and the receive power being the exact same. The peak power usage for the base model was 20mW, while the Pro version used up to 1W. The difference between these two is quite dramatic. The team decided that it would be best if we could find a product that lied between the two, perhaps a .5 watt unit. It was determined in the end that such a product did not exist.

Given the fact that both products fit within the requirements, we had to make a preferential choice between the two. The difference in cost was very significant in this decision. The base version cost $30 per unit, while the Pro version cost $100. While both units met our needs, the Pro version exceeded those tremendously. Consideration was made for the worst-case scenario.

The base version seemed to be capable of meeting the requirements in the best-case scenario, however it didn't seem to be enough for the worst case scenario. To elaborate, the power usage was well within requirements, however the range was slightly worrisome. The urban range was below 1km, which meant that we would have to use the product in a rural location to meet our requirements.

In contrast the Pro version fit within the power specifications but was much closer to the maximum acceptable value. The range of the Pro version was, in fact, much greater than this project needed. The maximum range, even in urban settings was advertised as being greater than the 1 mile that is required in the requirements specifications.

POWER REQUIREMENTS			
SUPPLY VOLTAGE		2.4 to 3.6 VDC, 3.3 V typical	2.6 to 3.6 VDC, 3.3 V typical
RECEIVE CURRENT	VCC = 3.3 V	40 mA	40 mA
TRANSMIT CURRENT	VCC = 3.3 V	55 mA @ 13 dBm; 45 mA @ 10 dBm; 35 mA @ 0 dBm	900 mA @ 30 dBm; 640 mA @ 27 dBm; 330 mA @ 20 dBm
SLEEP CURRENT	VCC = 3.3 V	2.5 uA	2.5 uA

Figure 10: Digi XBee Power Requirements

The maximum range posed a question to our team. This range was listed at 105km or 65 miles. This range requires a line of sight connection. The problem that we realized is that the curve of the earth at such ranges means that in almost all cases, such a connection is impossible. However, with a height of 2-3 meters off the ground a distance of 5-10 miles can be achieved. We used these distances as the basis to determine which of the chipsets to use. While the very long ranges would be nice, we figured that the use cases for those would be quite limited and therefore decided to base our project on the lower values.

The next three paragraphs describe in detail two solutions that were considered to circumvent these issues to increase the maximum range of the device. Ultimately, we decided that these solutions di not offer enough expanded capability to be worth implementing. Because of this, you will not see them in the final design. A great amount of effort went into researching the viability of these options. And because of this, they have been documented below.

One Solution that was considered was to use stationary base stations. These base stations would have to be either at geographic locations that allowed a long-distance connection (such as a hilltop or mountain peak) or would require a tower to elevate them very high up. The issue that we had with this solution was that it was either very restrictive, only allowing users to operate in mountainous locations, or it would require substantial infrastructure to create the towers required to achieve these ranges. This would go against the underlying principal of the project. We are seeking to creating a portable peer to peer or mesh network that allows the users to communicate without an outside network.

The second solution we considered was a module that would allow us to elevate the device, or a portion of the device to a high altitude, allowing it to send a message. The first solution for this was an arrow or slingshot device. This would be used in emergency situations and would increase the range greatly when fired into the air. The issues with this idea were considerable, however. Shooting the device, or a portion of the device with a preloaded message was deemed too risky.

Not only would such an action result in a high risk of losing the device, but another concern was that message transmission would be unreliable. We did not feel comfortable creating a system that is intend for emergencies and is also unreliable.

The final option we considered, related to the previous option was a balloon that could be deployed from a portable pack. This balloon would need to be lighter than air, portable, and safe as well as affordable. Considering all options available on the market, there was no option that fit these requirements. Because of this, we decided to forego this option.

In the end, we decided on the Pro version, with testing of the base version. One thing to consider with this version was that we could dial back the power setting to reach a better medium for power consumption and range. The power setting could be adjusted based on the use case, allowing the device to retain the benefits of both chips.

Alternatively, we did not want to exclude that 20mW chip, as it was priced very well. To ensure that we got the best final product, we chose to test both options. The current plan is to implement two units using the 1W chips, and one with the 20mw chip. This will allow us to test the maximum range for the 1W chips, as well as get some metrics for the 20mW.

In theory, we could produce both versions for different consumers. With a cheaper communications chip we can produce a cheaper end product. By offering this with a different version that has a higher range, the consumer could choose the product that works best.

3.2.3 Input and Output

When researching into the various keyboards that we would potentially use for our design, a few specific approaches and relevant technologies were considered. These approaches can be found below. The purpose behind keyboard interfacing is so that a proper input and output mechanism to the device can be devised. This will directly become our main functionality for the input and output of our project. Typing the characters will then be acknowledged by the MCU, which will then enable an output to the display screen. The main input interfacing will potentially be either a membrane or t9 configured keyboard or cursor input. For the display, we've opted for possibilities pertaining to LCD, OLED, and touchscreens. Concerning the status indicators, we're contemplating the use of RBG light diodes to convey the position of the message being sent.

3.2.3.1 Text Input

For our text input, there are a multitude of technologies we have looked into. The first of which involves using cursors as our form of input. The overall functionality of this input involves the use of three buttons, a left cursor, a right cursor, space, and enter. These buttons would act as shifters in the placement of the on-screen display cursor. Programming would be implemented to the microcontroller such that the letter on the screen with start at, 'A', and the shift cursors will change the letters accordingly. As expected, the space button will create a spacing for the cursor. Whilst using these buttons, once you've gone through and found your desired character, pressing enter will keep this ASCII code, and shift to the next cursor location. Considering the minimal number of push buttons being used, interfacing these directly with the GPIO pins of the MCU would be an optimal approach. One major disadvantage towards this input style, however, is that the feasibility aspect. The time taken to type out a simple message using the cursor method would be considerably longer than a using a completely functional integrated keyboard. However, utilizing the cursor would allow for a less complex program architecture whilst also needed less hardware pins.

Another of these input methods is integrating a functional keyboard. The process to this, although tedious, is unique. Typically, most keyboards are in the form of a two-dimensional array, usually consisting of rows and columns. SPST momentary switches are then soldered together to form multiple of these rows and columns. Take for example a 4x4 matrix, starting from row zero to row three, and column zero to column three. The columns are tied to a supply voltage, VCC, which sets all columns to active high. When the momentary push-button is pressed, this signifies a ground connection, which will set the column to active low. A program is written for the microcontroller, which will sequentially ground the rows in an individual manner, going from top to bottom, or bottom to top. When executed, the program will check each row that has been grounded, and then each column

pertaining to this row. If the button pressed does not lie on this row, then the column's will be read as all active high. It is with this algorithmic process that the momentary button that is pressed can be read, and thus will be acknowledged and an output can be displayed onto the LED. It is understood that while this method of identification of key-presses is used, the process may vary. It is seen that a look-up table for the programming may be seen as an optimal choice regarding the recognition of key presses. This approach will use more GPIO pins and require a more complex program to code. If accomplished, however, it will serve as a higher quality input mechanism.

One last choice for input is speech-to-text. This is a rather complicated approach, and requires a firm understanding into analog and digital communications. Furthermore, coding will prove to be a intricate task as a program will need to be written for the conversion of speech to text. This would however let a user provide vocalization as a form of input from conversion of audio signals to digital output. Still, it is important to weigh in the difficulties associated with such a task, and it may not seem as the most viable option when considering the interfacing of a tangible keyboard, or cursors.

3.2.3.2 Display

The screen display for our project will output the ASCII code relating to the character input by the user. Two specific types of these screens are vastly used in most products, such as TVs and smartphones, which are OLED and LCD. Liquid crystal displays are known for using a backlight, a special type of illumination aimed at producing a visible image. The light sources that are used as this form of illumination can be from LEDs, electroluminescent panels, cold and hot cathode fluorescent lamps, etc. Unfortunately, using this as an illumination method will draw copious amounts of power. However, reducing the brightness of the screen will also cut back on power consumption a considerable amount. Organic LEDs on the other hand require a specific amount of brightness to keep contrast ratios at an adequate quality. Interfacing either display is usually done by two forms of communication, I2C and SPI. The first, I2C, is an inter-integrated circuit. It utilizes synchronous communications between a main host and a peripheral device that is connected to it, such as the screen display being connected to a microcontroller. This specific interface only uses two signal lines for data transfer. Serial peripheral interfacing, on the other hand, uses four signal lines and synchronous data communications. It differs from I2C in that it allows for higher frequency devices to be used, and the interfacing can be set as full duplex. These interface methods are discussed in further detail in the following sections. Although there are still various strengths and weaknesses between these two types of screen displays, the more important factors to consider for our project include power consumption, screen quality, size, and ease of implementation. All these factors are considered when choosing the hardware modules, which will be found in a later section.

3.2.3.3 Status LEDs

The use of LEDs is another priority we currently have for our device. The colors of these illuminating diodes will signify a color-coded meaning, such as yellow meaning idle transmission or green acknowledging the message has been sent and received. Interfacing with these components, specifically with most microcontrollers, is a relatively easy task. One such way is to connect the anode to the microcontroller port pin directly, and then connecting the cathode to a resistor in series. This is then connected to ground. The resistance is typically within the range of a few kilo-Ohms. Using this resistor will limit the pin current to a safe enough amperage such that no damage can be inflicted on these diodes. Once these connections are made, interfacing with the microcontroller will be needed to control these diodes. This will be accomplished by programming the MCU to control the transceiver's signals and configuring them such that the proper color of a diode will be displayed. We also intend to use our LEDs to display when power is turned on, as well as when our battery is charging.

3.2.3.4 Internal Serial Communications

It is important to understand about serial communications, as the topic is relevant to our project. There are various protocols and systems in which communication between two devices, particularly a master and slave device, can occur. Two famous communication methods include I2C and SPI. They are very common and standard as the means for serial communication. The features of these two interfacing methods vary. Depending on the situation, such a type of communication may prove to be more desirable than another based on these features. For instance, SPI contains a full-duplex protocol, whereas I2C is half-duplex. This means that for SPI, data communication can be read and written simultaneously, whereas I2C can only read or write in the same allotted time interval. Another difference includes that I2C can allow for multiple master and slave devices on a single bus, whereas SPI is only limited to one master. It is with these features that careful consideration must be taken for all protocols of serial communication, so that specifications are properly met.

3.2.3.5 I2C

The I2C protocol (inter-integrated circuit) is a specific system in which a multitude of "slave" digital chips can communicate to one or more "master" chips. Implementation of this type of communication requires only two wires. These two wires contain the SDA and SCL signals, which are the data and clock respectfully. The clock signal is generated by the most-current bus master, and the slave devices that are connected to it may control the master to limit data being sent. Unfortunately, a disadvantage to this type of protocol comes from the bus drivers. They are in a configuration known as "open-drain", which essentially allows for signal lines to be pulled low. The issue, however, is that the signal cannot be driven to a high state, and thus pull-up resistors must be used. An example of this can be seen in a schematic shown below of a typical communication system between a slave and master device.
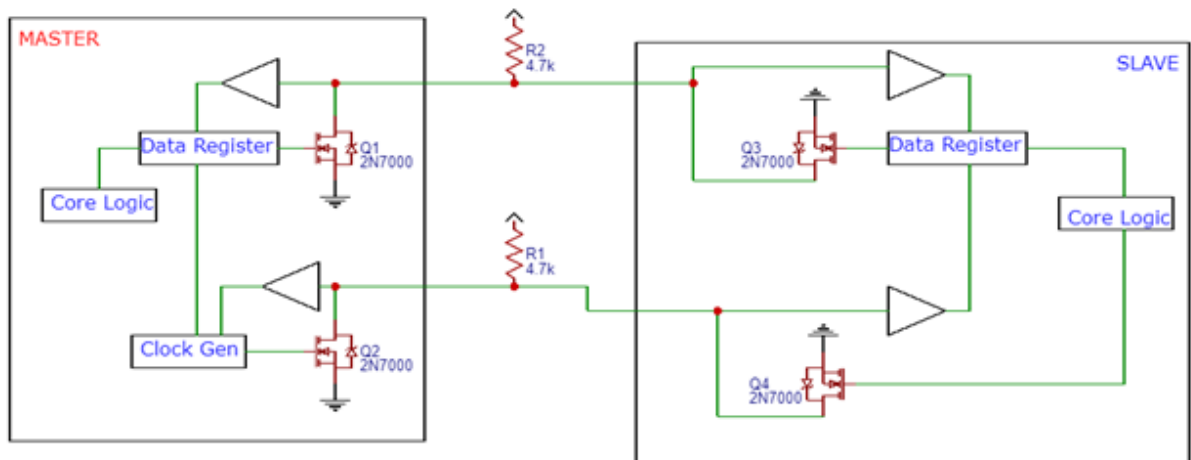

Figure 11: I2C Master-Slave Connection Schematic

From Figure 11, two 4.7 kilo-Ohm rated resistors are used on the data and clock signal lines. This effectively allows the device to restore to an active high setting when no device is asserting it to low.

3.2.3.6 Baud Rate

An important topic to discuss behind communication data is baud. This terminology appears countless times when mentioning the speed of transferring frames or packets. It has a unit measurement related to the number of symbols per second, usually related with bits or pulses. As an example, a baud rate of 4500 means that 4500 bits are transferred across a serial port per second. This rate effectively controls how fast bits can be sent; thus a very easy equation can be formulated. The equation below relates the time to the frequency (baud).

The frequency can simply be replaced by the baud rate, and the symbol duration time can be found. For instance, the time duration for a baud rate of 1000 is going to be 0.001 seconds, or precisely a millisecond.

3.2.3.7 TTL Logic

TTL, otherwise known as transistor-transistor logic, is a specific logic family pertaining to a type of integrated circuit used in various applications. Specifically, this IC was used in computers, electronics, synthesizers, microcontrollers etc. Fortunately, the voltage levels of TTL logic are within the range of a typical MCU's voltage supply range, from 0 to around 3.3-5V. The high voltage would signify an idle line, a bit value of one, or a stop bit. The low voltage would reference a start or data bit of value 0. TTL could then be utilized as a means of serial communication, and it had similar relations to how serial communication of RS-232 worked. Implementation of TTL logic of course consists of bipolar transistors that utilized their emitters as the inputs. For NAND inputs, these transistors are configured in a manner that the bases and collectors are fused together. The figure below shows the circuit schematic of a typical two-input TTL NAND gate.

3.2.3.8 SPI

Serial Peripheral Interface, known as SPI, acts as a bidirectional interface bus for communication purposes. Typically, this is used as a means of data transferring between a microcontroller and small electronic peripherals, such as miniature keyboards, SD cards, sensors, RAM, DACs and more. The interface is set to be a full-duplex master-slave that is synchronous. The term full-duplex means that the data communication between the master and slave devices can be simultaneous, and thus no delays are required. For instance, a phone call made between two users is an example of full-duplex, since both can speak and be heard at the same time.

From the figure above, we can see a configuration of the interface between a master and slave device, and particularly four signal wires. CS is our chip select, SCLK is the clock signal, MOSI is master out, slave in, and MISO is master in, slave out. It is noted that the clock signal is generated by the master, and the data being transmitted between the two devices is to by synchronized by this signal. This specific interface allows for a much higher frequency

3.2.3.9 Keyboard Interfacing

There are numerous types of keyboards that can be implemented as our input and output communications regarding the interfacing. Different types of ports can be used as means of interfacing with the microcontroller, such as PS2 (keyboard wedge), Serial (RS-232), and USB (universal serial bus). Although we have our sights set on a keyboard that is much smaller than that of the types listed above, it's important to list them for the sake of knowledge and backup planning. The list below covers research regarding the types of interfacing of keyboard ports and the specific architecture that each pertains to.

3.2.3.10 RS-232

Although outdated, serial communication is still used today, and its architecture is very well known. This standard interface was commonly used as a means of communication for low-speed serial data between relevant devices and computers. The interfacing of the RS-232 allows for asynchronous signaling. This means that the serial communication can support the transferring of data without the use of an external clock. Whilst this does allow for less wires used (I/O pins and common clk signal), it does require a more complex protocol system to ensure the stability and efficiency for accurately transferring data. The digital transmission process is set such that it is set to active high when idle (i.e. no communication is present), and active low for each bit. There is implementation of parity bits and one to two stop bits to help with error detection and correction. Figure 12: RS-232 Digital Signal Waveform Transmission of Data Packet helps display this incorporation of digital signaling for data packet transferring over serial communications.

The voltage levels for low has a range of +3 to +25 volts, whereas for high it is from -3 to -25 volts. Unfortunately, due to these ranges, they vary significantly from the TTL logic voltage levels. This means that they are not directly compatible when interfacing between the two standard protocols. A type of integrated circuit must be used to help make the voltage levels between these common, which is an absolute necessity in order to ensure that the data is not rendered corrupt.
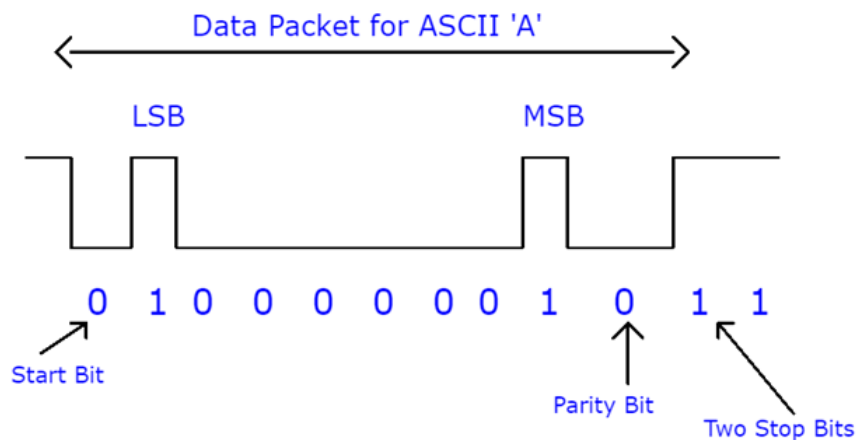

Figure 12: RS-232 Digital Signal Waveform Transmission of Data Packet, created in EasyEDA

3.2.3.11 PS/2

PS/2 is another variation of a keyboard port used in I/O for computers and microcontrollers, although it contains a historically outdated electrical interface. The protocol behind this is quite similar to that of its rival the RS-232. It provides a bidirectional synchronous serial communication between a host and device. The fact that this specific protocol is synchronous, however, is the largest dissimilarity in comparison with RS-232. This means that a clock signal must be generated, and that the device's data must be stable on the edges of the signal pulses to communicate in a timely and efficient manner. When the host chooses to communicate, a type of acknowledgement scheme must be made. The clock signal must be pulled low by the host, and then the data line as well so they're simultaneously active low, and then releasing the clock line. This will effectively signal the device of interest to generate a clock pulse signal to start the dispatching of data. Like the RS-232, a data packet is transmitted such that there is a start bit (always set to active low), eight total data bits, parity and stop bits as well as an acknowledgement bit. The host-to-device communication can be seen below in the diagram.

3.2.3.12 Hardware Selection

In this section, we will be discussing about the selection of hardware that will be used for the design stages of our project. This will consist of the components entirely pertaining to the hardware structure. Mainly, this will include selection of the microcontroller, transceiver, input device, screen display, type of LEDs, battery unit, and solar panels. We plan to begin our design stage once all the hardware components are chosen. Once the design stage is complete, testing for a prototype of the device will be conducted. Any potential issues concerning our chosen hardware parts will be identified during this testing period. If a hardware component becomes problematic to the infrastructure of the device, considerations will be made to select new hardware. This is under the assumption that selecting new harder will provide a solution greater than that of fixing the current problem with the previous module.

3.2.3.13 BlackBerry Q10 Input

Concerning the input aspect to our device, it was agreed upon by our team to use a miniature keyboard. The purpose of using this as our input is to allow our users the capability of inputting characters at a reasonable rate. After some research, a product stood out to us, most notably for its keyboard; the BlackBerry Q10. At first glance, the board appears to have an ergonomically appreciable look and feel and is just around the right size for use as our main input. The width is approximately 2.9 inches. A few drawbacks, although none seemingly the cause of being an ice-breaker, are noticed. The first being that this keyboard has an architecture much more aligned to modern-day designs, which serves to be more complex. To elaborate off of, a picture of the keyboard can be seen below.

Figure 13: BlackBerry Q10 [13]

The ribbon seen in Figure 13: BlackBerry Q10 makes interfacing with the Q10 a slightly awkward process. This may result in having to 3D-print a special case that will comply with the architecture of the keyboard. Furthermore, the ribbon requires a special connector to be properly interfaced. With these drawbacks aside however, and after determining the pinouts, the interfacing process with the MCU should be trouble-free. Once this process is complete, the functionality should prove to be a success.

3.2.3.14 LCD Display

The display screen we will incorporate into our transceiver device is a liquid crystal display, "EA DIP162-DN3LW". Our group collectively made the decision to use a display screen, as the overall specifications meet our constraints. The module we chose regarding the display appeared to contain the most key benefits when comparing to other relevant screens. For instance, we opted for technical specifications such as the physical dimensions, which would be approximately the same size as the keyboard's width. We also wanted to make sure that the

operating voltage of any display screen would be within the range of the voltage supplied by our power supply. One main benefit to the module we selected pertains to its pin-accessibility. Most screen displays contain special ribbons with the pin-outs, but the crucial disadvantage with this is that a special ribbon cable adapter is necessary. This adapter would breakout the pins to a more easily-accessible layout. The LCD our group chose specifically nullifies the need for a special adapter, as the display does not use a ribbon cable for the pins and instead are directly accessible. Having access to these pins directly will make interfacing processes a much easier task and will ultimately save time and effort which may be dedicated toward more complicated sections regarding the hardware.

3.2.3.15 Status LEDs

Another main hardware component we will plan on using is some basic status indicators. Our group showed an interest into illuminated indicator(s); thus we chose to use typical RGB light emitting diodes. As of now, we will be using these standard LED bulbs as a means of prototyping with our microcontroller. As previously discussed, a simple setup can be used for this implementation. The anode part of the LED will run to the microcontroller port pin directly, and then the cathode will be tied to a resistor in series, which is then grounded off. This resistor will help effectively limit the current flow to the diode, protecting it of any damage that could be inflicted.

3.2.3.16 Cursor Input

An alternative to a keyboard interface was considered, which involves the use of arrow keys and cursor input in order to provide a means of communication between the user and the Long-Range Wireless Texting Device. In place of a full miniaturized QWERTY keyboard, this 'Cursor Style' would have five keys: up, down, left, and right arrow keys, as well as a selection key. Using the character display module, which shows all input and output relevant to the user on the device (E.g. incoming, outgoing, and ongoing transmissions), the user would be able to see a blinking text cursor on the screen. This would signify that a character or value would be able to be inputted on the screen in locations identified using the arrow keys and selected using the selection button. Once at a desired location for text input, the user would be able to press the up and down arrow keys to cycle between the upper- and lowercase letters of the alphabet, including the digits 0-9 as well as period and space characters. Pressing left or right would allow the user to maneuver across the screen in order to input text at different locations, select the option for text input, in addition to selecting various other features available and shown on the character display.

There are several known and common ways to display a text cursor in applications involving text input. Most known methods include a blinking vertical line, a blinking underscore, or a blinking filled in box. Since this text cursor would also be used outside of text input as a way to select features not involved in directly inputting a character or text, a vertical line may be difficult to implement in a way where all used would show a clear purpose. A blinking filled in box would also be difficult to implement when it comes to selecting features as it may be the case that the entire selection option will have to be shaded or incased in the blinking box. The most reasonable and clear choice for the text cursor would then be the blinking underscore, which would be able to signify that a character can be inputted in its location, or that a feature choice can be made in its location as well.

Should the 'Cursor Style' be chosen, most of the efforts would be made on the software side, as the cycling of characters and result of using the arrow and selection keys would be made known by the microcontroller. The microcontroller would also have to interface with the character display module I order to should the features of the device, as well as to display and take in text that has been received and text that will be transmitted.

3.2.4 Power Supply and Layout

This device will, as all electronic (as opposed to electrical) devices do, require DC power to operate. The means by which this will be supplied will be discussed at length over the course of this section; both the means of charging as well as storage and supply for re-use will be discussed. It is fortunate for the purposes aimed at here that there now exists in the marketplace devices for load-balancing and charge control.

3.2.4.1 Battery

Before discussing the means by which the stored energy for this device will be brought into or used within the device, the most logical place to begin is a discussion of the means by which this energy will be stored- especially since this will be a major basis by which many other things will be determined. As such, let us discuss now the various options for batteries, as well as why the particular type that has been chosen for this project has been selected.

First, let us consider the requirements of the battery: it must have a relatively large energy capacity so as to allow extended use between recharges in the wilderness; it must be able to operate over the course of a relatively extended number of recharge cycles so as to extend the life of this device that will not cost an inconsequential sum; and it must be sufficiently rugged such that the battery can handle the shocks, drops, and vibrations that such a device meant for communication in off-grid situations will experience. And although it has not been mentioned thus far it is worth noting as a not-insignificant battery requirement

specification – that is, *rechargeability*. It is surely altogether desirable that in a device that is both expensive and meant to have a relatively long service life that the ability to recharge the batteries contained therein – be they removable or not – is surely something that would add a level of convenience and reliability to the device that is not in any small measure negligible. As such, it should be presently clear that the ability to recharge the on-board batteries of the device is of an importance that approaches paramount. To this end, several rechargeable battery solutions will be discussed in the following sections. Furthermore, for the sake of the convenience of the user, it should be considered desirable that the recharging of the batteries of the device should not require the removal of the batteries from the device; indeed, while this limits the user of the device to a regimen of recharge through whatever facilities for energy input exist on the device, it has been determined here that the inconvenience associated with this has been all but precluded from resentment on the part of consumers in the contemporary marketplace on account of cellular phones creating within the minds of consumers a sense of normalcy with being unable to hot-swap batteries within their devices; as such, a battery that is both rechargeable and fixed within the case of the device shall be sought-after as the energy storage solution of this project, and several such apparatuses for these ends shall be considered in turn in the immediately following sections.

3.2.4.1.1 NiMH

NiMH is the abbreviation for *Nickel-Metal-Hydride* a popular battery chemistry among general households and hobbyists alike. One of the early players to the rechargeable battery market, NiMH batteries grew to become a permanent fixture in the minds of many hobbyists, as well as those that simply grew weary of purchasing new batteries of an alkaline or other non-rechargeable battery chemistry for use in their household electronics with the regularity with which they had become accustomed. Their availability in a variety of standard sizes makes them convenient for replacement of batteries in household devices; in addition, their high capacities leave them with longer useable lives between recharges than, say, NiCad (Nickel Cadmium) cells. However, their low voltage, relatively short service life, and fast comparatively high self-discharge make them less than ideal for our purposes here; another battery chemistry will be sought.

3.2.4.1.2 Li-Ion

Before moving completely into this next technology, it is deemed that a brief justification of the dismissal of the other principal rechargeable battery chemistry mentioned briefly in the last section, the *NiCad* battery, is owed the reader; the low energy density coupled with the relatively strong memory of NiCad batteries were determined sufficiently crippling as to be disqualifying for the NiCad chemistry for

this device under consideration of the end uses of the final product; indeed, in spite of the ease of recharging as well as ruggedness and standard sizing presented by the NiCad offerings currently available in the marketplace, these hurdles are too significant to be ignored in the face of modern alternatives, as shall be seen.

Moving our consideration to the "elephant in the room" of rechargeable batteries that has in this age become the Lithium-Ion battery chemistry (and the some dozen variants thereof) - a development spurred on predominantly by the revolution in mobile devices that has taken place over the past decade or so – we see that there are several reasons these batteries are extremely desirable for our purposes, their downsides notwithstanding. We shall consider each in turn.

i.   Incredibly high energy density

The Li-Ion battery chemistry claims the highest energy density of all chemistries available to consumers (that is to say, those that are stable); this allows us to keep our device small and lightweight while still allowing the user to bring with them the greatest amount of energy possible.

ii.   Ability to charge and recharge quickly or slowly many times

As long as the input current is limited to within a specification given by the battery manufacturer, the batteries do not care whether they are charged or discharged partially or to their minimum threshold voltage, as long as they are charged from a Constant-Voltage/Constant-Current (CV and CC, respectively) power supply. This is desirable, since we are planning to have power inputs that are variable as well as use that will cause the battery to be drained in small, unpredictable bursts over the course of a charge.

iii.   No appreciable memory

If the battery is only recharged to, say, 80% of its maximum voltage before being drained down again, the maximum capacity is not only 80% of its original capacity, as can be seen in some other chemistries. Since we will be continuously charging and discharging the device while in the field, this is one of the most crucial characteristics for the battery to possess in order to see any sort of appreciable longevity in the device's ability to hold a charge.

iv. High current output capability

Many of the other chemistries get quite hot and become unstable when called upon to deliver currents in excess of a couple of hundred milliamps, whereas Li-Ion chemistries are able to easily achieve high current outputs on account of their naturally reactive chemistry based on Lithium, the lightest of the metals. As such, we can reduce the number of cells required to deliver our device's current needs, saving size, weight, and potentially cost.

Of course, as with all exercises in engineering, these advantages are not to be gained without some downsides as well. We shall consider each of these in turn as we have the advantages above.

i. Potentially dangerous

As was just discussed in point four of the advantages, Lithium is naturally one of the most reactive elements in nature – in fact, all of the elements in its column are never found in a pure form in nature on account of their high reactivity. As such, when exposed to air it will react violently with it, resulting in a large fire and/or explosion. Given that this device could reasonably be expected to be given a fair amount of shock and vibration over the course of its life (though not that dissimilar to that experienced by a cellular phone, however), it is desirable that the battery be as rigid as possible to avoid safety issues that could arise from abuse of the device; to this end, a Li-Ion cell in a hard casing has been determined to be preferable over a Lithium Polymer (Li-Po) battery.

ii. Ages poorly in the heat

Unfortunately, the Li-Ion chemistry does not enjoy nearly as long of a lifespan when exposed to heat for prolonged periods. Given that this device is meant to be used in wilderness-type outdoor conditions, the encountering of such conditions is all but a certainty for many of the potential users of the device. There is little that can be done to negate this besides keeping the temperature cooler during recharges through the use of a temperature sensor of some sort.

It is the opinion of this group that the listed advantages outweigh the disadvantages for this technology, and therefore it has been selected to serve as our energy storage medium of choice for this project.

3.2.4.2 Power Input

Though the medium in which it is intended that the device's energy will be stored has been discussed at some length above, it is still necessary to discuss how it is supposed that the energy will be brought into the device – that is to say, by what means input power will be brought into the device (as power is but the *time differential* of energy). It has been determined that given the environments in which the device is intended to be used, the ability to recharge the device in remote locales is invaluable. To this end, we will have multiple power inputs, as discussed below. A brief discussion of a couple of possible inputs has been included, as well as the reasons for their inclusion or exclusion from the device.

3.2.4.2.1 Thermoelectric Panel

The thermoelectric effect is the *direct* conversion of temperature gradients to electrical potential. The *Seebeck Effect* is the mechanism by which these devices work – the same effect that thermocouples use to generate voltage measurements that are used to measure temperature at a point. These devices can thus effectively turn temperature differences into voltage differences, generating electricity with naught but temperature gradients. Thus, in theory, one of these cells could be used to generate electricity anywhere in portable package.

They are not without significant limitations, however; the panels require not just high temperatures, but high temperature *differentials*, which is something hard to produce in the field outside of lighters, fires, or other such sources. As such, this was discarded as a practical source of energy in the field.

3.2.4.2.2 Photovoltaic (Solar) Panel

One of the best means for recharging a small device away from locations with electrical outlets available is the use of photovoltaic (PV) cells connected together in an array to form a panel. Photovoltaics get their name from the fact that they generate a voltage in the presence of light; this is done through the photons that constitute the light serving to create *electron-hole pairs* in the panel that become electrical current when passed out of the panel and into an electrical device or circuit. The voltage produced by each cell when exposed to a strong incident light is fixed by the chemistry of the PV cell; for silicon cells, which constitute the overwhelming majority of PV cells in the marketplace at the time of writing, this voltage is about half a volt (0.5 V). Thus, to get a voltage of, say, three volts, we simply string six of the cells in series, as voltages in series simply sum with one another. By this method, a solar cell of voltage $V = k/2$ for non-negative integral k may be easily constructed.

There are several advantages and disadvantages of solar panels for remotely charging devices; we shall consider them in turn.

Advantages:

i. **Reliable**

PV cells experience very little wear with extended use, and thus will last for a long time absent being subjected to mechanical or electrical stresses. Because the mechanism is a one of changing the energy state of electrons within atoms of a fixed position and elemental makeup rather than one relying upon a chemical reaction, there isn't an opportunity for unwanted products or be produced, or for reactions to fail to reach completion, among other things that can cause wear in devices that operate via some form of chemical mechanism.

ii. **Ease of use**

Note that this advantage is for the end-user of the device, not the designer – that will be discussed more below. Because all that is needed for the conversion of energy is exposure to light, all the user has to do is set the device in direct light (sunlight, usually) for it to work; there are no mechanisms to set up or things to plug in (if the device is integrated, which it is planned to be).

iii. **Small and light (portability)**

Because the panel can be contained within the device and embedded into one of its surfaces, the device is not forced to be separated from this as a power source, nor does the user have to extend or connect larger apparatuses that would take up room on the user's person. In addition, these panels are relatively lightweight, and so don't leave the device significantly heavier for their presence.

<u>Disadvantages:</u>

i. Costly

For a given power output capability, solar panels are comfortably on the expensive side. Of course, their ability to be used: anywhere with sunlight; for extended periods; without (time-of-use) waste products is what makes them so valuable. It has been decided that this cost, though high, is justified for the extra utility brought by the feature's addition.

ii. Brittle

PV panels don't tend to flex well (unless designed to do so), and will often crack or break if made to do as much. Even small cracks can often take the devices out of commission, and as such there is a certain amount of ruggedness that is lost in the employing of such devices. That said, the loss from damage to the panel is of the panel, not the device as a whole – as such, there will be no "collateral damage" from any mechanical failure of the PV panel regardless.

iii. Variable power output (input) and ineffective in some weather or locales

It is an inescapable reality that PV panels require photons to produce current, and as such need light to work. In addition, the current produced by the panel is proportional to the incident light, and as such will produce little output in conditions where there is not a strong light beam upon the panel. There exist many locales where it is simply either very rarely or never very bright out. As such, the PV panel will be effectively useless in such locations. There is little to be done about this – it is a weakness inherent to the technology. In addition to all of this, a variable output current makes the use of the device to charge a battery with a minimum input voltage set by an internal regulator challenging; a charge-controller device would be required to couple a solar panel to a battery and create of the combination an efficient charging mechanism. Nevertheless, the utility brought to many users that do operate in areas with enough sunlight to produce appreciable current is deemed sufficient to warrant the addition of the technology to the device.

3.2.4.2.3 Hand Crank

The most common method of producing electricity is the use of electromagnetic induction through Faraday's Law. When the magnetic flux through a loop of wire changes, an electromotive force (*emf*) is induced in the wire. Since flux is a function of field strength *and* orientation, we have the option of changing the orientation to produce an *emf*; changing the orientation rather than the strength is desirable, since the orientation can be changed by mechanical means. Such a setup constitutes that which is present power plants.

We could use a similar mechanism that contains a loop of wire and a magnet to produce electricity given that either the wire loop or the magnet are connected to shaft that we spin with a hand crank. Thus, we would have created a portable generator. As long as we can turn the crank, we can produce electricity.

Of course, there are a couple of downsides to this approach: first, the mechanism is not of a negligible size and weight; second, the naturally extremely variable output of such a system makes a challenge in the regulation of its output. For these reasons, it was narrowly decided not to include such a mechanism in our device.

3.2.4.2.4 Wall Wart Input

A means of powering devices that remains popular is the use of a small power supply, often with a transformer, that plugs into the wall – these are often called "wall warts." They have the ability to transmit an appreciable amount of power (into the tens of Watts or more), and are almost universally the means by which medium-sized electronics like laptops are charged. They can be found in a variety of sizes in both DC and AC variants, and are relatively inexpensive. Because many of them rely on a transformer, they often only have two power pins for AC power input, and are not polarized.

They do have some downsides, however: they are generally heavy, as transformers cannot be lightened but so much; in addition, they require a wall plug at 120V, and if the power provided is not of a high quality, such as that from a generator (common at campsites), the wart may pass power that will require additional conditioning by the recipient device's onboard PSU before it can be used. As such, the wall wart has been discounted for the purposes of this project.

3.2.4.2.5 USB Power Input

The Universal Serial Bus (USB) has become ubiquitous on consumer electronics: it is the go-to standard for the transmission of both data and power for applications where the levels of throughput of either are relatively limited in scope. As such, many consumers already own at least one USB cable and charging port, be it in a wall-plug form or on a larger device such as a computer. Thus, it has been deemed extremely desirable to have USB as a power input; this will likely serve as the main power input for the device.

The USB standard varies in its current-delivery capabilities depending on the generation of USB considered. USB 1.0 and 2.0 were specified capable of delivering 500mA, with USB 3.0 capable of 900mA. The voltage, however, has remained fixed at 5V; some quick math shows us then that the USB standard is capable of either 2.5 or 4.5W of power delivery, depending on generation.

Conveniently, the USB standard is backwardly-compatible; however, the ability to use USB connections of any generation can be hindered by the connector used, as there have been several over the standard's history. The connector opted for in

this study is the USB-C format – one that, while still fairly new at the time of writing, is convenient on account of its ruggedness, compatibility with Apple or Android phones, and ability to be used in either orientation. In addition, by using this standard we will give the user the ability to use the phone charger that they may already have with them in the field, as portable batteries for recharging cellular phones in remote location enjoy relative success in the marketplace, and especially-so among campers and the like. This will have the effect of limiting us to the USB 3.0 standard, giving us 900mA of current at 5V with which to work for the rest of our power supply design.

3.2.4.3 Power Regulation

3.2.4.3.1 Charge and Load Management

'Figure 14: I-V Curve for a solar panel Credit: Linear Technology' displays the current through a solar panel versus the voltage across it, along with power curves to show the point on the curve that yields that greatest power (the greatest product of current and voltage). We can see that as we increase the voltage across the panel, we decrease the current through it; as we decrease the voltage across the panel, we increase the current through it. This can also be viewed from the inverse direction: as we seek to pull more current from the panel, we will decrease the voltage across it; if we decrease the amount of current we're trying to draw, the voltage across it increases. Which curve from the family is the active curve is simply a function of intensity of the light incident on the panel at the time.

Because of this behavior, a problem that can arise when using solar panels with voltage regulators then is that the regulators have certain minimum voltages to turn on. If they attempt to draw more current than the panel can provide towards the higher voltage values, the voltage drops too low for the regulator to stay on. However, because the regulator turned off, the current draw has dropped, causing the voltage to come back up. This will cause the regulator to switch back on, and thus the current draw to come back along with it.
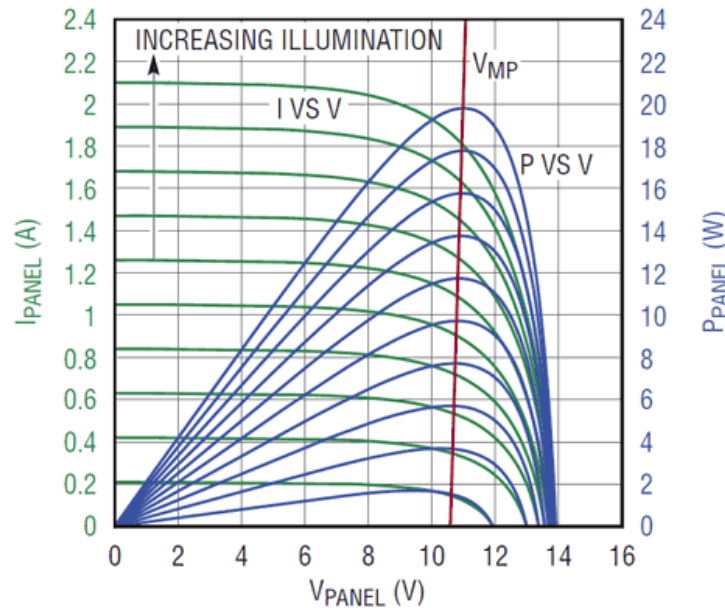
Figure 24: I-V Curve for a solar panel Credit: Linear Technology

Clearly this creates a cycle where the device is constantly switching between on and off, resulting in terrible efficiency for charging a battery and a totally unusable setup for any type of computer system. Certainly then we need some sort of additional system for keeping the current draw limited so as not to draw the voltage down too low to allow charging of the battery; of course, this still leaves questions of how to deal with the systems that require constant power to operate. While it is desired that some current would go to the battery to charge it, current must also be supplied to the rest of the device for operation. Thus, there must be some type of system for dealing with this split – that is, making sure that there is enough current being fed to the actual system to run it, with any leftover current going to charge the battery. This sort of system is what we will refer to as a *load management* system. Ideally, this system would present to the solar cell a load that at all times leaves the cell operating at the maximum point on the *power curve* (simply the current curve times voltage: that is, I(V)*V); when such measures are implemented the circuit is said to be *Maximum Power Point Tracking* (MPPT), a scheme that maximizes power extracted from the cell at all times.

In addition, it must be borne in mind that the battery is not just to sink current, but to source it as well. As such, a means of allowing current from the battery to supplement or take the place of current from an input source is obviously a requirement for this power system – this particular system is sometimes called a *charge management* system, as it manages the input and output of charge on the battery.

Lastly, a point should be made about fact that battery management has a unique issue that other power systems often do not: namely, the need to switch between

constant-current (CC) and constant-voltage (CV) operation. When a battery has been significantly drained and is first charging, it is important that the battery be fed by a CC source initially so as to prevent too large of an inrush of current from flowing into the battery. Sometimes excessive charging currents can be prevented by protection circuitry in the battery pack, but not all packs include this functionality; in addition, to engineer around the regular use of a last-resort safety mechanism is unequivocally irresponsible and unethical engineering practice. As such, the excessive current that would occur should a CV source (such as a standard DC-DC converter, be it linear or switching) be connected to a battery that has been discharged sufficiently so as to decrease the cell's voltage from that of its fully-charged state needs to be limited and regulated by a CC supply; after allowing the cell's voltage to increase back to its maximum point, the battery charger IC can then switch to CV operation, where it will remain until current into the battery reaches approximately zero. This is the only way to ensure that the battery is both fully-charged and that this has been done both safely and efficiently. Note that especially high-quality battery charger ICs will also include the ability to feed a thermistor back into them so that real-time temperature data on the battery can be used to determine if the current being fed into the device is causing excessive heating and should be dialed back (not to even speak of chargers that accept serial control inputs).

Voltage Conversion

Upon successfully implementing battery charging circuitry, the internal components of the device such as the display and the MCU will need specific DC voltages to run. Since the energy in the device is already stored in a battery, some form of DC-DC voltage conversion and regulation will be required. Provisions for this could be made through the use of several methods of varying complexity: Zener-diode regulators; voltage-dividers with capacitor banks; linear regulators; and switching regulators. It is the opinion of this team that while reduced cost and footprint are laudable goals in most any project (and especially-so with those that are mobile), this project in particular is found in need of both efficiency and stability: efficiency due to the need for extended device usage in remote locations; and stability due to the wildly different current draws that will be demanded by the transceiver of the device in sending and receiving modes, the former of which will be on the order of a magnitude larger than the latter.

Resistor and capacitor banks would require a tolerance of inefficiency that is simply unacceptable, not to even mention that the varying voltage levels would cause some of the systems such as the display that are more sensitive to particular voltage levels to malfunction; and while Zener regulators can do a decent job of holding the voltage constant within a small, defined range, the need to have a more stable and efficient solution does preclude their implementation here. This leaves linear and switching regulators; while linear regulators are small, inexpensive, and easy to employ, they suffer from dual prongs of inefficiency and an inability to increase voltage (higher output than input). In order to achieve either of these goals, charge pumping will be necessary – thus necessitating the use of switching devices.

Should the use of multiple, differing voltages be required, this can be achieved in a couple of different ways: first, a switching device could be used to create the highest-voltage rail, with the lower voltages coming stepping that voltage down with voltage-divider networks; second, separate DC-DC converters could be used to create each rail needed; third, some combination of these methods. The obvious advantages of the first method are its simplicity and frugality (including in footprint); the disadvantage being that the lower-voltage rails are all inefficient and current-limited – not to mention the heat generated by running a significant amount of the device's power through resistor networks. The advantages of the second method are the abilities to have increased efficiency, current, and even negative voltages; the disadvantages of course being a larger footprint and cost. The third method could thus be used to great effect in the case that the lower-voltage rails need very little current, making the inefficiencies of voltage-divider network immaterial. As will be developed later in this document, though this was the preferred method of the designers, the reality that the main current load of the device was the lower-voltage rail lead to the impracticality of methods one or three – leading to the use of multiple DC-DC converters, each of which was made switching to maximize efficiency, since the inefficiency of a battery-charger IC was already suffered.

3.3 Software

The following is the software design. This serves as a general reference for the software structure. All of the software is held on the MCU for this project. Despite this, however, an Object-oriented approach has been used to break down the programming into manageable chunks. The primary focus is on supporting the maximum performance from the hardware used. The secondary goal is to optimize the code to run as efficiently as possible, including using as little memory on the MCU as possible.
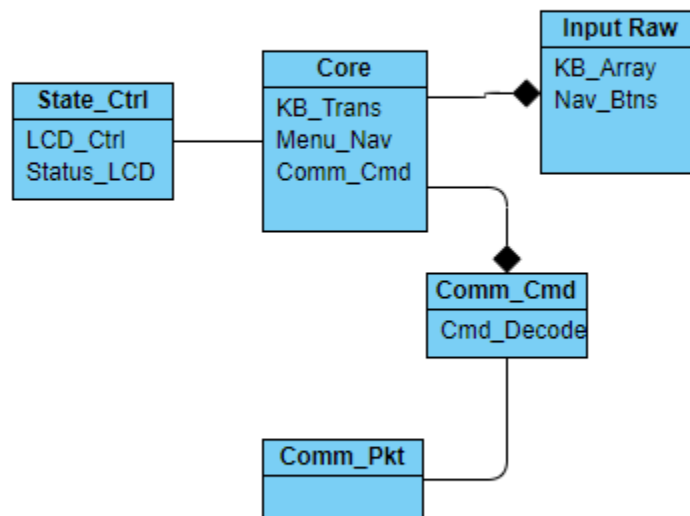
Figure 15: Software Class Diagram

3.3.1 Class Diagram

Emphasis was placed on not only breaking down the code into manageable segments, but also keeping the overall structure logical. The core reads input and sends output in it's raw sense. The Input class sends messages to the core that can be easily deciphered. Comm_Cmd sends the UART messages to the comm device. State control takes the input and shows the proper leds and also makes sure that the LCD shows needed information.

3.3.2 Development Environments

A development environment is a space that software developers can use to create, edit, test, and run software for use in their designs of choice. These environments are usually equipped with the basics that the developer would need to manage their design successfully. For the purposes of the Wilderness Information Link Device, a development environment will be needed in order to produce the software necessary to properly program the microcontroller of choice, the transceiver functionality, as well as the related peripherals that are mostly related to the user interface and ergonomics, such as the keyboard interface and status LEDs.

Several development environments were considered when designing the software portion of the Wilderness Information Link Device. When choosing which environment to use, special considerations were made towards the general usability of the environment, how technically supported each environment was, and which environment would best suite our needs for one based around embedded development using embedded C programming. The three main development environments that were considered when trying to decide which environment would be best to design and create the software involved in the Wilderness Information Link Device are Eclipse IDE, Atmel Studio, and Code Composer Studio.

Eclipse IDE: Eclipse IDE is a universal development environment that is capable of being used with just about every software development endeavor today. When it comes to embedded development, Eclipse offers a multitude of extensions that make it possible to program various types of microcontrollers and other peripherals in a way that is focused on efficiency. The Eclipse IDE offers a user-friendly experience tailored towards making the embedded development environment as organized and simple as possible. Although the appearance and usability of the IDE seemed to be the largest selling point, there were concerns about whether Eclipse would have the necessary extensions for the microcontroller of choice— more specifically, if Eclipse would be compatible with a microcontroller produced by Texas Instruments.

Atmel Studio: Viewing Table 11: Microcontrollers Under Consideration, it can be seen that two of the microcontrollers that were considered for use when designing the Wilderness Information Link Device are Atmel microcontrollers. This led to the question of whether Atmel's native development studio would be the best fit in programming the Atmel microcontroller, should one of the Atmel models be chosen. Atmel Studio allows the developer to create, edit, and debug code for Atmel microcontroller that is written in either C, C++, or assembly language. There is an integrated compiler, debugging capabilities, as well as the possibilities to view a simulation of the microcontroller model. Atmel Studio was mostly promising due to the in-depth support that was provided though the parent company, Microchip, as well as the offer of a extensive library and extensions that would make adding in additional features used for and with the microcontroller a simple and seamless experience. Should one of the Atmel microcontroller be chosen for use with the

Wilderness Information Link Device, Atmel Studio would be a clear choice for the development environment to be used.

Code Composer Studio: Code Composer Studio is a development environment created by Texas Instruments in order to support their line of microcontrollers, such as the microcontrollers available within the MSP430 series. Code Composer Studio was developed based on Eclipse—which is evident in the appearance of the environment. The program is organized with the purpose of being both efficient and user-friendly and supports in house hundreds of microcontrollers without the need of extensions. Much like Atmel Studio provides a thorough collection of documentation, libraries, and extensions that make using their microcontrollers with their development environment a seamless experience, TI's Code Composer Studio does the same for their line of microcontrollers. As most of the microcontrollers considered for use for the Wilderness Information Link Device were Texas Instruments made microcontrollers, Code Composer Studio would be the clear solution should a TI microcontroller be chosen for use. After deliberating on which microcontroller to use based on the parametric, cost, and power consumptions data seen in section 3.2.1.7 Microcontroller Comparisons, consequently resulting in the choice of the MSP432P401R, Code Composer Studio was made the designated development environment to be used to create, edit, and debug all software relevant to the Wilderness Information Link Device.

3.4 Budget

3.4.1 Account Management

To ensure that we maintained not only a professional atmosphere, but also maximum accountability, we sought a method to track our finances while also allowing the team to make purchases as needed. We considered a number of options to this end. One of these options was a pay as you go system, where we would try to balance the amount of money that each member spent, while making purchases with our own money. We decided that this was not ideal, and searched for a way to make the project more accountable.

One option was to set up a pool account. While this was simple, it did not allow some of the features that we wanted. So we sought an alternative that would feature more accountability. Ultimately we did not find the perfect option, but found one that worked well enough.

An online card service, Akimbo, offers a prepaid card service that gives not only virtual card services but also a physical card that does not need to be tied to an individual, but allows usage as a group account.

After deliberation, we determined that a reasonable budget for our project would be $1000. This is based not only on the parts required, but also the estimated contribution from each member. This amount was split equally amongst the group and then deposited in the pre-loaded account as the funds were deposited by the project members.

One downside to this method was that the funds were deposited into the bank account of one of the project members before they were deposited to the card. Direct deposits were desired.

3.4.2 Fund Allocation

Table 12: Budget is a table encompassing all planned and expected project expenditures as they relate to the three Long-Range Wireless Texting modules to be developed. The total cost for all three modules is estimated to be $584.47, with the cost for production of each individual module to be $194.82 when neglecting the cost of the Microcontroller Development Kit. The project is currently without a sponsor—therefore, all four members have agreed to personally provide funds during the duration of the project's lifetime. A shared fund will be created that will be used only for procurement and maintenance purposes. Should a team member commit to the acquisition of a part to be used in development, they will first have to append an entry to a log containing a record of all parts and materials procured using the shared fund. Both the log and the shared fund will encourage transparency between team members as well as provide a clear table of records to aid in regulation and status of funds.

Part Name	Quantity	Price (x1)	Price (All)
PCB	3	$ 50.00	$ 150.00
Electrical Components	AS REQ.	-	$ 15.00
Microcontroller	3	$ 10.00	$ 30.00
Microcontroller Development Kit	2	-	$ -
Solar Panel	3	$ 19.95	$ 59.85
Rechargeable Battery	9	$ 2.11	$ 18.99
XBEE SX 900MHZ RF Module Dev Kit	1	$ 199.00	$ 199.00
Character Display Module	3	$ 27.26	$ 81.78
Blackberry Q10 Keypad Keyboard	3	$ 9.95	$ 29.85
		Total Cost	$ 584.47
		Total Cost Per Board/Module	$ 194.82

Table 12: Budget

4 Design

The design of our project will consist of a thorough explanation into both the hardware and software aspects composing of it. This is considerably the most important part of the project, as the stages following this part of the document will necessitate the functionalities of the device. Our electrical section will consist primarily of the input and output modules, including the keyboard, display screen, and status indicators, as well as the power supply. A brief discussion into the transceiver will also be discussed. The software section will mainly consist of UML diagrams that go into depth concerning the logic behind specific programs. These programs will serve as the source of control for the hardware modules, namely the I/O and transceivers. Fortunately, in the preceding sections of this document, thorough research pertaining to the specific hardware modules and approaches to software programming. This has helped us envision better routes into our implementation of these modules and program approaches. Although the current researched modules are planning to be used, any complications in the testing phase may result in further research and a new component may be chosen. This may significantly change the design of the project, however, our team is confident that such drastic changes will not be necessary. In the following sections, detailed schematics, diagrams, and explanations of relevant hardware and software design will be properly introduced.

4.1 Hardware Design

Bringing about the hardware design, there are a few key points that must be addressed before the actual design and the process can be discussed. First and foremost, the software we intend to use for our schematics as well as PCB design is Easy EDA. As previously mentioned, this software application has an easily accessible interface with a large variety of libraries that consist of many specific hardware components. The footprints and packaging may also be found and edited within this software and importing and exporting of files from this application as well as other PCB developing software, namely Eagle and KiCad work interchangeably. A majority of all the schematics that will be seen below are produced by using Easy EDA, and the PCB design as well, however, some files used are imported from these other types of applications. The idea for the hardware design is to properly introduce each module of interest, whilst designing a complete schematic that connects all the modules to their respectable pinouts to the MCU. Once this schematic is completed and all the connections are properly made, the PCB will then begin to be designed. Once the PCB is successfully made, all the parts pertaining to it will be soldered accordingly and then the testing stages for the hardware can begin.

4.1.1 Keyboard

The implementation of an input is a necessity for this project, as previously discussed. The decision was made to use a keyboard-style input system. The BlackBerry Q10's keyboard seemed perfect for our project, and so thorough research is conducted pertaining to this product. Specifically, this keyboard consists of thirty-five keys, and includes a built-in microphone. Although the microphone is built in and can be powered, it will not be functional for the purposes of our project. The width of the keyboard is approximately 2.63 inches, which will be the driving dimension behind the actual size of the enclosed casing. As far as the ergonomics are concerned, the keys are of a dimension that allows for comfort when interacting with. A qwerty-styled keyboard is the configuration used, as opposed to the T9 style that some older models adopted. This component will offer user's a compact and comfortable feel for the typing, which is an important aspect to consider.

A modern ribbon-cable can be seen from the Q10. A special type of ribbon cable is seen, which makes finding a connector an imperative task in order to properly interface with the keyboard. After seeing the complexity involved with this, some research was necessary in order to find the right connector. After stumbling across some websites, where fellow hobbyists including "JoeN" from http://www.eevblog.com and Mr. Artur Pacholec from GitHub have very fortunately already figured out a lot behind the architecture of this module, including the connector required. Considering the pitch of the ribbon cable, a size of 0.4mm was determined, and thus a connector of this dimension was necessary. Twenty pins are to be used, and further research along with the help of these individuals gave rise to a integrated chip. This particular IC is a Hirose BM14B connector, and contains twenty-eight conductors, although eight of these are connected to ground. More importantly, it has the same pitch dimensions as the Q10's ribbon, and these twenty-eight conductors allow for more than enough pins to be connected and properly interfaced with. The ribbon cable from this keyboard will snap on to the connector. The IC will then utilize these conductors to map out the pins of the keys accordingly. Unfortunately, the dimension of this connector is exceedingly small, and thus requires PCB design for a proper adapter. As far as prototyping is concerned, it is necessary to design an adapter board for the connector to make access to the pinouts an easy task. For the entire device, however, we intend to utilize the layout of the board in a surface mounted manner such that the printed circuit board is fully integrated. This through-hole adapter board will essentially consist of a simplified circuit which expands the individual pins of the tiny hirose connector. Fortunately, the prototyping stage of the project will prove to be much easier than having to directly interface with the ribbon cable, and as such will alleviate some of the time deficiencies of our team. Figure 16: BBQ10 Schematic will show the pinout of this connector which we will be using later when assigning the connections made between the receptor and the microcontroller. It is

understood from the schematics made by Mr. Pacholec that power supply inputs are to be made for the MIC, and LED pins. Unless necessary for functionality purposes, we intend to use minimal voltage inputs. Although no specific datasheets can be required pertaining to the BlackBerry Q10 for copyright protection, a supply voltage must be determined. We will conduct a lab test to achieve the correct operating voltage and ideally the supply current that the Q10's keyboard will run at in conjunction with the power supply being used as well as any variable components. Considering the pinouts, an easier layout of the BBQ1O may be seen below. The figure is designed by Mr. Artur Pacholec. This schematic will be used in our completed hardware design. Although the pins are in a disorganized fashion due to the awkward architecture of the keyboard, the connector for the Q10 will help expand them and thus interfacing can be made directly to the MSP432. The number of columns seen below is five, and the number of rows seven. Although this architecture does not conform to the same number of rows and columns that common matrices consist of, implementation with code is still the same process.
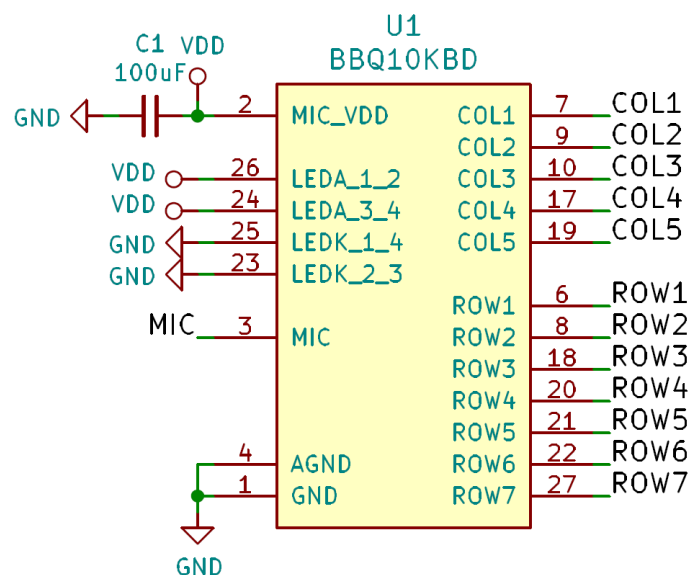

Figure 16: BBQ10 Schematic made by Mr. Artur Pacholec

Considering the receptable, it is noted that there is twenty-eight conductors available for use, although the Q10's plug only utilizes around half of this. The remaining pins are set to be grounded out. It is also seen that the numbering of the pinouts is disorganized, however, this is caused by either the connector or the plug, and must be tolerated with. Fortunately, as previously stated, only eighteen pins are being used, and the rest being grounded which allows for an adequate working space when prototyping with the PCB adapter board. Although the number of rows and columns aren't the same dimensions, this will not disrupt the logic of the

programming associated with it. It is understood that the normal keys are the set priority as the means for our ASCII char output to the display module. However, if in the allotted time we are provided that there is flexibility for further implementation, we will consider involving the special keys into our project. A special note is to be made concerning the mic. Our project will strictly conform to mechanical user input, and thus audio input, in this case the microphone, will not be relevant. With this in mind, the pin will be held null for the remainder of the schematics and designs shown below. The two tables concerning the keys associated with the rows and columns can be found below. The first table (Table 13) is a layout for normal keys used by the Q10 (pressed directly), and the second (Table 14) consists of the special keys (requires an extra button to alternatively select). These tables were created by Mr. Pacholec, and a link to his GitHub post may be found in the resources section of the paper.

Normal Keys	COL1	COL2	COL3	COL4	COL5
ROW1	Q	E	R	U	O
ROW2	W	S	G	H	L
ROW3	sym	D	T	Y	I
ROW4	A	P	R	Enter	Delete
ROW5	alt	X	V	B	$
ROW6	space	Z	C	N	M
ROW7	MIC	L	F	J	K

Table 13: Normal Key Layout by Mr. Pacholec

Special Keys	COL1	COL2	COL3	COL4	COL5
ROW1	#	2	3	_	+
ROW2	1	4	/	:	"
ROW3		5	(	)	-
ROW4	*	@			
ROW5		8	?	!	Audio
ROW6		7	9	,	.
ROW7	0		6	;	'

Table 14: Special Key Layout by Mr. Pacholec

As mentioned, the Hirose BM14B connector is a very small component, and thus interfacing and soldering with it is one of the main concerns. The SMT sockets as mentioned in the various threads on the internet mention the complexity it has when being interfaced with. The pins must be soldered to our printed circuit board, however, considering their relatively small size, proper tools and equipment must be used. One such solution to the soldering process could be the utilization of a toaster oven reflow in order to attach many of these pins to their respectful pads. This would then allow for the correct tracings to be adjoined, a crucial aspect of the hardware that will be necessary for proper composure and functionality. There are other aspects to the Hirose connector that make this an important component for the Q10 and the entirety of the device. This particular IC allows for a very nice and reliable contact with other chipsets. It also contains a high retention force for printed circuit boards. More important, it allows for prevention against solder wicking, a type of issue in which residual solder paste advances onto the lead of a component. This is a very useful feature, as the reflow oven option may leave these leads susceptible to wicking. Other features include dust protection, surface mounted style, shock absorption, and gold finish for the leads. A picture taken by JoeN of his special PCB adapter board, and the pinout of the receptacle may be seen below.
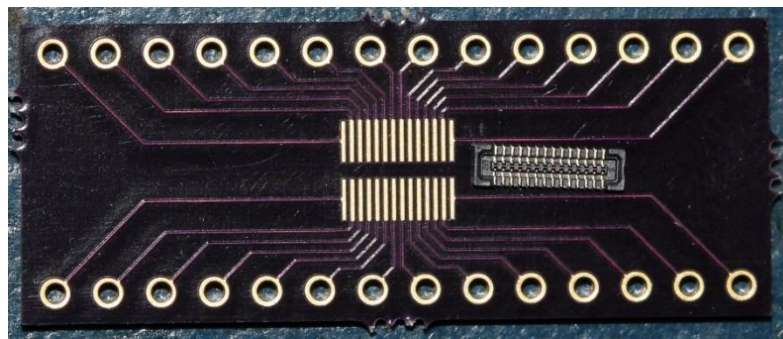

Figure 17: Ribbon Cable photo taken by JoeN

Pin 1: GND	**Pin 28:** GND
Pin 2: MIC_VDD	**Pin 27:** ROW7
Pin 3: MIC	**Pin 26:** LEDA 1&2
Pin 4: AGND	**Pin 25:** LEDK 1&4
Pin 5: GND	**Pin 24:** LEDA 3&4
Pin 6: ROW1	**Pin 23:** LEDK 2&3
Pin 7: COL1	**Pin 22:** ROW6
Pin 8: ROW2	**Pin 21:** ROW5
Pin 9: COL2	**Pin 20:** ROW4
Pin 10: COL3	**Pin 19:** COL5
Pin 11: GND	**Pin 18:** ROW3
Pin 12: GND	**Pin 17:** COL4
Pin 13: GND	**Pin 16:** GND
Pin 14: GND	**Pin 15:** GND

Table 15: Receptacle Pinout, found by Mr. Artur Pacholec

After proper interfacing with the ribbon cable and the receptor as well as the other relevant modules, prototyping will be conducted with the keyboard. This is to help ensure that functionality is present within the project. Unfortunately, the issue is persistent when concerning the dimensions of the receptor. A solution to this was created by hobbyist Mr. Pacholec which involves a simplistic PCB board. This board essentially is a large extension of the very tiny pins that translate the connection from the Q10's ribbon. This specifically will allow for breadboard testing, which offers a convenient method in quickly testing out the design aspects of the project. The through-hole design will require soldering of the individual pinouts, however, once accomplished the keyboard may be used in almost any applicable situation involving a microcontroller. The board below specifically makes use of labeling out the pins, as connecting the ribbon cable to the receptor will give a specific layout of the pins. It can be seen that the rows are accessible via the left hand the side of the PCB, and columns on the right-hand side. A ground and VDD pin are also seen, as well as the mic pin, although we have no plans to use the microphone of the input module.
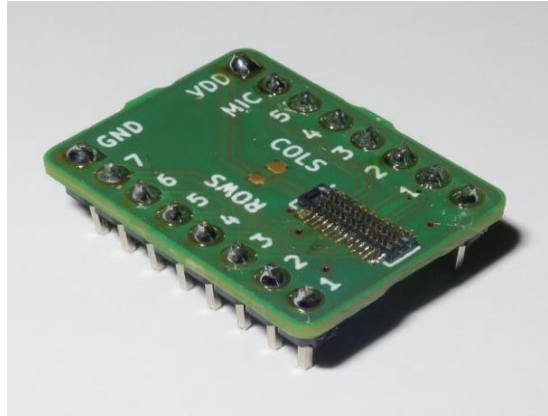
Figure 18: Adapter Board made by Mr. Pacholec

In order to better understand how a keyboard is properly interfaced, a schematic below helps convey the row x column matrix formed by our Q10. The supply voltage VCC is connected to the resistors to power on the ports of the rows and columns. One side of each switch is connected to a row and column, and the resistors are used to limit the current flow to prevent damage to the switches. An algorithm is implemented to check which specific key is pressed. The process starts by initially grounding the columns to active low when any momentary key is pressed. The program will then begin to ground the first row, depending on order of ascension or descension. It will then procedurally check each key pertaining to its respective row, and then provided the key is not acknowledged, will continue the process in a consecutive manner, and thus the columns will be read as active high. When executed, this algorithmic process will autonomously determine which button that the user has pressed, and once this data is read and acknowledged, the next part of the program will be carried out such that output can be displayed onto the LED. Although this process is a simplistic version of how keyboard key detection is applied, there are other programs and processes that may serve as a more functional or efficient way of identification.
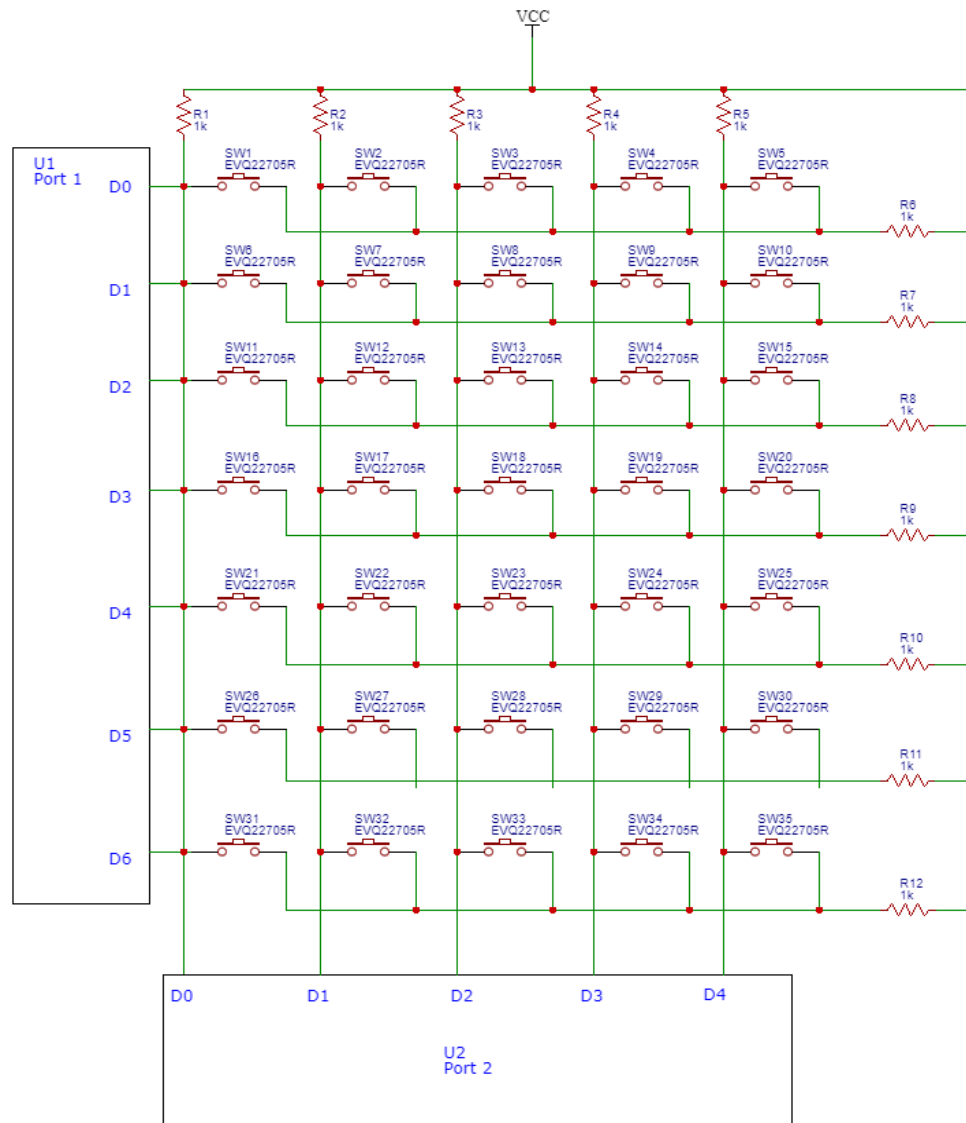
Figure 19: Keyboard Switch 7x5 matrix, made in Easy EDA

4.1.2 LCD

The second crucial component for our design is the liquid crystal display. A few types of displays were considered, however, including OLED, LCD, and LED. A decision was made to stick with a standard LCD screen that could be easily interfaced with our MCU. When searching for the right screen, a few specifications needed to be met. An operating voltage of 3.3 is necessary, as our power design is based on a supply of that scalar quantity. Also, appropriate dimensions in the length of the screen needed to be approximately the same as that of the Q10. This is so that the screen does not cause the case to be a size that would make typing an awkward experience. The ribbon cable was also a concern for our team. A modern-style cable is used with a variety of the LCD screens that were candidates for our display, however, there is complexity in finding the right connector for this cable. Fortunately, there are display screens that contain access to pins directly, and this specifically influenced our decision to opt for a output module. This module, " EA DIP162-DN3LW", fits the criteria previously mentioned, and will be our main display unless a better product can be found. The screen's data specifications may be seen below. The char x lines is 16 by 2, and consists of either a four or eight bit data bus. Operational supply voltages range for different types of modules, ranging from 2.7, 3.3, and 5 volts. The length of the module is 75mm, which is approximately 2.95 inches. This is very close to the Q10's length of 2.63 inches. Our module's display will allow for some moderate sizing of the case that will most likely diminish any discomfort in typing.

As far as interfacing with the module, there are nine pins on both sides of the board. A type of socket connector IC will be used for them. These pins will go to each individual socket, which will then convert to usable pins for prototyping. The hardware connections for these pins to our microcontroller are seen in the design schematic in the below section. To note, the U4 module below only contains eight pins, as we are neglecting the use of two particular pins from the datasheet.
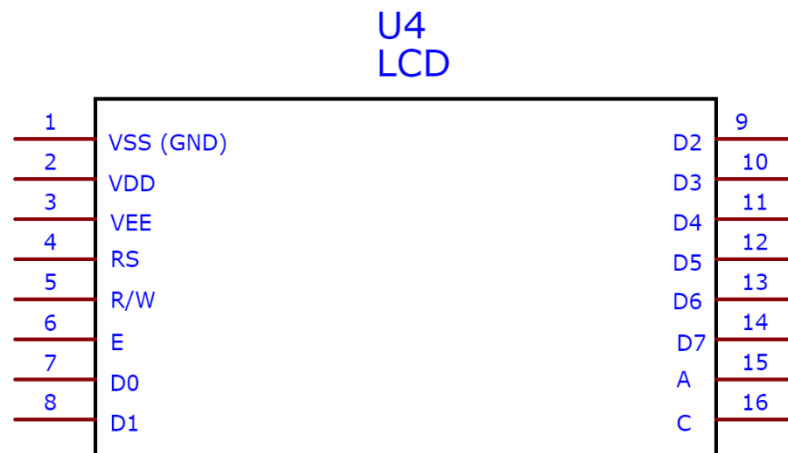

Figure 20: LCD Pins [22]

Character Count x Line:	16 x 2
Module Size - W x H x T:	75 mm x 27 mm x 11 mm
Backlight Type:	LED White
Interface Type:	4 bit, 8 bit Data Bus
Operating Supply Voltage:	2.7 V/3.3 V/5 V
Packaging:	Bulk
Operating Temperature Range:	- 20 C to + 70 C
Brand:	ELECTRONIC ASSEMBLY
Product Type:	LCD Character Displays
Factory Pack Quantity:	21
Subcategory:	Displays
Unit Weight:	0.846575 oz

Figure 21: LCD Parametrics [22]

Pinout

Pin	Symbol	Level	Function	Pin	Symbol	Level	Function
1	VSS	L	Power Supply 0V (GND)	10	D3	H / L	Display Data
2	VDD	H	Power Supply +5V	11	D4 (D0)	H / L	Display Data
3	VEE	-	Contrast adjust. (about 0V)	12	D5 (D1)	H / L	Display Data
4	RS	H / L	H=Command, L=Data	13	D6 (D2)	H / L	Display Data
5	R/W	H / L	H=Read, L=Write	14	D7 (D3)	H / L	Display Data, MSB
6	E	H	Enable (falling edge)	15	-	-	NC (see EA DIP122-5N)
7	D0	H / L	Display Data, LSB	16	-	-	NC (see EA DIP122-5N)
8	D1	H / L	Display Data	17	A	-	LED B/L+ Resistor required
9	D2	H / L	Display Data	18	C	-	LED B/L -

Figure 22: LCD Pin Descriptions [22]

4.1.3 Status Indicators

The status indicators of our project will consist of a few main signals. These will designate functionality of the project. For instance, an indicator may be a surface-mount LED that will acknowledge when the power to the device is turned on or off. There will also be RGB bulb-styled light emitting diodes that are implemented into the case. They will help the user acknowledge status updates of the message. For instance, yellow may indicate that the device is in an idle state, blue will signify that the message is transmitting, and green will acknowledge that the message has successfully been sent and received by both ends of the transceiver. Currently, we intend to use these bulb-LEDs for prototyping, however, in our PCB design we may consider the incorporation of surface mounted LED's for efficiency and to present a more professional aesthetic to the entirety of the project. For the configuration of the status indicators, we used the commonly seen set-up of a typical LED in series with a resistor, typically with a value of around a single kilohm, which is tied to VCC or VDD. The use of this resistor is to help limit the amperage such that the diode is not susceptible to damage. Once this configuration is properly set, the microcontroller will be programmed in accordance with the pinouts that are used by the emitting diodes. The program will use the logic necessary to control these indicators in the manner that we desire, such as with the color symbolism mentioned. A schematic for the design of three LED's may be seen below.
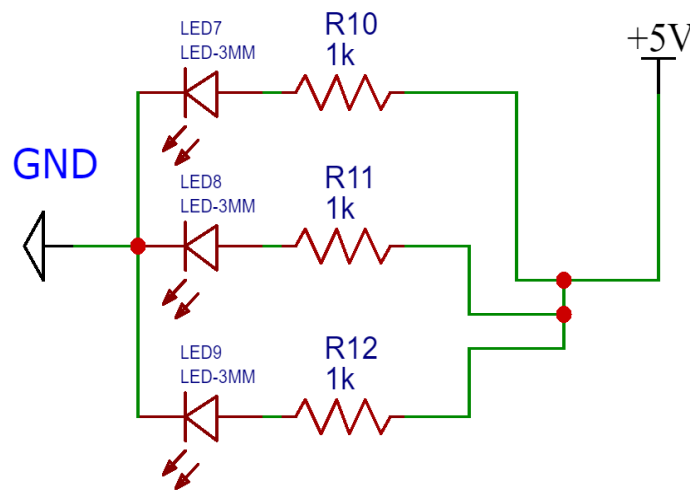

Figure 23: LED Schematic, made in EasyEDA

4.1.4 Power Supply Design

The following sections will delineate the design process that was undertaken for the PSU. The unit has been designed to accept input power from both USB-C and a solar cell; these inputs will then be used to charge a Li-Ion battery and from there the necessary voltages to operate the device will be provided. The design of each piece of the PSU will be discussed in turn.

4.1.4.1 USB-C Input Regulation

The use of separate input regulators has been chosen for this device, the principal reason for which is increased efficiency; the reason for this shall be elucidated. As was explained in the research section of this document, the battery ought to be charged via an IC specifically designed for charging batteries – something that offers increased efficiency, safety, and reliability. As discussed in the research section, solar cells become "bogged" when presented with an excessive current load; in order to prevent this, some sort of mechanism to track when the solar cell is reducing its power output is desirable. While this can be achieved through the use of PWM methods in switching devices, Texas Instruments produces a line of devices that uses *Dynamic Power Management* (DPM) technology to see the changes in current versus changes in voltage and adjust the load presented to the cell attached to the input accordingly. This allows us to achieve much of the efficiency of a switching regulator without the design and testing complexity associated with one. However, in order to use this functionality, the device must be allowed to "track" the battery; this would greatly limit the speed of charging that is possible with USB. As such, a decision has been made to create separate input regulation circuits for each device: this subsection shall discuss the USB input, with the solar input to follow.

Given all of this, a choice of battery charger IC had to be made for the USB-C input. The choice between a linear and switching device was made in favor of a linear device; the reasons for this being: small footprint; low cost; ease of design; and cognizance of the likelihood of the USB bus from which we are charging having an extremely large reserve of energy from which we can draw, making efficiency less important in this case. As such, the choice to use a common device for this application, the MCP73831 by Microchip, was made. Fortuitously, the typical application schematic given in the component datasheet is already set up to provide the current that is desired for our battery; note however that this is not surprising, seeing as it was already established above that most batteries on the market that are rated for the voltage at which the output of the battery charger is fixed (4.2 volts, the most common voltage for modern Li-Ion batteries) are rated for a maximum current input of 500 milliamps. That said, it should also be noted that the design for this part is extremely simple, with a single programming resistor

being the entirety of the design required to set the output current (again, the output voltage is fixed).

Compared with the reference design, however, it has been decided to increase the input and output capacitances to 10 μF each from their original 4.7 μF. The purpose of this change in values is to help accommodate the sharp changes in load through the system that will be experienced when switching between sending and receiving modes, as the power (and thus current) requirements of the transceiver change wildly between these two modes. Since the charge of a capacitor is

$$Q = CV$$

where C is the capacitance of the capacitor in Farads and V is the voltage of the capacitor in volts, and thus

$$\Delta V = \frac{\Delta Q}{C}$$

(noting C remains unaffected by the change because it is a function of geometry and materials) we can therefore say that

$$\Delta V = C^{-1}\frac{dQ}{dt}t$$

$$\Delta V = C^{-1}i(t)t.$$

As can be seen from the above relation, then, a given current drawing charge out of the capacitor will drop the voltage half as much for twice as large of a capacitor due to the linearity of the relationship between the two quantities.
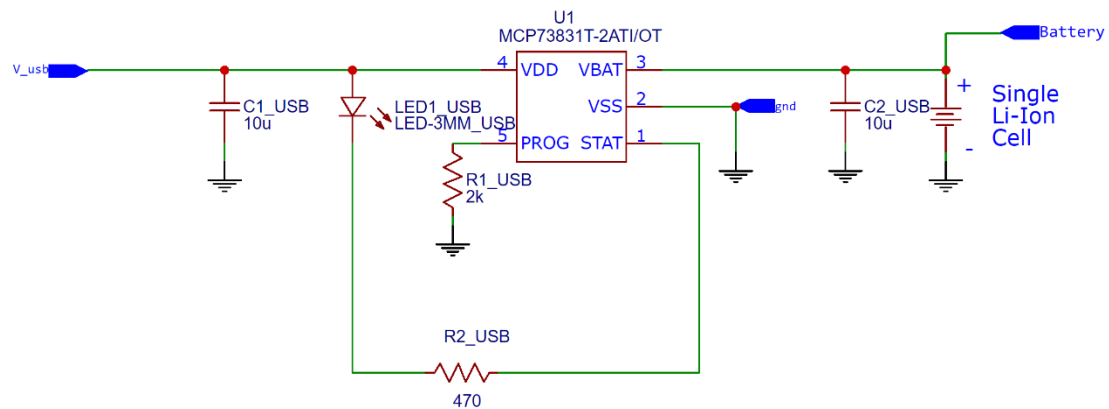

Figure 24: Li-Ion battery charger for USB input

It was determined that for ease of fabrication the SOT-23-5 package would be selected; there lies a concern in the heat output of the device in that package, and where it will go due to the lack of a thermal pad, but this is something that testing may demonstrate to be a nonissue. It should be recalled that linear devices such

as this one are inherently quite inefficient (relatively speaking, of course) – all of this inefficiency translates directly to heat output. However, power dissipated is equal to the difference between the input power and the output power; that is,

$$P_{dissipated} = P_{in} - P_{out}.$$

Since no device is completely efficient, the input power will always be greater than the output power, and thus this quantity will always be positive (mind that this is basically just a restatement of priors – if used as reasoning it is circular). We also know that in electrical systems the power in a device is simply equal to the product of the voltage across the device and the current through it.

Therefore, then, the ideal device that dissipated no power (and a zero-dissipation device would indeed be ideal) would be one in which powers at the two terminals are equal; that is,

$$P_{in} = P_{out}$$

$$I_{in}V_{in} = I_{out}V_{out}.$$

It is known that the current at the input terminal of linear regulating devices is very nearly that present at the output (for most devices in most operating regions); therefore, the difference in voltage between the two terminals will be the main defining factor of efficiency seen in the device. (Note that this analysis does not apply directly to switching devices because of the fact that not all of the energy difference between their terminals is dissipated at heat, as in linear devices – much of it is stored in reactive elements.) Since our input voltage of 5 volts is not massively greater than our maximum output voltage of 4.2 volts, the inefficiencies suffered at the hand of the regulator will not be massive, or at least for the CV (final) stage of the battery charging process. Thus, if the power dissipated is low, the heat generated should be reasonably low, and thus the selected package should suffice, even though it lacks a thermal pad.
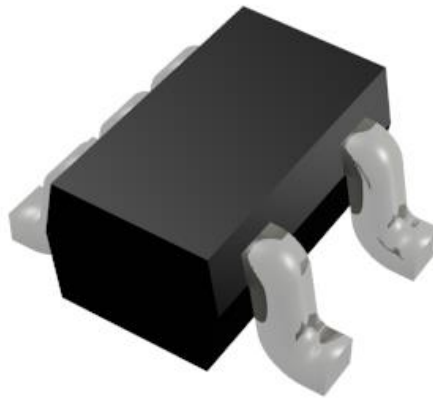
Figure 25: View of an SOT-23-5 package

4.1.4.2 Solar Input Regulation

As was discussed above, the solar regulator device that has been employed here – the TI BQ24210 - is one that combines (relative) efficiency and ease of design and testing; alas, while more sophisticated switching designs exist, they possess a level of complexity that has been deemed excessive for the scope of this project – especially in light of the quality performance that can be found in devices that are far more readily-usable. The design process for this circuit is laid out piecewise in the datasheet. Since the battery that has been selected does not have facilities for temperature monitoring through thermistors, the temperature sensor (TS) pin his been tied at 30% the voltage value of VSTB through the use of a voltage divide, as per the datasheet. The output current has also been limited at approximately 500 milliamps via the 787Ω resistor on the ISET pin.
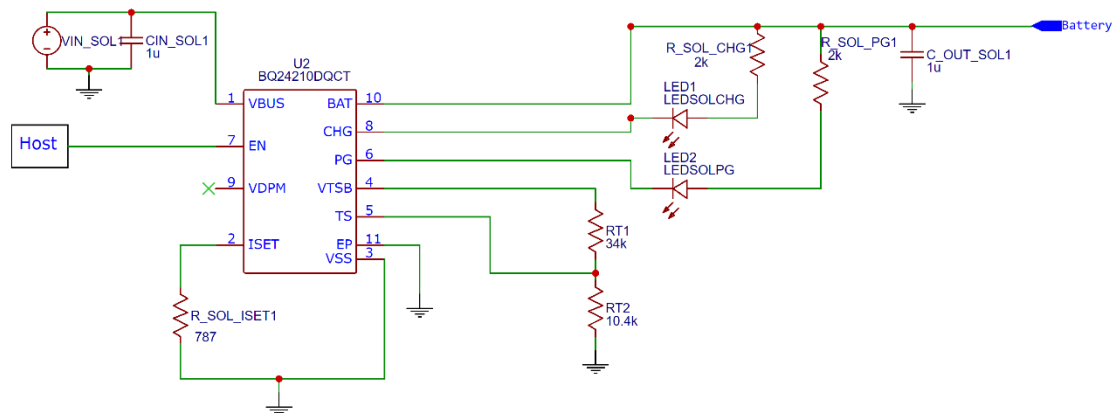

Figure 26: Schematic of the solar battery charging circuit

The "host" block in 'Figure 26: Schematic of the solar battery charging circuit' shall for now be tied to the voltage input (VBUS) so as to always enable the device when there is voltage present, though it is possible that during prototyping the pin will be connected to a digital output pin on the MCU, allowing disabling of the device and saving of leakage current that causes battery discharge (though this effect is small). Note that this is, including the USB circuit shown in the preceding section, the second device that connects straight to the battery and load. The high end of the battery and the load shall be connected together through a common bus, with the load being the DC-DC converters that will produce the voltages we need to run the subsystems of the device.

The only form in which these devices are offered is a 10-pin WSON package. The packages are 2-mm x 3-mm, and so are quite small, making for a small PCB footprint in spite of the need for some external components. In addition, there is a thermal pad on the bottom of the devices for ensuring cool operation under high loads; this is generally connected to the ground plane so as to enable the use of the copper ground plane of the PCB as a heatsink. The use of these planes on the board as heatsinks is also why the thermal pad pins are usually connected to the ground or VSS pin on many integrated circuits – it is understood that the thermal plane is also serving as a ground plane, and thus the grounding pin can be safely connected to the thermal pad without concern for any potentials that might develop. While this is desirable to reduce stray ground noise and EMI in the devices on and around the board, if a mistake is made in connecting this pins such that a potential does develop between them, the device will very likely be destroyed as a result.
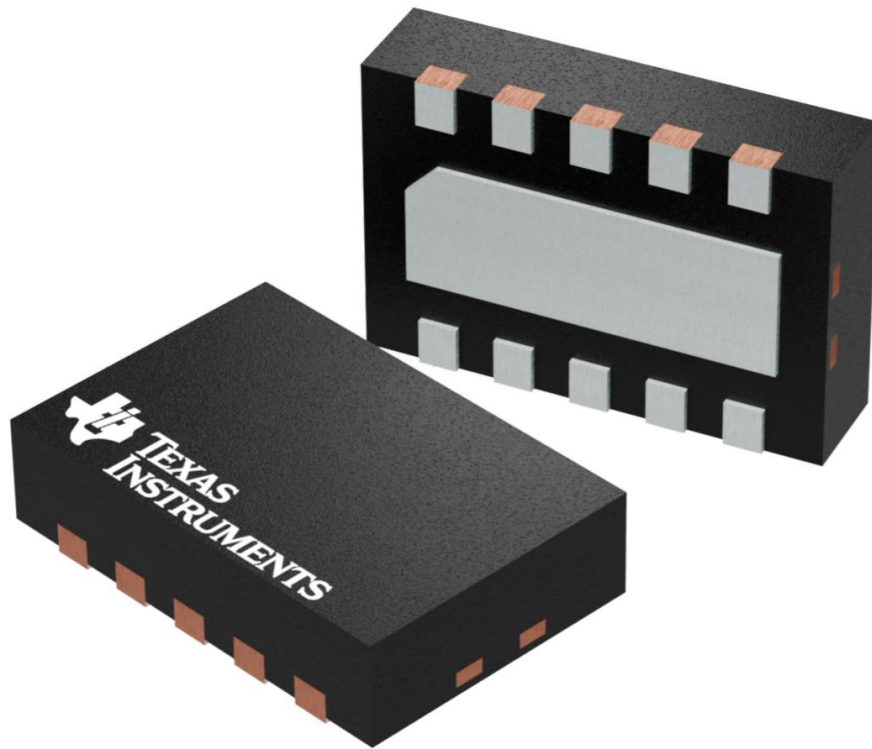
Figure 27: Image of 10-pin WSON package such as the package of the TI BQ24210
Credit: Texas Instruments

4.1.4.3 Battery

Though it has been alluded to over the course of the document leading up to now, a proper discussion of the battery intended for use in this device has not been had up until this point. The goal for the battery in this device is the storage of a large quantity of energy in a compact, lightweight, memory-free (or close) format; and though the additional goals of being as safe as possible and relatively free of self-discharge are present, they are not the absolute highest priorities of this aspect of the project. However, both of these concerns have been able to be at least reasonably-well achieved through the use of a Li-Ion hard-shell battery – not to even mention the enormous margin by which this format surpasses our requirements.

At the time of writing, Li-Ion hard-shell batteries exist in the marketplace with built-in self-protection circuitry that prevents excessive voltage or current on the actual chemical cell – something that *vastly* improves the safety of the device (as well as the long-term reliability, for that matter). This has proven to be one of the motivations for the selection of a Li-Ion battery over a Li-Polymer – the other being that the hard shell of the Li-Ion cells provides extra rigidity that is valuable in a device that will be used in wilderness (and thus be in danger of abuse from time to

time). In addition, the self-discharge of these batteries is quite low, so both of our stretch goals have been reasonably well-realized.

Of course, the main goals should be the primary concern – how well can *those* be satisfied through the use of Li-Ions? Well, at present the battery under consideration for this project is a solution sold by the vendor Adafruit that is a cylindrical battery with a capacity of 2200mAh and dimensions of 69-mm length and 18-mm diameter; the nominal voltage is 3.7 volts, and the maximum charge voltage is 4.2 volts. It has a mass of 46g, and lacks temperature-monitoring thermistors. All of these specifications well-exceed our expectations of the battery system in light of the load we anticipate it needing to service.
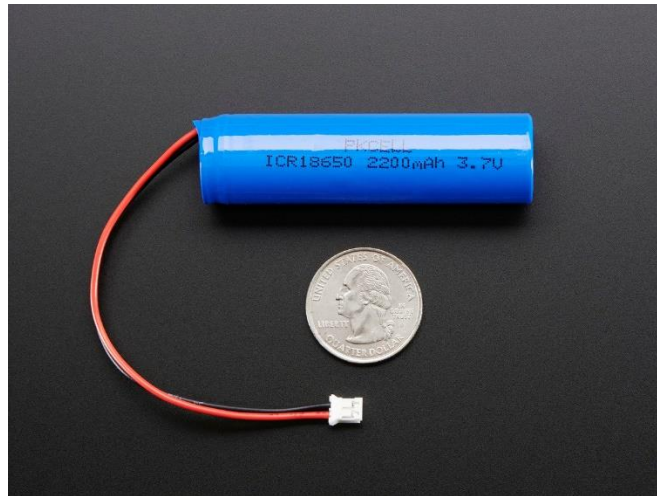

Figure 28: Adafruit battery under consideration, with a U.S. quarter dollar for scale

4.1.4.4 DC-DC Converters

Now that the systems for charging the battery as well as a potential battery, have been determined, the next logical design step is the DC-DC converters that will turn the voltage at the battery into what's needed to run the subsystems of the device, as well as regulate these voltages so as to keep them steady even during loading peaks and troughs. Let us consider the voltage requirements of each of our subsystems to determine what our voltage rail(s) need to be; this will be done through the use of Table 16: 'The requisite voltages of each of the device subsystems'.

Subsystem	Required Voltage (V)
MCU	1.62 – 3.7
XCVR	2.6 – 3.6
Keyboard	3.3 (nominally)
Display - LCD	±2.7 or ±3.3 or 5
Display - Contrast	5
Display - Backlight	3.0 – 3.6

Table 16: The requisite voltages of each of the device subsystems

Two things immediately jump out upon inspection of the table: first, there is no one voltage that covers all of the things in the table; and second, a voltage of 3.3 volts would be able to cover most of the things in the table. In fact, a 3.3-volt rail would cover every voltage needed except the display contrast voltage. As such, our main voltage rail to cover most of the device ought to be 3.3 volts, and we can have a secondary one that's 5 volts for the display contrast line as well as other small things (such as status indicators) that can be off-loaded to the 5-volt rail for the sake of load balancing.

Besides voltage, though is another consideration: current! The currents in each device subsystem are mostly very small – on the order of a few milliamps or less (including the MCU). However, there are a couple of systems that have large current draw: the Digi XCVR, which has a current draw of 40 milliamps while receiving and 900 milliamps while sending; and the display backlight, with its current draw of 45 milliamps. Most of the life of the device will be spent "listening" or receiving data rather than sending it, thus meaning that most of the time the XCVR will only be in need of 40 mA. In addition, the backlight of the display will not be turned on most of the time, and thus that 45 mA draw is relatively minor in terms of overall current drawn or energy used over the course of a battery charge. However, the abrupt switching back and forth between 40 mA and 900 mA on the part of the XCVR leaves (or by all rights should leave) the designer concerned for the voltage stability of their power supply output, as current is drawn from the output capacitors faster than they are replenished. Fortunately, as per the discussion above, the effects of current draw on a capacitor decrease linearly with increasing capacitance. This thus leaves the insightful designer with an obvious solution: place "bypass" capacitors on the power input of each subsystem, sized according to the amount of current they might need on short notice, thus reducing their influence on the voltage across the output capacitor on which they all rely.

Given all of this discussion then, our first step in the design of this stage of the PSU should be the design of a 3.3-volt DC-DC converter with a high current capacity. For this purpose, it is desirable to make every effort to design a circuit that is as efficient as possible, since our energy has already had to pass through the

inefficiency of the battery charging circuit(s), battery, small contributors such as resistors, wires, traces, and the like. Thus, a switching design will be sought-after for efficiency's sake. Conveniently, however, of the many reference materials offered by Texas Instruments to encourage the use of their products, one of them is their Webench service, which provides the user with the ability to input the required circuit input and output voltages, and the requisite current-handling capabilities; the service then allows for optimization and filtering based on numerous factors, and the ability to export the generated circuit. Through all of that, then, the service serves to act like an applications engineer that suggests possible circuits for your application that use their company's products. In this way, it was reasonable to employ highly-efficient switch-mode DC-DC converters, as devices that were dependent on serial control could be filtered out, and those with complex external networks avoided. In this way relatively simple, yet high-performance designs were generated for both the 3.3-volt and 5-volt device rails.
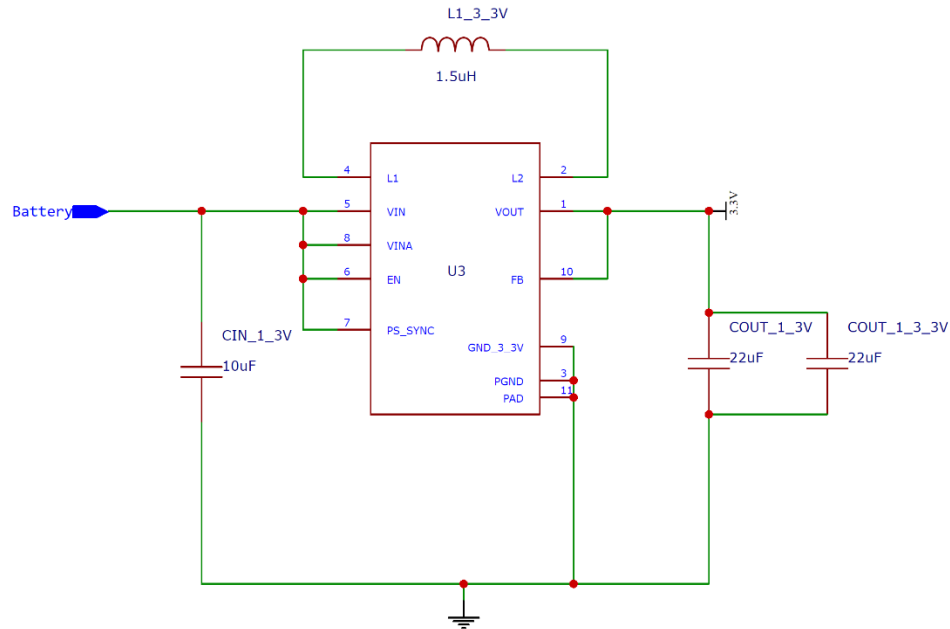

Figure 29: The Webench-generated switching DC-DC converter for the 3.3-volt rail

It is intended that all of the device subsystems will run off of the 3.3-volt rail except for the display contrast line, some indicators, and the LCD backlight. Though the LCD backlight is in the voltage range to run off of the 3.3-volt rail natively, the 5-volt rail will be under very little load, and as such it would be good to move one of the larger current drains onto this system so as to reduce the load on the main rail – especially since the backlight will be a system that is turned on and off fairly quickly and sporadically as the user needs. This stepping down of voltage could be accomplished fairly easily through the use of diodes – though it is not the most efficient scheme possible, the relatively small proportion of the time the device will spend with the backlight activated means the extra cost and footprint associated with a proper step-down converter is likely not justified.

Figure 30: The Webench-generated switching DC-DC converter for the 5-volt rail

4.2 PCB Software & Design

Easy EDA is a free software application aimed as the use for hardware and electrical engineers. This application includes a multitude of different design capabilities, ranging from drawing to wiring in schematics as well as the design of printed circuit boards. The software contains a plentiful number of presets, such as the schematic and PCB libraries that are very common in the engineering world. As far as the UI of the program, it is relatively easy to use for beginners, yet offers users as a powerful design tool. There are a few features to this software that gives this specific application it's title as being a powerful tool. A Bill of Materials may be generated using easy EDA, which is extremely convenient for designers that are creating a product. There is also Gerber, which is a feature that involves an ASCII-styled vector format of two-dimensional binary pictures. This is specifically aimed towards descriptions involving PCB design, namely copper layers, data, masks, legends, etc. Still, other features are present such as component presets, which include: resistors, capacitors, inductors, diodes, transistors, switches, voltage and current sources and more. One main contributing factor into why we used this software for schematic design is due to its versatility towards other design software. The schematic and PCB files can be imported and exported from easy EDA to other design software, namely Eagle and KiCad. This becomes especially important when you want to transfer over work established on these different types of software to a common application. This may also be reverse engineered, such that you may design something on Easy EDA which may then be imported over to other applications for further use.

Another very famous PCB design application is Eagle. This software has been used for decades, and was acquired by Autodesk as of 2016. This application is known specifically for it's vast features. It contains a schematic editor which is typically utilized for the design of circuit schematics, and also has a PCB layout editor. Eagle contains a wide variety of built-in libraries consisting of endless number of electronic parts. This includes the component's schematic, PCB, footprint, etc. One very unique and convenient feature offered for this software is the auto-routing pertaining to the schematic made by the user. The traces are automatically made in the PCB editor, which is based off of the same connections made in the schematic editor. There is a multitude of GUI windows within the program, and a menu system which digs deeper into the range of tools used for design. As far as interfacing with Eagle, keyboard and mouse input may be used for editing, however, specific commands may be utilized at an embedded command section. Similar to Easy EDA, Eagle saves layout files including Gerber and PostScript which are essentially a file format that the majority of PCB fabrication businesses use. However, Eagle's userbase consists typically of smaller design companies that generate their own board files, and so these companies also accept these filetypes. This acceptance of the board file is very important, and is the reason why many designers and engineers use eagle as their design software. This essentially allows for the wide range of compatible filetypes that fabrication companies will accept.

Yet another printed circuit board editor is KiCad. This software is an electronic design automation, just like that of eagle and easy EDA. KiCad facilitates the design of schematics pertaining to electronic components, that are also translated to PCB layouts, much like that of Eagle. This software also contains tools that may generate Gerber files, bill of materials, three dimensional views of printed circuit boards and furthermore.

4.2.1 Input and Output

Taking into consideration the various PCB design software, we intend to use EasyEDA for its versatility, efficiency, and ease of implementation. The application of Eagle and KiCad will still be important for the prototyping stage of the project and will also be useful for opening relevant designs and models made by manufacturers or hobbyist. As far as the input and output section is concerned regarding the PCB, a variety of designs will need to be created. The Q10's ribbon connector, the Hirose BM14B will need a specific PCB design for it. The connector will need to be created, and the dimensions will need to be precise for functionality to ensue. During the prototyping stages, an adapter board will be used which will take traces from the conductors of the connector and run to through-holes at the edges of the PCB. Some soldering of these holes will be necessary, but once completed these holes may be easily accessed for interfacing directly with the microcontroller. After functionality based off the programming is accomplished by the computer engineers, further design of the connector can be made. This will consist of proper traces that will run directly from the connector design to the MCU. It is imperative that the dimensions of the connector are properly set up during PCB design. This will ensure that when the connector is soldered to the board, it will fit correctly and the traces will be correctly aligned, which will ultimately save ourselves time and potential complications. The figure below of the PCB design of the Hirose connector may be seen below. It is seen that eight of the twenty-eight conductors are grounded, just as it was mentioned in the research section. These two columns will contain the twenty pins consisting of the different columns and rows specifying the characters made by the keyboard. Once the connector is soldered, the ribbon cable may snap into place and the traces running from this connector will go to the microcontroller.
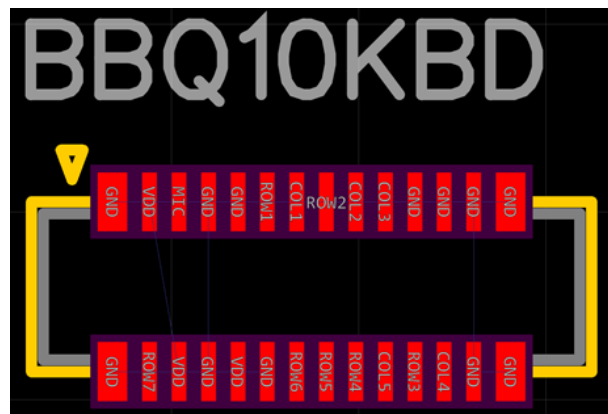

Figure 31: BBQ10KBD PCB Screenshot

The second main priority of the PCB for I/O is the use of a connector regarding the crystal display screen. The connector we will be using is the EA B200-9, which is a very simply 9-pin socket connector. Two of these will be used to interface the nine pins on both sides of the display screen we have chosen. A PCB design regarding these two integrated chips are to be made, so that when soldering is undergone the LCD may connect to both with no issue. The PCB design for this connector will be made in a similar fashion to the Q10's connector. Essentially the design will consist of two sets of nine through-holes. When the screen's pins are connected to these IC's, it will then enable the pins from the IC to pass into the through-holes. The pins will then be soldered carefully to these holes. Traces are then to be made from these holes to the correct pinouts of the MSP432. Again, it is important to note that the dimensions pertaining to the IC must be very precisely implemented into the PCB so that it is functionally sound. The figure below shows the PCB for a typical nine-pin connector. Two of these will be incorporated onto the finalized circuit board for our display screen.
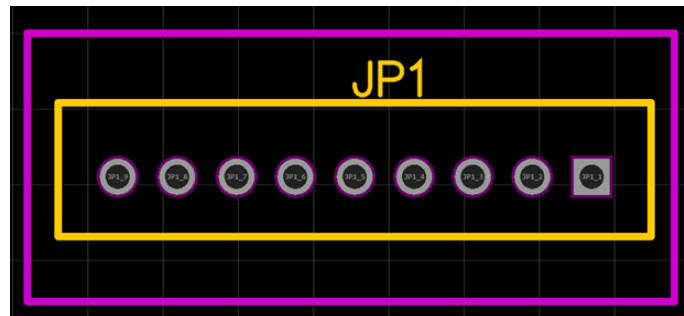

Figure 32: JP1 Standard Nine-Pin PCB Screenshot

Moving onto the status indicators, it is noted that the process is not as involved as with the Q10 or display screen. The simple resistor and light emitting diode will be enough for the purposes of our project. The resistor and diodes of interest will consist of a packaging that is small enough to be compact for the board, as our dimensions are limited to roughly around three to four inches. These components will be surface-mounted, and the status LED's will stay on the same side of the PCB where the display screen is located. A figure of the schematic pertaining to the LED configuration may be seen below.
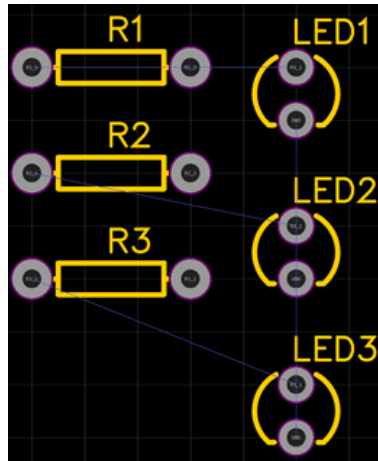
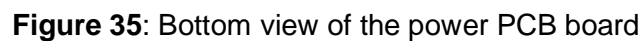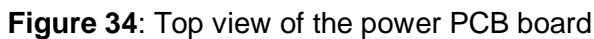**Figure 33**: Status Indicator Configuration PCB Screenshot

4.2.2 Power Supply

Upon designing each power subsystem considered in the preceding sections, they had to be combined into a single cohesive power supply design. The PCB layout was then completed; this will be spoken to in greater detail now. After producing a complete schematic containing all subsystems, footprints were selected for each component. Where possible, surface-mount components were used so as to reduce footprint; in addition to the smaller device sizes, the fact that the other side of the board behind a part remains empty means ample room will remain available for placing the rest of the circuit. All passive components were chosen as either 0805 or 1210 imperial sizes. These sizes were chosen for a couple of practical reasons: first, some inductors or capacitors simply weren't available in sizes smaller than 1210 due to their large values; and second, while 0805 isn't the smallest sizes available for many of the resistors and capacitors, it was deemed desirable that the components could potentially be soldered by hand, leaving the option to potentially opt not to use a solder stencil. While it has not yet been determined with certainty whether a stencil is to be used, the ability to use one or not was deemed valuable enough that the larger part sizes were considered a worthwhile tradeoff.

Next, several board properties had to be selected. A board size of about 2.3x1.1 inches was selected, as it was both small enough to fit inside of the intended device case and big enough to fit the entire power supply; a final revision of the board will include the rest of the circuit – this is just a prototype board for the power supply so that it can be tested, since SPICE compact models were not available for the battery charger ICs. While the power supply board is currently two layers, additional layers may be added to the final version in an effort to keep the board as small as reasonably possible.

Because of the nature of the power supply as effectively four separate subsystems that together define the PSU, there are essentially four different "blocks" in the PSU layout: the USB charger in the upper-left, the solar charger in the lower-left, the 3.3-volt DC-DC converter in the upper-right, and the 5-volt DC-DC converter in

the lower-right. There are three connectors on the board: USB, solar, and battery. It is planned that prototypes different will be constructed some of which will have the solar panel integrated in the device case and some of which will have the panel outside of the device case, so as to compare the efficiency of charging, case rigidity, and ergonomics of the two schemes. However, for the PSU board for testing a barrel connector for the solar panel will be used, as it will facilitate simpler testing procedures. A connector for the battery connection was also used in hopes of facilitating testing by obviating the need for solder and unsoldering the battery.

Pictures of the top and bottom layers of the PSU board have been included below.

Figure 34: Top view of the power PCB board

Figure 35: Bottom view of the power PCB board

While not absolutely necessary to technically be able to see the layout of the board, due to the difficulty in seeing some of the component pads and traces brought on by the ground plane 3D render have been included as well.
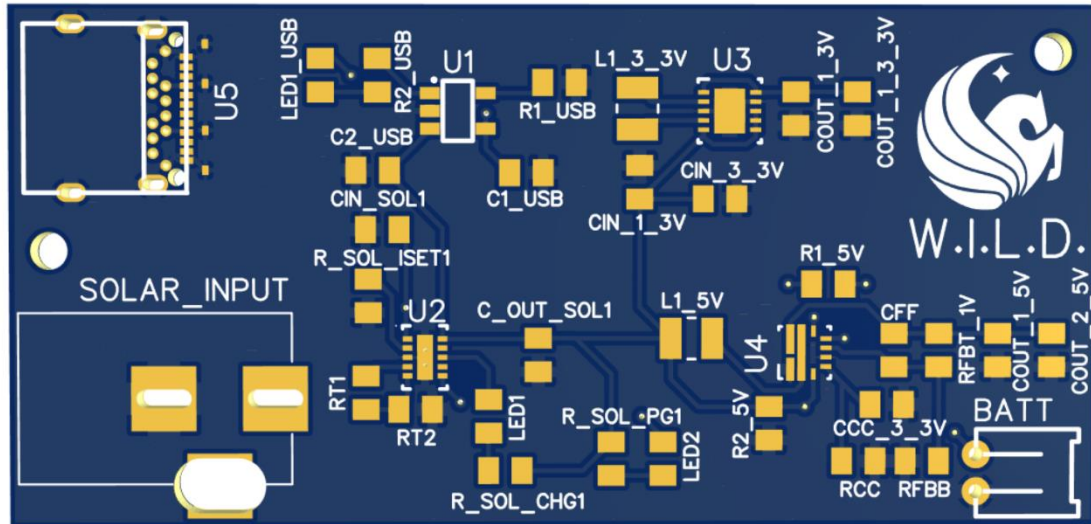

Figure 36: 3D render of the top of the power PCB board
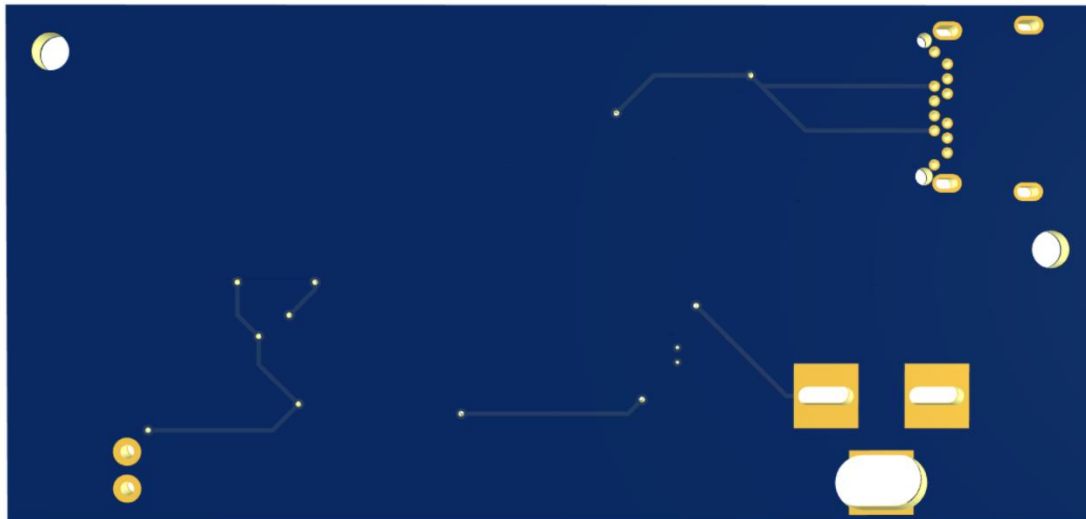

Figure 37: 3D render of the bottom of the power PCB board

4.3 Software Design and Architecture

The following section details the design methodology as well as the class structure and general software architecture. This section has been written as a reference not only for the software development phase of the project but also to serve as a guide for revisions and trouble shooting. For all code written on the project.

The Software architecture, as define in other parts of this document, consists of 3 main parts. The first of these parts is the referred to as the core. This is the portion of the device that will handle all functions of the device that interact with the input and output modules. Due to the hardware that we have chosen for this project we have deemed it necessary to include additional subsections for both the transmission of messages, as well as the control for LCD and keyboard data. Because the chip we are using for the transmission of data come packaged with support for use and configuration through a UART serial communication line, we will use this as the exclusive method of interfacing with the MCU. For the Keyboard and LCD, as well as any remaining status indicators, it is useful to segregate that portion of the software design into its own subsection. This is primarily due to the fact that the keyboard we will be using will need to have its own code to translate the input to meaningful data.

4.3.1 Core

The core will consist of three main parts. State Control will handle what the user sees on the LCD screen. This might be referred to as the UI in other more robust software developments. Because our device will only display data with the character-based LCD as well as a few status lights. The core will also handle status monitoring. It will perform routine checks that ensure the device's status is displayed appropriately. When sending and receiving messages , the device will display the status of those messages using the LCD and LED's. When the Device is in a resting state, it will show the power status as well as possibly including the ability to show the status of any active connections. The Final part of the core is the part that handles messages sent to, and received from the transceiver chip. The Input and output from this chip will need to be appended with a number of status flags and information such as the device to send the transmission to. Prior to this, the message will need to be compiled in such a way that the transmissions portion of the code can alter it and send the final version to the transceiver chip.

This core is anticipated to comprise most of the code written for the device. A priority while writing this code is the interaction with the other modules. As this module will form the basis of our device's software, it is paramount that we write it in such a way that it interacts predictably with the other modules. Any modifications made will also need to be traceable and preferably, the code to be modified will be easy to read. Because of this we plan to break the core down into several smaller classes. These classes will consist of, and only of, clearly defined functions and

variables. By breaking the structure of this module into small and manageable parts, we will streamline our troubleshooting and development process.

4.3.1.1 State Control

State control will consist of two parts. The User Interface portion, and the flag setting portion. While these two sections are separate, they are closely related in function. The flags will be used to give the user information on the status of the device and the user interface will define the prompts and screens that the user will interact with during utilization.

4.3.1.1.1 User Interface

The user interface, consisting of a character-based display, will accurately and effectively display all information needed to operate the device. Additionally, the interface has been designed to operate with ease using the input methods available to the user. As the keyboard input we have chosen will also offer a type of cursor, we can utilize this to navigate a menu. All interface pages have been designed with hardware limitations of the character based keyboard as well. With a maximum number of characters on one line being 16, minimizing extraneous information was extremely important.
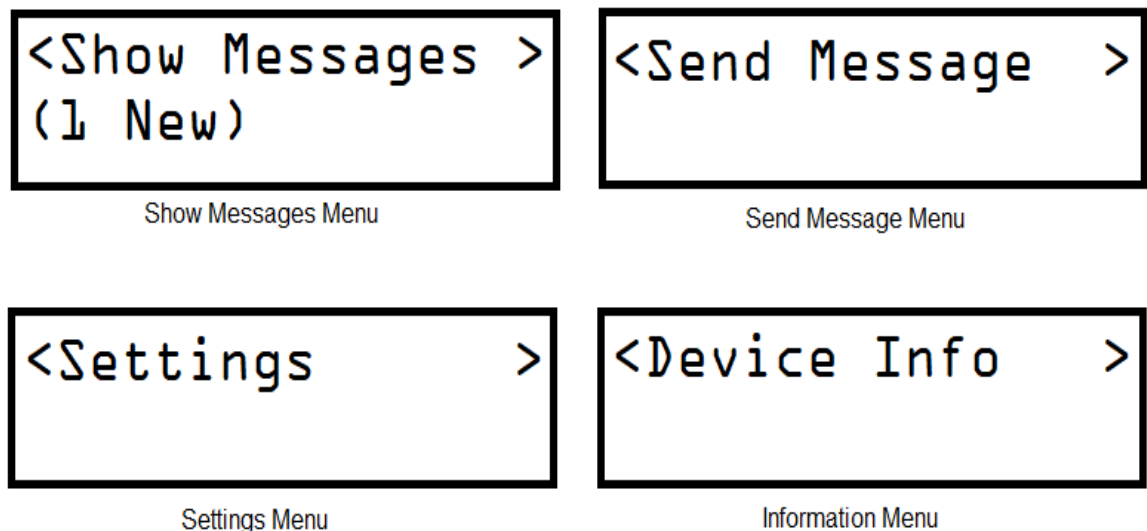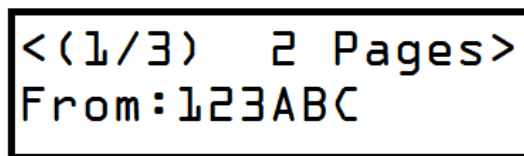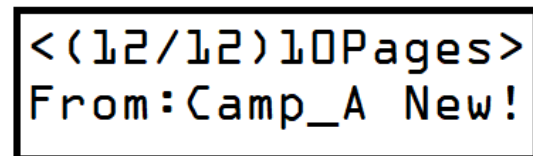
```
<Show Messages >
(1 New)
```
Show Messages Menu

```
<Send Message   >
```
Send Message Menu

```
<Settings       >
```
Settings Menu

```
<Device Info    >
```
Information Menu

Figure 38: User Interface Example 1

The Main menu will consist of roughly four options. These four options will be: Show Messages, Send Message, Options, and Information. The Show Messages menu will allow the user to read messages that have been sent. The number of messages available will be limited by the programming, as well as the hardware limitations imposed by available memory. The top menu for this portion of the device will also show the number of new messages, if any, that the device has received and that have not been viewed. The Send Message will allow the user to send a message to another device. This menu will also have built in options that are deemed necessary for each message. One such setting is specifying whether the device will receive a read acknowledgement from the destination device. Due to the emphasis on using this device in potentially dangerous or emergency situations, the user might need to know that the message has been read; or the user might wish to save the battery power required to send and receive additional information. The third menu is the Settings menu. This will be used to set global settings that are used at all times the device is powered. Options such as the power level and sleep or wake times will be included. Finally, the Information Menu will offer information on the device such as the device ID, version information, and usage statistics. This menu could also be used to display things such as current voltage and other information that might be useful but would clutter the display if it were included in other areas of the user interface.

When a user proceeds through the Show Messages menu, they will be greeted with a page showing the first message in the list. Messages will be ordered as received with messages that are unread being indicated as so. If the device has support for multiple page messages we will also show the number of pages that the message contains on this page. This page will also display the sender of the message. This sender will be displayed either as a device ID, or as a friendly text tag define by the user. This functionality, if added, will be implemented in the settings menu. By pressing the forward or back buttons the user will be able to select a different message. By pressing the select/enter button, the user will be able to view the message.

```
<(1/3)   2 Pages>
From:123ABC
```
Minimal message display showing device ID

```
<(12/12)10Pages>
From:Camp_A  New!
```
Demonstrating the possible device tagging function. Note the limitation on space for additional characters

Figure 39: User Interface Example 2

After entering the message, the user will then see the first two lines of text. At this point we are allowing word wrapping and will possibly implement automatic return function. With automatic return, if a user enters a word that is too long to fit on the remaining spaces in a line, the first portion of the word will drop to the second line. Due to the limited display space, we do not plane to make this feature default if implemented. With multi-page message implementation, when the user presses the right arrow key or enter key, the message will proceed to the next page. If the user is on the second page or further, pressing the left arrow will allow the user to return to the previous page. Additionally, the display will show chevrons to indicate if the message has other pages either before or after the current page. Finally when the user presses the enter key from the last page, the device will return to the menu showing the meta information for the message.

```
This is a sample    <ed on another p
message continu>    age w/ wrapping
```

Message with the default text-wrapping

```
A message           <text wrapping
demonstrating   >   disabled
```

Message without wrapping

Figure 40: User Interface Example 3

The Send Message menu starts by displaying a prompt to enter text. The user can either begin typing a message or press enter. When the user begins typing the message a cursor will appear to guide the user's input. If the device supports only one page, it will not respond to additional input after the last character the display can support. The user will be able to erase characters or use the arrow keys to move the cursor to another letter in the display. The cursor will function similar to the cursor in a text processor that uses the insert function. Any letter that is selected will be over written and then the cursor will move to the next letter on the display. If the device supports multiple pages, when the user enters an additional character after the first page is full, the last character will be moved to the next page. This is to make room for the chevrons when the message is sent. A maximum page number will be set to ensure that the device can send the message. Should the user get to this page limit, they will no longer be able to add pages, similar to as described above.
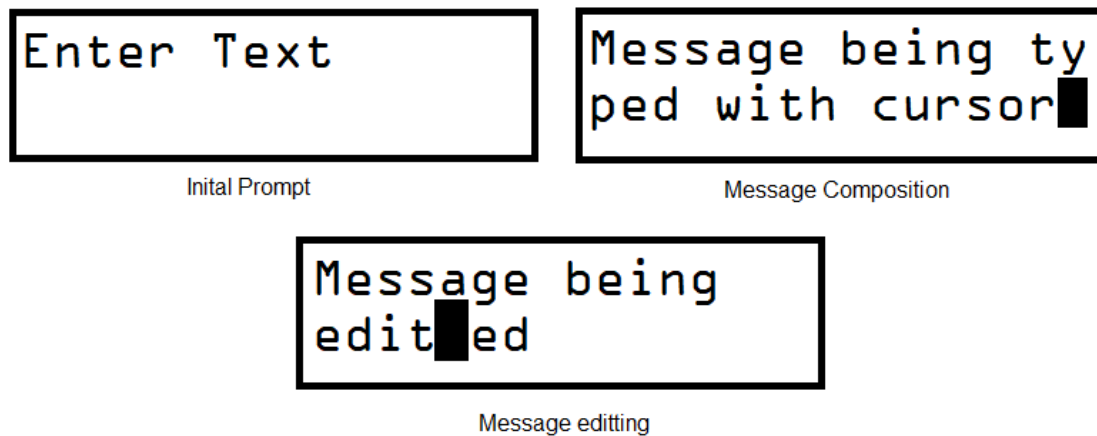
```
Enter Text
```
Inital Prompt

```
Message being ty
ped with cursor█
```
Message Composition

```
Message being
edit█ed
```
Message editting

Figure 41: User Interface Example 4

After typing the message, the user can then press enter. This will bring up the select recipient screen. Here the user will select who to send the message to. We will offer two functionalities from this screen. The user can either select a device that has been sent a message recently, or they can manually enter a recipient based on the device ID. Should the device support the use of friendly device names, they would be shown in place of the device ID's in this screen. If the user chooses to enter a new ID, the device will show a screen allowing the user to enter the id number. For the purposes of this device, we plan to use 6-character device ID's. Because of this, we will limit the entry to 6 characters and will only allow the user to continue if they have entered a valid ID number. A third option will be the alert option. This option, although last, will be selectable by pressing the left button from the first recipient. This will allow a quick way to send all devices within range of the device a message. By setting this option last it will keep it away from the default options, while offering a way to access it quickly in emergencies. After entering or selecting the recipient, the device will show a confirm dialog with basic information about the message including the device ID of the recipient and few other details. This page will also ask if the user wants an ACK. If they select yes, they will get a notification back when the message has been received, and another when read. If the user selects no, then the device will send the message and not offer any feedback on the status of the message.

```
Send To:
<abc123          >
```
Device ID

```
Send To:
<BaseCp          >
```
Friendly Name

```
Send To:
<Manual Entry    >
```
Manual Entry

```
Enter ID:
abc12█
```
Entry Page

```
To:abc123 1Pages
Get ACK? Y/N
```
ACK selection

```
Send To:
<EmergencyAlert>
```
Emergency Alert

Figure 42: User Interface Example 5

The last dialogs for the send message menu will show the user the status of the message being sent. If the user does not choose to use ACK's, the device will only show the message send status. Once the message has been sent, no further information will be displayed. If the user does choose to use the Ack's option, additional information will be displayed. If the user waits at the message sending page, the device will show when the recipient's device has sent the return ACK for the message. If the user continues to wait at this page, and the message sent is subsequently read, the device will show a message read page. If at any time the user presses the escape or enter keys, the device will return to the main menu. Additional statuses may be displayed by the status LED's. If the user elects not to use ACK's, the device will simply return to the main menu after the message is sent from the sending message screen. If the message fails to send, the device will display a screen informing the user. At this screen the user can either choose to retry the send function, or can choose to return to the main menu.
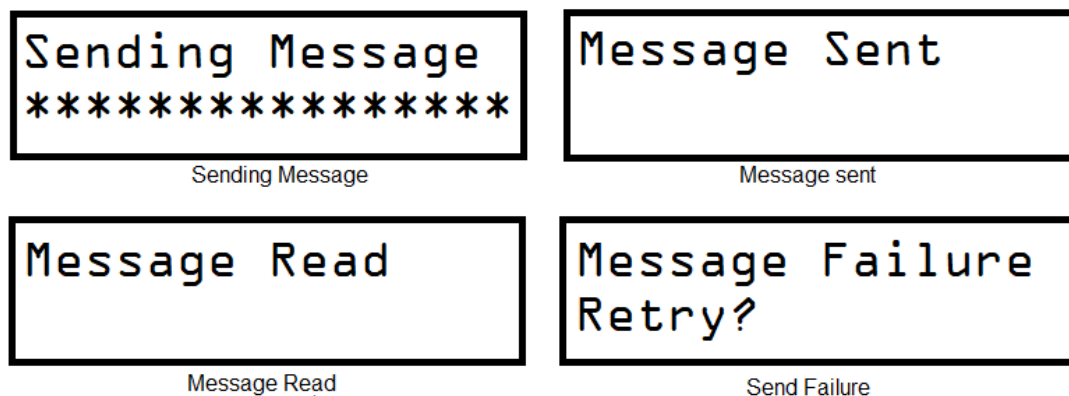


```
Sending Message
***************
```
Sending Message

```
Message Sent
```
Message sent

```
Message Read
```
Message Read

```
Message Failure
Retry?
```
Send Failure

Figure 43: User Interface Example 6

The settings menu will allow the user to define certain options for device operation. We intend to keep this device as simple as possible, so the number of options presented in this menu will be kept to a minimum. Because of this, the menu will feature an advanced options mode, that will present options that most users should not need to use. These options, which will be defined during the software development process will serve mainly for debugging and other purposes as needed. The list of options currently planned is limited, and we intend to try and maintain this. The balance between allowing the user to operate the device as needed as well as keeping the device simple is a compromise. There are four primary options planned at the current time. All options will be accessed by pressing the up and down buttons and changed with the left and right buttons. The first option is the power level. The X-Bee chip supports a number of power levels. The higher power levels give greatly enhanced range, while using exponentially more power. The power level option will allow the user to set the max power that the device will use under normal operation. The second option is a toggle for emergency power. This setting, if enabled, will automatically transmit an alert message at the maximum power, even if the power setting is lower. The third option is the potential word wrapping option. This will set whether the device automatically wraps the words or formats the words without breaks when sending messages. This setting will not affect messages that are received. To ensure that messages stay within the required restraints, word wrap will only handle messages typed on that device.
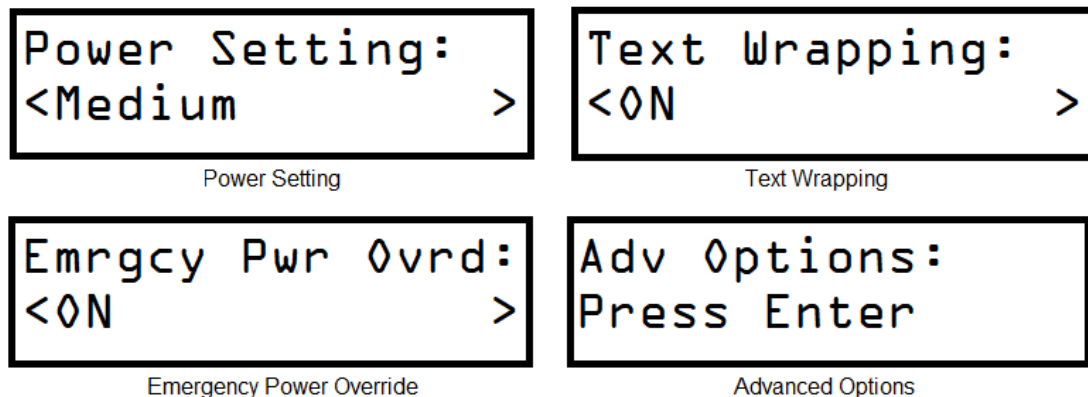
```
Power Setting:        Text Wrapping:
<Medium         >     <ON             >
```
Power Setting   Text Wrapping

```
Emrgcy Pwr Ovrd:      Adv Options:
<ON             >     Press Enter
```
Emergency Power Override   Advanced Options

Figure 44: User Interface Example 7

The last option will be that friendly device names option. This option will allow the user to assign device ID's a friendly name that is then stored in the device's memory for sending messages. When the device receives a message from a device with a friendly name, it will display that name instead of the device ID. To differentiate the friendly names from the device ID's, a standard will need to be adopted. This standard is still under consideration and will be determined during testing. One option currently being considered however is to start device ID's with numbers, and only allow friendly names that start with letters. After selecting the friendly names option, the user will be able to scroll through existing names. A limited number of names will be able to be stored on the device. The user may select a name to view the ID associated with it. If the user does this, they will be able to delete this user, or simple return to the previous menu. If the user presses enter or esc, they will return to the previous menu. If they press the delete key(most likely assigned to "d") they will be shown a confirmation prompt to make sure they don't delete a saved name by mistake. The other option from the root menu is the option to add a name to the list.
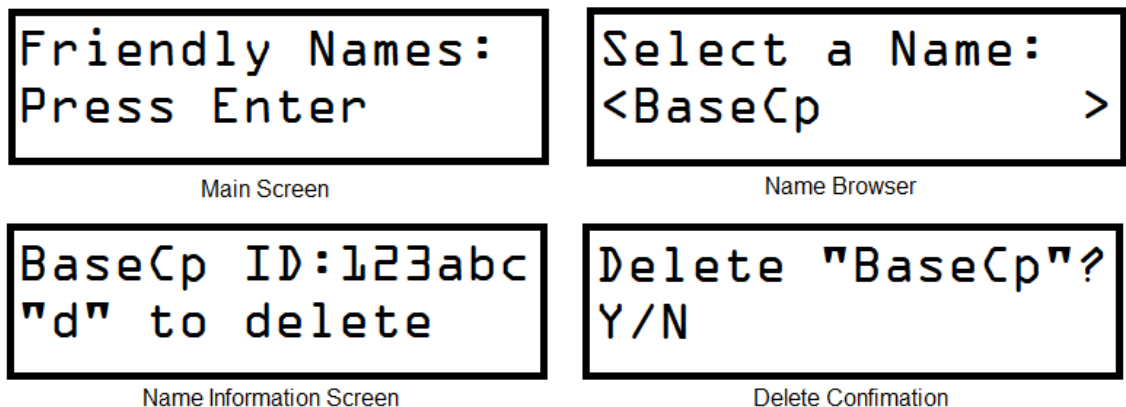

Friendly Names: Press Enter	Select a Name: <BaseCp >
Main Screen	Name Browser
BaseCp ID:123abc "d" to delete	Delete "BaseCp"? Y/N
Name Information Screen	Delete Confimation

Figure 45: User Interface Example 8

The add name screen will present the user with the necessary prompts to add a name to the friendly names list. This dialog will have 3 screens. The first screen will ask the user to enter the device ID for the name to use. This ID must be the proper length for an ID. Another feature we might add is to check that the ID is on the network in range. After confirming the device ID, there may be an automatic check to see if that device can be found. If the device is not found, the device ID entry screen would be shown. After selecting the device ID the name selection screen will be shown. This screen will enforce all rules for name selection, including limiting the name to 6 characters. After entering the name, a confirmation page will be displayed that will allow the user to review the information before confirming and adding the name to the database.

The last section of the top Menu is the Info Menu. This menu will feature any information that is deemed necessary during the development and testing phases. While there are no current items in this menu, we anticipate that it will be necessary. Because of this we have listed it at the root of the device's menus. It is possible that during the development this menu will be moved to a submenu inside the settings menu.
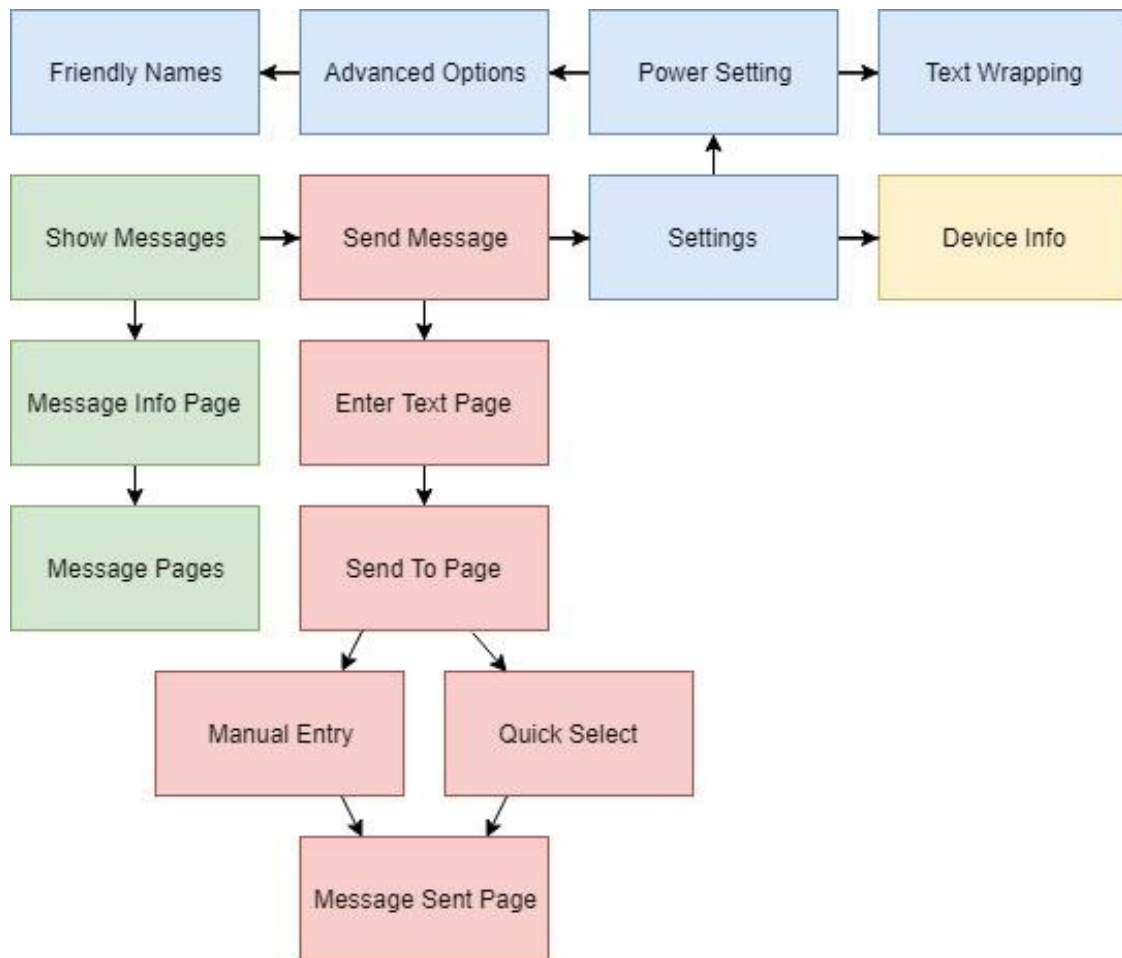
Figure 46: User Interface Flowchart

4.3.1.1.2 Flags

The State control portion of the software will be responsible for ensuring that the devices various components respond in the manner in which they are meant to. The primary concerns that this portion of code is meant to address are the need to monitor the status of various components should one or more components produce errors, as well as giving the user information on the status of operations such as send and receiving messages during normal operations. This module will utilize general purpose input/output pins. These pins will be used to connect the status lights to the code. The light emitting diodes used will either be RGB diodes, or a number of various single color diodes. The main function for this code will be a call to check binary flags provided by the IO devices. Depending on the status of these flags we will set different lights. Should the number of IO pins needed be determined to not be an issue, we will assign each light to a flag. If it is an issue, the lights can be time multiplexed and the device documentation will provide a list of light codes.

The method for checking these flags will be callable from other functions, as well as be run at a predetermined interval. Should battery use be adversely affected by the frequency of this check, the setting may be adjusted. If there is some tradeoff in device reliability and battery life, a setting might be added to the devices options menu to allow the user to decide the best state check frequency. During certain operations it might be needed to check the state of a flag before or after the operation is performed. One example would be the action of sending a message. After sending a message it should be confirmed that the message was transmitted after the device has accepted the input. This will ensure that the device is working properly and attempting to send messages even if the message is not acknowledged, or if ACK's are not enabled.

The statuses that we plan to monitor at this time also include a few creature comfort features. The device will indicate to the user when a message has been received by illuminating a status light. The device will also feature a light indicating that the power is on and/or a light that will indicate that the device is actively listening for messages. Additionally the device will allow the user to run diagnostics such as a dummy message to verify that the device is connected to the mesh or peer to peer network. While the device might benefit from a setting that would allow the user to determine if mesh networking is allowed, it is not a feature that we plan to add at the outset. This is a goal that the development team will need to determine priority over the course of the development cycle.

4.3.2 Message Translation

The chip that we have selected handles all input and output over a UART connection with an MCU. The use of the UART connection will allow the device, as well as the user, to not only handle the sending and reception of messages but also the configuration of the transceiver module. This is defined in the documentation for the module provided by DIGI.

The chip has built in support for a variety of configuration options. The main concern with this method for configuration is that there is no direct way to change the settings for the transceiver chip without writing software to handle the commands necessary for the device these options to be selected. Because of this limitation all configuration options will be handled by the LCD display and input selections. The device will also offer two advanced developer configuration options. The device will have a developer configuration mode that will directly send UART messages to the device that will allow the full suite of features that the transceiver chip supports to be implemented. This mode will be enabled by a non-trivial combination of inputs to avoid the device being put in developer transceiver configuration mode by mistake. Because this mode will allow the device to respond to a number of commands that are not inherently supported by the MCU, the device will not show any confirmation for these commands. This implies the user, while executing commands in DTC mode will need the refer to documentation to ensure the device is configured properly. The device will at a minimum display the UART text to be sent to allow the developer to check that there are no errors in the commands. After entering commands in this mode it is possible that the user will set a command that will hinder the operation of the device. To guard the device against errant command entry, there will be a device configuration reset feature. This will be activated, similarly to the DTC mode, by a non-trivial input combination. This feature will then show a confirmation dialog prior to the user executing the command.

As these features are within the same module, there is some overlap between the different sub-modules. During the software development phase we will determine the best way to relate these features in a way that allows for easy troubleshooting and development later on.

4.3.3 Input and Output Translation

The Second main module for the software of this device is the I/O module. This module is tasked with translating data sent to and from the user interface devices. As the translation of messages sent to and from the transceiver chip is handled internally, and the user will typically never see these, that function will be found in the core module. The IO module will have 3 main subsections.

The first of these subsections is keyboard translation. The keyboard translation sub-module will take input from the keyboard and translate that input before sending it to the core module. This will allow the user to enter text-based data as needed. The text-based data will then be handled properly to produce the expected result. The keyboard will be interfaced with the MCU via a number of general-purpose input/output pins. The Pins used will form a 2d array that will allow the device to determine a wide range of button inputs from a limited number of pins. As the keyboard will also have the controls for menu navigation, the number of buttons used will be quite significant, requiring more than simple alphanumerical keys. The keyboard we plan to use for the prototype device is a repurposed smartphone keyboard. While this keyboard will be compact and allow the device to feature all the input buttons and modifier keys needed, there will be a significant amount of code written to translate the input to a useful data stream. The keyboard also supports a backlight option. This is something that we do not plan to enable, due to the focus on low power usage and infinitely sustainable solar power. There is the possibility that an option will be included later in the development for this feature; therefore, while it will not be supported during the initial development, it should be noted that the keyboard sub-module will need to support its addition.

The second sub-module is the LCD sub-module. The LCD that we have chosen supports both 8bit data bus and 4bit data bus connections. As the keyboard will require a significant number of pins, we would like to keep a number of extra pins for additional status LED's and I/O pins that are "hot pluggable". The LCD will operate in 4bit mode for this reason. Because of this, the device will need to send two 4bit data sets for each letter sent. The commands, as defined in the LCD's datasheet specifies the data that will need to be sent. This module will also handle updating the display as the user selects different menus.

As the device will mainly use a system of menus combined with a few pages for user input and display of received messages, this sub-module can be further broken down into two more subsections. The first subsection shall consist of the predetermined menu pages. These pages will be set to display a title for the current menu item and show directional arrows that indicate when there are other options in various directions. These menu pages will each have pointers to the

proper associated menu pages so that when the user presses the arrow keys, the proper page will be displayed. The second group of pages is the group that will be custom made. These pages will consist of the pages that take user input, and the pages that display custom messages.

The third submodule will be responsible for handling all other input and output devices. The status lights will be controlled by the state control sub-module; therefore, they will not be accounted for in the I/O. Should the device be built to use RGB LED lights, they might be used for this submodule. There are a small number of settings that might be implemented with hardware switches. To save battery power, the device might include an option to power the device on but put the device in airplane mode in which the device will not send or receive messages. This is an option that will be determined after determining the battery usage effect of having the device's antenna constantly powered.

4.3.4 Performance Considerations

The device will be operating, at all times, either on battery power or battery power with solar charging. Due to these conditions, the device will need to reduce the overall power consumption as much as possible. To keep the device small and portable, reducing the power consumption will not only increase battery life, but also allow for smaller solar panels. The device will also need to send and receive messages as quickly as possible. The size of the messages sent will be quite small. The small size of the messages will allow fast transmission by the transceiver chip. This does not preclude the need to consider the processing time for the messages on the MCU after the message is transmitted.

Another consideration for performance is memory usage. To avoid memory allocation errors, development should focus on minimizing the memory that is actively used, as well as ensuring that there are no memory leaks. Because the device software will be written in C, special care should be taken to manage memory properly. As a safety measure, the device should clear all memory when it starts up. The device should also have the option to clear all messages and data that is stored on the non-volatile storage.

The main source of power draw is the transmission of messages. The second highest power use is the persistent monitoring for messages to be received. The device must always be capable of receiving messages, however, a solution that allows the device to only search for messages a few times per minute might save a considerable amount of power. Should development find a way to implement this feature without risking lower transmission success rates, this will be a priority to implement.

4.4 Casing Design

As far as casing is concerned, we wish to keep the dimensions to a size that will let the user feel comfortable when typing. Considering the keyboard is two point nine inches in width, we'd like the width of the casing to be about half an inch to an inch wider at most. The length and height are dependent upon the design of the printed circuit board and the relevant components inside. As for the casing of the device, we have considered various types of material including plastic, wood, and metal. Ideally, we want to use plastic as our choice of material concerning the casing of the device. The reason for this is because of 3D-printing. This technology has become very advanced in the creation of miscellaneous objects, ranging from hobbyist entertainment to full medical use. Keeping this in mind, the use of 3D printing appears as the optimal choice for creating the casing of our device. This type of casing will be relatively inexpensive, and the design may be printed in a short amount of time. Taking into consideration the structure of the design that will be printed is important factor, as this will be the backbone behind the device's safety and protection from environmental harm.

Once the proper drawing of the ergonomics and appearance of the casing is made, the design may be processed. This will take place in a three-dimensional modeling program. Fortunately, there is a multitude of different modeling programs and tutorials on how to use them. To name a few, we are interested in using Blender, Autodesk Maya, 3D-Coat, AutoCAD, etc. We aim to design a case that has the capabilities of handling drops, specifically at a height of at least 3 feet. This will need to be one of our main concerns when designing. The case will most likely need to have a type of plastic that is reinforced, such as carbon fiber, and with multiple layers. If implementation of this is successful, then the case will most likely be able to handle the mechanical requirements our team imposed from the previous section of our document. Furthermore, the case will likely need to be designed in such a manner that transportation is easy. We will need to devise a layout of the case such that all the components will fit accordingly. There will also need to be some implementation of a design that will allow for protection against weather, namely rain, as any contact it has with our hardware will most likely render the device dysfunctional. It is understood that while these modeling programs may have some complexity involved with them, through enough research and tutorials a basic design can be made. This digital design will then be saved by its filetype and uploaded to a printer that can handle the dimensions specified. Once this is printed out, we will be able to test out the durability of the casing and establish a determination as to whether it is suitable as protection for our devices.

5 Prototyping and Testing

5.1 Software Testing

Status LEDs: The Wilderness Information Link Device uses status LEDs in order to communicate its current state. There are several possible states that the device may be in, including an off state, a charging state, an idle state, a message sending state and a message received state. View Figure 40: Status LEDs UML Activity Diagram in order to view the states of the device as well as actions that will occur relating to the status LEDs when any of the states are activated. When in the off state, there are no processes running within the Wilderness Information Link Device. Unless the device is charging, all LEDs will be off and no messages will be received or sent. When in a charging state, the yellow LED will steadily blink until the device is no longer charging. The device will trigger this charging state whenever the solar charging capabilities of the device are in use or the device is being charged by way of USB. Once the device is fully charged, the LED triggered when in the Idle state will begin. When in the Idle state, the device is actively waiting for transmissions to occur. No actions involving sending a message occurs while in this stage. To signify an idle state, the yellow LED on the device will illuminate without blinking. This is the default state of the device. It should also be noted that the device can be in multiple states at once—for example, when the device is both idle and charging. When in a message sending state, the device is in the process of sending a message. When this stage is over, the device will momentarily enter a 'Message Sent' stage, where the green LED will illuminate in a solid, non-blinking state for five seconds before signifying the end of the state. The end of the message sent stage automatically results in a return to the Idle state. As the device is always waiting for a transmission to receive when in the Idle state, the device will enter the 'Transmission Received' state, where an incoming messaged has been logged as received by the device. The device will then illuminate the red LED to signify that a message has been received for the user to view. Once the message has been viewed and received, the red LED will turn off and the device will return automatically to an idle state.

Once all basic functionality of the Wilderness Information Link Device has been verified as functional and acceptable, testing of the status LEDs can be done. The following procedure can be used to make sure that each LED is blinking as they should during each of the aforementioned states that the device can be in.

Off State: When the Wilderness Information Link Device is in the Off state and not charging, the device should be completely inactive. No LEDs will be on in this state. While still off, test the device by engaging USB charging. While off yet charging, the device should signify activity by showing a blinking yellow LED. Stop USB charging the device. Now, place the device in an illuminated environment such that the device may begin charging using solar power. After placing the device in a well-illuminated environment, check to make sure that the yellow LED is blinking to signify charging is occurring. Remove the device from the illuminated environment—this should cause the yellow LED to stop blinking.

Idle State: When the device is in an Idle state and not charging, the device should be actively waiting for a transmission to either occur or begin. When in an idle state and not charging, the yellow LED will be illuminated continuously and not blinking. While idle, test the device by engaging USB charging. While idle yet charging, the device should signify activity by showing a blinking yellow LED. Stop USB charging the device. Now, place the device in an illuminated environment such that the device may begin charging using solar power. After placing the device in a well-illuminated environment, check to make sure that the yellow LED is blinking to signify charging is occurring. Remove the device from the illuminated environment—this should cause the blinking yellow LED to return to a continuously on and non-blinking state.

Transmission Received State: While in the 'Transmission Received' state, the device has received a transmission from another device. In order to display this status to the user, the red LED will illuminate in a solid non-blinking action to show that a message is available to view. Verify that the device is able to receive transmissions by using another device to send a message to the unit under test. Once the message has been sent, verify that the red LED is illuminated on the target device. Once illuminated, this LED will not shut off until the message has been viewed by the user. View the message. After the message has been viewed, the red LED will shut off and the device will return to an idle state where the yellow LED is solidly illuminated. Please note that should either form of charging occur while in the 'Transmission Received' state, the yellow LED will continue to flash continuously regardless of the activity relating to the red LED.

Message Sent State: While in the 'Message Sent' state, the Wilderness Information Link Device has completed sending a message and will momentarily provide visual feedback that the message has been sent successfully. After sending the message, the green LED should illuminate in a solid, non-blinking state for five seconds. Verify that the green LED is illuminated for five seconds after sending the message to confirm that the 'Message Sent' state is functional. It should be noted that if the device is charging while in this state, the yellow LED will flash as well to signify that charging is occurring. If a transmission has been received by the device, the red LED will also be illuminated in a solid and non-blinking fashion until the received message has been viewed by the user. After verifying that the green LED has been lit for five seconds, verify that the LED turns off and that the device either returns to an idle state or a 'Transmission Received' state depending on the actions which may have occurred within the device while a message was being sent.
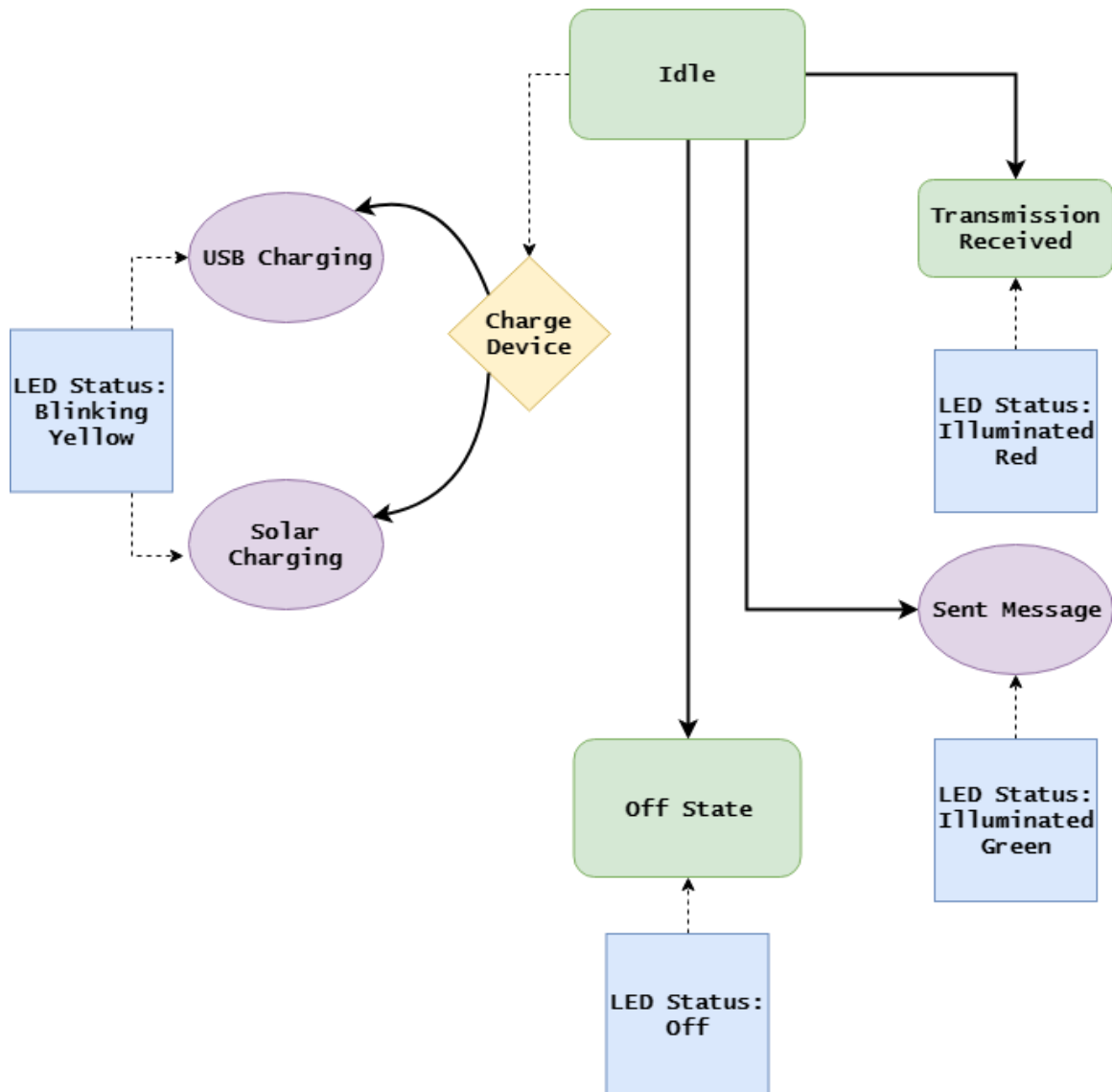
Figure 47: Status LEDs UML Activity Diagram [3]

Keyboard Functionality: The keyboard on the Wilderness Information Link Device is the primary interface given to the user by which they are able to communicate with their own device, as well as other nearby devices. Testing the keyboard and the code associated with it is a matter of ensuring that the specific key that the user presses appears on the display correctly and without inaccuracies. The following procedure can be used to thoroughly test that the keyboard is functioning as it should.

Begin by initializing a new message using the WASD keys on the keyboard interface to move the on-screen cursor. Use the return key on the keyboard to select. Once in a messaging state, the keyboard will only be able to input ASCII characters onto the screen and remove them by using the backspace key. Once a message is finished, pressing enter with change the display to ask the user if they are sure they would like to send the message. Pressing the return key will send

the message. In order to test all the characters on the keyboard, multiple messages should be sent to the device of choice. Within each message try to include every character shown in the keyboard as possible in order to make sure that every key on the keyboard is functioning and that the code is processing the correct input from the user. As long as the characters that appear on the display correctly correspond to the input that was provided by the user, the keyboard can be considered functioning as intended. View Figure 41: Keyboard Functionality UML Activity Diagram in order to view the various stages of the keyboard interface as well as main functionality during each state of operation.
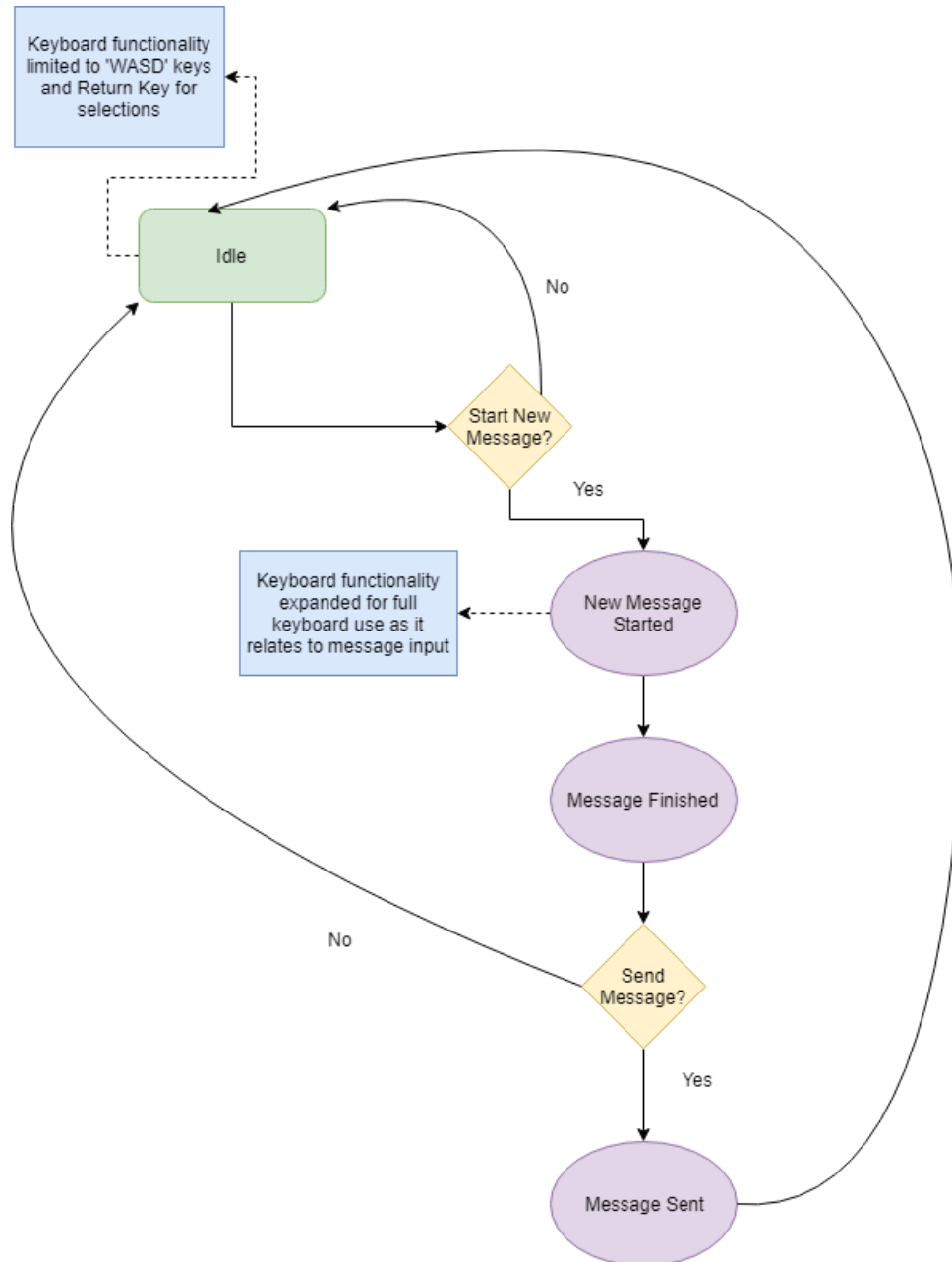

Figure 48: Keyboard Functionality UML Activity Diagram [3]

6. References

[1] "ATmega1284P," ATmega1284P - 8-bit AVR Microcontrollers. [Online]. Available: https://www.microchip.com/wwwproducts/en/ATmega1284P. [Accessed: 28-Mar-2019].

[2] "ATmega328P," ATmega328P - 8-bit AVR Microcontrollers. [Online]. Available: https://www.microchip.com/wwwproducts/en/ATmega328P#datasheet-toggle. [Accessed: 28-Mar-2019].

[3] "free flowchart maker and diagrams online," Flowchart Maker & Online Diagram Software. [Online]. Available: https://www.draw.io/. [Accessed: 22-Apr-2019].

[4] "House of Quality Template," Google Sheets. [Online]. Available: https://docs.google.com/spreadsheets/d/1o4IY09RHViCinCXxokaZXzDaCYI5eeHeV9xQ4dlW_ck/edit#gid=0. [Accessed: 22-Apr-2019].

[5] "MSP430FR5964 (ACTIVE)," MSP430FR5964 16 MHz Ultra-Low-Power MCU With 256 KB FRAM, 8 KB SRAM, AES, 12-bit ADC | TI.com. [Online]. Available: http://www.ti.com/product/MSP430FR5964/description. [Accessed: 28-Mar-2019].

[6] "MSP430FR5994 (ACTIVE)," MSP430FR5994 16 MHz Ultra-Low-Power MCU With 256 KB FRAM, 8 KB SRAM, Low-Energy Vector Math Accelerator | TI.com. [Online]. Available: http://www.ti.com/product/MSP430FR5994/description. [Accessed: 28-Mar-2019].

[7] "MSP430FR6989 (ACTIVE)," MSP430FR6989 MSP430FR6989 16 MHz ULP Microcontroller - 128 KB FRAM, 2KB SRAM, 83 IO, ADC12, LCD, AES, Scan IF | TI.com. [Online]. Available: http://www.ti.com/product/MSP430FR6989/description. [Accessed: 28-Mar-2019].

[8] "MSP432P401R (ACTIVE)," MSP432P401R SimpleLink Ultra-Low-Power 32-Bit ARM Cortex-M4F MCU With Precision ADC, 256KB Flash and 64KB RAM | TI.com. [Online]. Available: https://www.ti.com/product/MSP432P401R/description. [Accessed: 28-Mar-2019].

[9] Swarthmore.edu. (2019). [online] Available at: https://www.swarthmore.edu/NatSci/echeeve1/Class/e91/Lectures/E91(10)Serial.pdf [Accessed 22 Apr. 2019].

[10] GitHub. (2019). arturo182/BBQ10KBD. [online] Available at: https://github.com/arturo182/BBQ10KBD [Accessed 22 Apr. 2019].

[11] GitHub. (2019). arturo182/bbq10kbd_breakout. [online] Available at: https://github.com/arturo182/bbq10kbd_breakout [Accessed 22 Apr. 2019]

[12] Electrosofts.com. (2019). electrosofts.com. [online] Available at: http://electrosofts.com/8051/keyboard.html [Accessed 22 Apr. 2019].

[13] Eevblog.com. (2019). How to connect to a very, very challanging BlackBerry Q10 keyboard connector? - Page 1. [online] Available at: http://www.eevblog.com/forum/beginners/how-to-connect-to-a-very-very-challanging-blackberry-q10-keyboard-connector/msg735622/#msg735622 [Accessed 22 Apr. 2019].

[14] Instructables. (2019). I2C Keypad. [online] Available at: https://www.instructables.com/id/I2C-keypad/ [Accessed 22 Apr. 2019].

[15] Adafruit Industries - Makers, hackers, artists, designers and engineers!. (2019). Interfacing a BlackBerry Q10 keyboard into your microcontroller project #BlackBerry #Arduino #Microcontroller. [online] Available at: https://blog.adafruit.com/2019/01/14/interfacing-a-blackberry-q10-keyboard-into-your-microcontroller-project-blackberry-arduino-microcontroller/ [Accessed 22 Apr. 2019].

[16] Analog.com. (2019). Introduction to SPI Interface | Analog Devices. [online] Available at: https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html [Accessed 22 Apr. 2019].

[17] What-when-how.com. (2019). KEYBOARD INTERFACING. [online] Available at: http://what-when-how.com/8051-microcontroller/keyboard-interfacing/ [Accessed 22 Apr. 2019].

[18] Learn.sparkfun.com. (2019). PCB Basics - learn.sparkfun.com. [online] Available at: https://learn.sparkfun.com/tutorials/pcb-basics/all [Accessed 22 Apr. 2019].

[19] Scharfglass, K. (2019). Regrowing a Blackberry from the Keyboard Out. [online] Hackaday. Available at: https://hackaday.com/2018/03/08/regrowing-a-blackberry-from-the-keyboard-out/ [Accessed 22 Apr. 2019].

[20] Learn.sparkfun.com. (2019). Serial Communication - learn.sparkfun.com. [online] Available at: https://learn.sparkfun.com/tutorials/serial-communication/all [Accessed 22 Apr. 2019].

[21] En.wikipedia.org. (2019). Transistor–transistor logic. [online] Available at: https://en.wikipedia.org/wiki/Transistor%E2%80%93transistor_logic#Analog_applications [Accessed 22 Apr. 2019].

[22] Mouser.com. (2019). [online] Available at: https://www.mouser.com/datasheet/2/127/dip162-de-274831.pdf [Accessed 22 Apr. 2019].