Team08: NeverLost

Senior Design Project Report

Eric Anderson, CompE, Shelby Anderson, CompE, Louis Gencarelli, CompE, and Eric Sutherland, EE

***Abstract*—**

When hiking through long trails such as the Appalachian trail, it is not uncommon for a hiker to become injured making it difficult to complete their journey. It is often difficult to gauge a loved one's progress on these trails to confirm their location and that they are ok. In addition, many of these hikes are in remote areas with no cell service and no way of contacting emergency services or a loved one. If a hiker is injured, it might be days before anyone is aware. Our NeverLost survival beacons will provide checkpoints along hiking trails that would include a check-in system and the ability for search and rescue teams to begin with exact coordinates within minutes of their emergency. Our beacons would be accessible to everyone, especially to injured hikers that may not be able to walk to the trailhead.

I. Introduction

A. Significance

Hiking is one of the most common outdoor activities. From 2014 to 2016 a total of 990 deaths were reported in national parks which equals to an average of 330 deaths per year or 6 deaths per week [1]. Most hikers are traveling solo when accidents happen. If in a remote location with no cell service it may be difficult for a Search and Rescue (SAR) crew to locate that injured hiker and could take hours or days until that person receives any medical assistance. Around half of injuries are related to the ankle which means a hiker might not be able to make it to a spot where they can call for help. Our system is designed to address these problems and provide an alternative for injured hikers to track and call for help.

B. Context and Competing Solutions in Marketplace

In the past people have used more traditional solutions utilizing hiking travel logs and trail markers, however, these methods provide no means of communication with emergency services in case of a life-threatening incident. In recent years, devices like Personal Locator Beacons and Satellite Messengers equipped with two-way communication, GPS navigation, and additional application features are much more common. Such devices tend to cost over $300 and require additional subscriptions and licenses to operate. The use of Long Range Radio (LoRa) technology has recently become popular in low-power long-range open-source IoT devices. Similarly, BLE, WiFi, and LoRa can be used in indoor/outdoor localization, smart sensing, agriculture, and healthcare. LoRa has the ability to adapt to a large number of node devices while consuming a small amount of power. LoRa devices can be implemented with a high level of security making it ideal to use for our problem statement.

Similar solutions like Meshtastic [2] exist as an open-source mesh communication device and platform. Several low-cost Meshtastic beacons combined with a smartphone app enable in-app communication and live tracking as each beacon in the mesh is carried by users. Our device, which differs from others on the market, is designed to send precise coordinates from a fixed location to emergency services in a short period of time. Additionally, we implement a tracking system – viewable on a private HikerLog webpage – that uses beacons as check-in locations.

C. Societal Impacts

Our prototype is aimed to help and aid everyone from beginners to the most experienced hikers on the trail. The goal of our device is to provide reliable communication for anyone who might need to contact emergency services especially injured hikers that are alone. Our device will be able to operate in remote locations and will be able to be scaled with multiple devices to populate entire trail networks. Our prototype is designed to be installed on trail systems and stay there to provide infrastructure for communication to those who may not be able to afford an expensive device as described previously.

D. System Requirements and Specifications

We broke down our System Requirements and Specifications into three main subsystems: Beacon System, Power System, and Mobile Application. For each subsystem, we have set certain parameters and requirements that our system should meet in order to achieve our goals. The Beacon system should consist of a multi-hop linear networking topology with a 500m distance and a 300ms propagation delay between beacons. We have chosen to evaluate our system with two beacons and one base station at a set distance. Our goal is to reach a 97% success rate with line of sight in order to reliably send emergency packets as well as check-in messages. To confirm that packets are sent correctly in addition to determining when we need to send re-attempts, a two-way acknowledgment system is implemented. We vary the number of retransmission attempts in order to maximize the probability of a successful packet being sent while keeping total power consumption in mind.

For the next system, the Power System, we based our specifications and requirements on the battery and solar panel we chose for our design. We have chosen a 2.5 Ah LiPo battery. With low power consumption, our system will be able to be active for about 10 days in sleep mode without relying on any power from the solar. Our solar system should output 2600 mAh under optimal conditions on a sunny day.

The last subsystem is the mobile application which consists of the app and the backend that receives and uploads check-in and emergency data to the cloud platform. The mobile application platform enables bi-directional communication as well as incorporates a Bluetooth module. When a check-in message is received by the base station, a hiker's location will be updated onto a hiker's logs as well as a real-time map which is stored on a secure webpage. On the other hand, when an emergency signal is received, an SMS message will be sent directly to emergency services with coordinates showing the exact location of the beacon where the signal originated to allow a quick and accurate search and rescue team to arrive.

II. Design

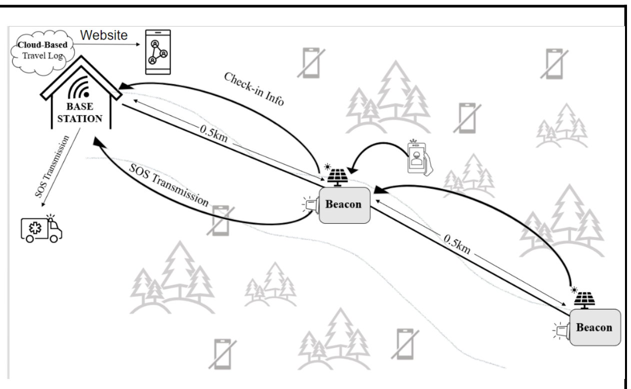


Figure 1: System Overview

A. Overview

We solved this problem by creating a solar-enabled, low-power system that allows hikers to either check-in or send an emergency while on the trail. A hiker can create a profile using our mobile application so that as they encounter a beacon along the trail they can either check-in or send a signal to emergency services if required. AES encrypted messages are broadcast from the user's mobile device over Bluetooth. Once the beacon is given data to transmit it will communicate over LoRa to the next beacon in the line. Each beacon works together to forward the data down the line until it reaches a base station. The base station is connected to power via a wall outlet and has access to WiFi. Base stations use WiFi to upload the received data to the cloud where it is either decrypted and stored on a secure Hiker Log webpage, or is paired with precise coordinates in a message to be sent to emergency services.

We considered technologies that would allow us to communicate with satellites but came across many roadblocks including budget and complexity when planning to implement these. We settled on LoRa which allows for long-range communication with low power consumption which helps to achieve two of our major goals for the project. We are balancing a trade-off between transmission range and power consumption.

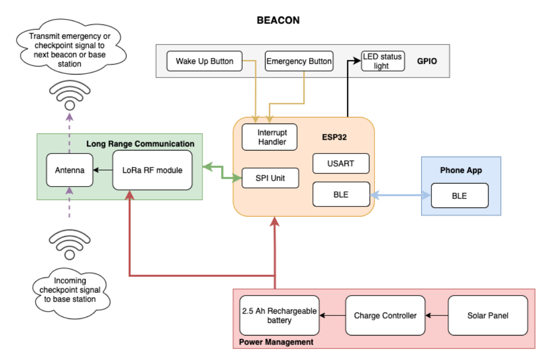


Figure 2: Beacon Hardware Block Diagram

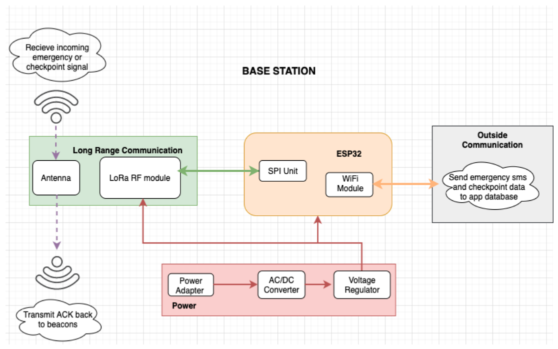


Figure 3: Base Station Hardware Block Diagram

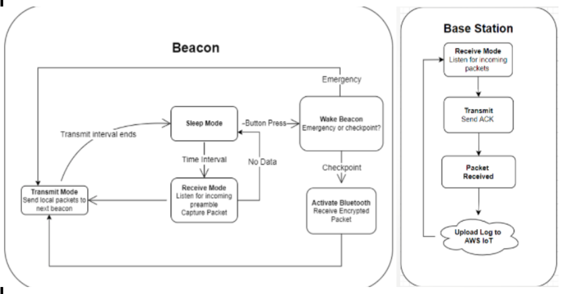


Figure 4: Software Block Diagram

B. Beacon System

The beacon system is designed to reliably forward data from the point of a check in to the nearest base station. To do this we used the RFM95W module which implements LoRa communication to provide secure, low power, and long-range communication.

Our system operates at 915MHz which is an FCC-designated Industrial, Scientific and Medical (ISM) band that allows us to legally communicate without the need to register our devices. The RFM95W allows us to set up a system of acknowledgments so that after a message is sent, the receiving beacon should send a message back to the original sending beacon telling it that a message has been received correctly. If no acknowledgment is received, then the beacon will make attempts to send the message again. If power consumption was not an important specification of our system then we could try to resend the message as many times as needed until it is correctly received. However, with power considered, we must limit the number of retransmission attempts.

We have two different types of transmissions, those being check-in data and an emergency signal. We consider the emergency signal to be more important and are willing to spend more power to ensure this message gets through to the base station. With this said, we have programmed 3 reattempts for the check-in data, and 7 reattempts for the emergency signal. These were decided through testing to ensure that check-in data is received 98.5% of the time while emergency data is received successfully at a rate of 98.99%.

This limited our project, ideally, we could transmit at lower frequencies and higher output power to improve range, but to stay within FCC regulations we were constrained. Each of these changes would significantly affect our power system.

C. Power System

Our power system is designed to allow our beacon system to be self-sustained without the need for external intervention. The goal of this subsystem is to prevent the need for a person to go to each beacon to change the battery. To ensure this was implemented properly, we designed each beacon to consume low power as well as be set up with a solar panel and charge controller to recharge the battery.

We chose a microcontroller that had options to enter sleep mode to allow for very low power when it was not necessary to actively transmit or receive data.

Our LoRa module has a feature to create a duty cycle that allows the MCU and most of the other active components of our system to sleep until a message is received. The RFM95W transceiver will wake up long enough on set intervals to catch any incoming packet, then send an interrupt to the rest of our system to wake up and actively receive and process the message. The amount of time spent sleeping is determined by the length of the preamble of each transmitted message. As the preamble length increases, we can spend more time sleeping, but also consume more power per message sent. To find the optimal period of time spent sleeping we created an equation to model the consumed power versus preamble length and found the minimum power consumed when we use an 8 symbol preamble, which leads to the 75% sleep duty cycle.

Another important component of the power system is the 2W solar cell and charge controller. To test our solar system we measured the output current every hour over the course of a day under various conditions including sunny, partly cloudy, and under tree cover. Then plotted each set of points and created a best fit line so that we could integrate and find the total power produced over a day in each set of conditions. See Figure 5 below.

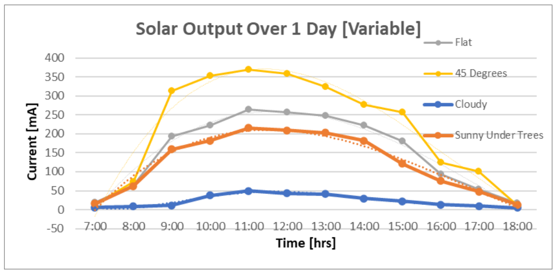


Figure 5: Solar output over one day

As previously mentioned, our system is designed for situations like the Appalachian Trail. We kept this in consideration when designing the power system. Our low sleep current means that the overall power consumed depends heavily on the number of beacons in a chain to the base station, and the number of hikers using this path. We chose to model our system on a section of the trail in New Hampshire called the Presidential Range. This 88 mile section would require 282 beacons, and based on our battery we could support 480 hikers in a single day on one full battery charge. To be more conservative, using only the energy produced by our solar panel in one day, we could support 306 hikers in a single day. We model and plot our power using:

(# Hikers)*[(# Beacons)*(8 µAh) + (1.5 mAh)] + Sleep Power

Where 8 µAh is the power used in transmitting a single message, and 1.5 mAh is the power consumed in each Bluetooth connection to a beacon. See Figure 6 below.

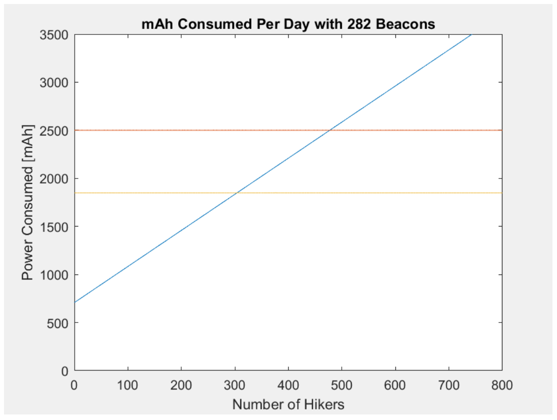
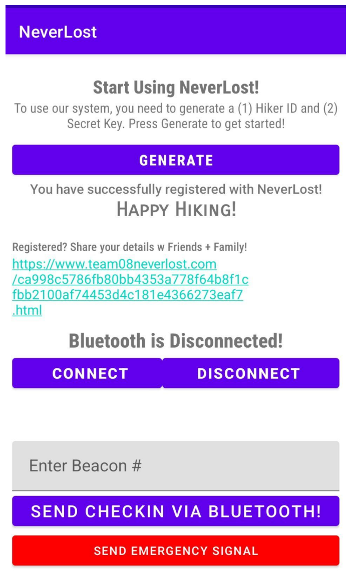


Figure 6: Power Consumed vs. Number of Hikers

D. *Mobile Application*


NeverLost

Start Using NeverLost!

To use our system, you need to generate a (1) Hiker ID and (2) Secret Key. Press Generate to get started!

You have successfully registered with NeverLost!

HAPPY HIKING!

Registered? Share your details w Friends + Family!

https://www.team08neverlost.com/ca998c5786fb80bb4353a778f64b8f1cfbb2100af74453d4c181e4366273eaf7.html

Bluetooth is Disconnected!

Figure 7: Mobile Application

NeverLost's accompanying mobile application allows hikers to update their location as they progress through the trail, regardless of their access to cellular service. Neverlost's frontend is written in Java and was designed, built, and tested using Android Studio, in conjunction with AWS Cloud Services (see *E. Cloud*) as well as Bluetooth technology on the ESPs. The mobile application allows hikers to:

(1) Register with the NeverLost Application

(2) Connect to NeverLost Beacons along the trail without cell service

- Update their location
- Send an emergency SOS signal

(1) Register with the NeverLost Application

Prior to beginning their journey, a hiker must download the Android application and proceed to registration, which requires a simple button press. Upon registration, two unique random strings are generated by the mobile app: a HikerID that ties the hiker's device to their check-in data and a secret key that is used later to maintain system security. Both nonce values are then stored locally on the user's phone and uploaded to NeverLost cloud servers. In order to add these values to the cloud, the app invokes POST requests, directing the two parameters("key", "HikerID") to a php script running on our website, https://www.team08neverlost.com. Once registration is complete, the NeverLost mobile application provides users with a secure link to their check-in HikerLog webpage, which they can then share with friends and family. The registration process requires a connection to the internet enabling at home registration prior to beginning a hike.

(2) Connect to NeverLost Beacons along the trail

As a hiker begins their journey through remote hiker trails, cell service can become unreliable or completely unavailable. Neverlost's mobile app mitigates some risks associated with hiking by allowing users to update their location and contact emergency services when in Bluetooth range. When a hiker arrives at a NeverLost beacon along the trail, they are prompted to connect to the beacon via Bluetooth. When a user selects a beacon on their phone, the mobile app gets the associated device information and attempts to connect to the selected address associated with the beacon. Upon success, a communication socket is established between the beacon and the individual's device, allowing bidirectional communication between the devices. If a user is checking in, they input the beacon number they are at. Once the user presses the "Send Msg" button, the OnClickListener() retrieves the current date and time of the button press, as well as the locally stored HikerID and key. NeverLost concatenates the HikerID, the check-in date/time information, the beacon number, and pads the message with 5 zeros, as well as a success message that indicates successful encryption once received and decrypted at the base station. The mobile app then encrypts this concatenated string with the hiker's secret key and sends the encrypted message through the comm. Socket. If there is an emergency along the trail and they are sending an emergency signal, only the beacon number is sent through the communication socket.

E. Cloud

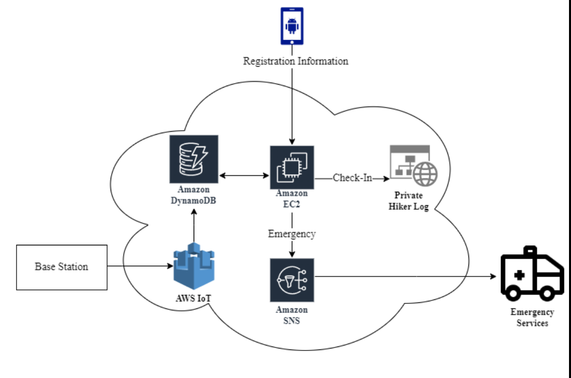


Figure 8: Cloud Computing Platform

The NeverLost system uses a number of Amazon Web Services as backend infrastructure. AWS IoT maintains NeverLost's connection to the outside world by linking the base station device to the cloud over the internet. Incoming signals from the base station are posted to MQTT topics which trigger cloud actions. The main cloud action is to enter incoming packets directly into an AWS DynamoDB table which maintains the list of encrypted messages.

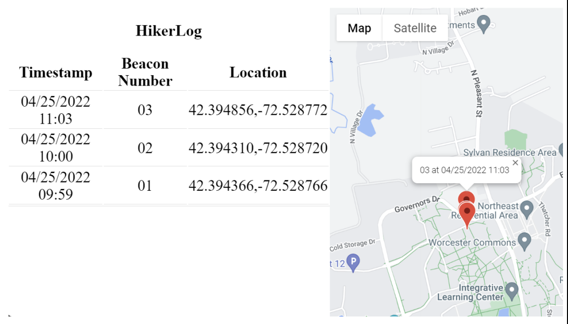
These messages are then read by an Apache server running on an AWS Elastic Compute Cloud (EC2) instance. The EC2 instance has four main functionalities:

1. Maintain Hiker credential records
2. Decrypt incoming messages
3. Serve and maintain a secure HikerLog website
4. Generate and send SMS messages to emergency services

To accomplish functionality 1the EC2 server uses a webpage running PHP scripts to read user credentials sent as POST requests from the NeverLost mobile application. Hiker IDs and symmetric keys are saved to a list of credentials on the server. Now that credentials are in the EC2 instance, encrypted messages are read into the server by downloading the contents of the encrypted messages DynamoDB table using the AWS CLI.

To accomplish functionality 2, a script on the EC2 instance attempts to decrypt each message using the list of downloaded keys. Upon reading a success message hidden within the successfully decrypted string, the message is processed to serve functionality 3.

To accomplish functionality 3, processed messages containing hiker location information are paired to the hiker's corresponding HTML file and are appended to their HikerLog webpage. See Figure 9 below.



HikerLog

Timestamp	Beacon Number	Location
04/25/2022 11:03	03	42.394856,-72.528772
04/25/2022 10:00	02	42.394310,-72.528720
04/25/2022 09:59	01	42.394366,-72.528766

Figure 9: HikerLog Webpage

To accomplish functionality 4, scripts on the EC2 instance detect emergency signals, pair embedded beacon numbers within emergency messages to their corresponding beacon coordinates. This message to emergency services over AWS Simple Notification Service.

To test all four functionalities, the AWS IoT MQTT test mechanism was used to simulate encrypted data transfer of signals sent from the beacon system. On a local machine, messages are encrypted with private keys and published to the MQTT test topic. Hiker ID and encryption key credentials were uploaded to the proper server files. Upon running server-side scripts, corresponding webpage entries were generated and viewed in the server's webpage directory. Successfully decrypted page updates were viewed on private web pages. To ensure proper and robust cloud functionality, random encrypted messages and random hiker keys were uploaded followed by real keys and real encrypted messages to ensure the system would not crash in the case of ill-formed or invalid entries.

F. Software Security

One of NeverLost's main purposes is to provide a sense of safety and security to hikers utilizing our system. Given that we are transmitting sensitive location information through long-range communication, confidence that this information is being sent and received securely is required. Our security approach was created and perfected through the guidance of security experts, notably Professors, Arman Pouraghily and Wayne Burleson.

Implementation of our security approach begins at Hiker Registration. When a hiker registers, the mobile application utilizes Android's internal random number generator to produce 2 nonces, or unique random numbers: a HikerID of 5 characters and SecretKey of 16 characters. Once these numbers are generated, they are automatically inputted into a DynamoDB table; we are able to assume that the connection to the internet is secure. Connection to the NeverLost website is also secure through our SSL (Secure Socket Layer) certification, indicated by the HTTPS:// prior to our URL. The SSL certification authenticates the website's identity and enables an encrypted connection, enabling our symmetric encryption and allowing us to share the secret key without the threat of eavesdropping. Once a hiker is registered, they are given a link to share with friends and family; this link will have sensitive location information once they embark on their journey so it is vital it is secure. To ensure this security, we utilize the one-way hash function – SHA-256 – hashing the user's secret key and creating a log at https://www.team08neverlost.com/*hash(secret_key)*.html

Once a user successfully registers, they can embark on the trail knowing their data is encrypted when sent from beacon to beacon through our Loras. We guarantee this security by utilizing the user's personal device to implement symmetric encryption. Both the base station and the individual's device have the given hiker's ID and encryption key; once a check-in occurs, the information is encrypted on the mobile application prior to it being sent over Bluetooth. This ensures that only unrecognizable ciphertext is being sent over insecure channels (i.e. Bluetooth, LoRa). Once the ciphertext ultimately arrives at the base station, the secret key is applied. Successful decryption is indicated by the "success" message embedded in the plaintext.

G. *Hardware Security*

An additional incentive to choose LoRa technology is dependability and security. LoRa uses IEEE 802.11 Frequency Hopping Spread Spectrum (FHSS) and Frequency Shift Keying (FSK) to avoid eavesdropping and packet sniffing. Essentially the transmitter "hops" packets between different available narrowband frequencies within a specified channel. Unless the adversary is aware of the pseudo-random hopping pattern or hopping algorithm, the message will be highly resistant to jamming. Many private and military applications implement these technologies as well.

III. The Refined Prototype

A. *Prototype Overview*

To initially prototype our system, we used an ESP32 development board as well as an RFM95W LoRa radio module. We constructed 2 beacons and a base station on breadboards while adding a charge controller, solar panel, and some LEDs. We based our custom PCB on this prototype as it was very similar.

B. *List of Hardware and Software*

Major hardware components include the ESP32WROOM module which is our microcontroller responsible for communication to the mobile application using BLE as well as communication to the cloud using WiFi. For long-range communication, our prototype uses the RFM95W which implements the LoRa communication protocol to allow us to communicate from beacon to beacon. Our prototype uses a 2.5Ah battery coupled with a 2W solar cell and the MCP73831/2 charge controller IC to create our power system. All of this is housed on our custom PCBs.

Major software components included interfacing and coding on various platforms in order to tie our system and the application together. Languages we used include: Python, HTML, CSS, php, and C++. In addition, we made use of libraries such as Cryptodome, hashlib, os, and RadioHead. A majority of the backend was handled by AWS. Some of the AWS features included; Elastic Compute Cloud, Simple Notification Service, DynamoDB, and Internet of Things.

C. *Custom Hardware*

To arrive at our final PCB design we started by prototyping on breadboards to get our desired functionalities. We have two slightly different designs: the beacon and base station, however, they share the same major hardware components. Once satisfied with the functionality of the prototype we moved to Altium where we recreated the schematic for our designs, here we found our specific parts. Our designs used mostly size 0603 SMD components and various through-hole components. From this schematic, we placed and routed components and traces in our PCB layout. Following the Altium tutorials created by Chris Caron, we were able to order our first revision. When it arrived we quickly realized that we forgot to repour our ground planes before creating fabrication outputs and decided the boards were not worth trying to fix.

We ordered a second revision which included the ground plane. Using the stencils and hot plate we were able to solder on all of the SMD components and make an attempt at testing. While trying to program the board we found that the reset and boot buttons were not set up correctly to program the microcontroller. Cutting a few traces and soldering an extra button onto our PCB allowed for successful programming of the ESP32. With this design, we could successfully send a single message. We forgot a connection from an IO pin on the RFM95W to the ESP32, this connection is responsible for sending an interrupt to the microcontroller when a transmission is finished. The green wire shown below created the connection and gave a working prototype PCB.

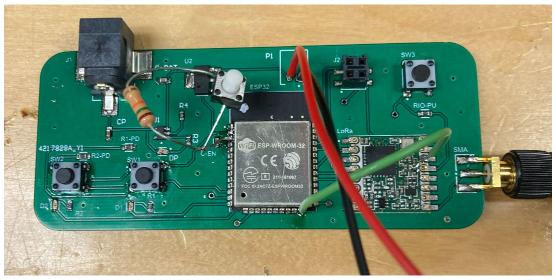


Figure 10: PCB Revision 2

This second revision was extremely important to our progression through learning PCB design. Once we had a PCB working in hand it became a lot more obvious why you should be so intentional about routing traces so that they are easy to follow and ensure reliability. With this learned, our third design was made with more intention, especially when designing the power and ground portions. We also designed the beacon and base station such that they use the exact same stencil even though they have slightly different designs. This decision was made in an attempt to save the budget as we saw possible. These designs can be seen in Figure 11 and Figure 12.

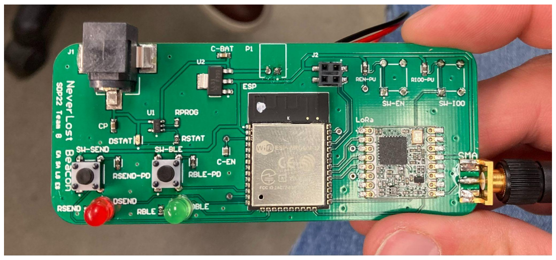


Figure 11: Beacon PCB Final Revision

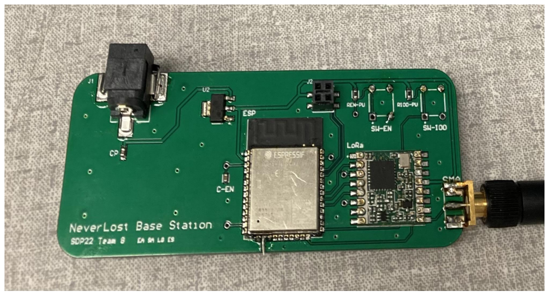


Figure 12: Base Station PCB Final Revision

IV. Conclusion

As a team, we have created a prototype system that securely transfers a hiker's personal data to a base station where it can be uploaded to the cloud or allow an injured hiker to get help from emergency services. We successfully met all of the system specifications outlined in our Preliminary Design Review. Our hardware has been successfully implemented onto a custom PCB. Our mobile application successfully encrypts and communicates sensitive data to our beacon system. The cloud hosted website successfully decrypts and displays accurate hiker location information on private web pages. Finally, emergency services are quickly alerted of all emergency requests.

Acknowledgment

We would like to thank our advisor, Professor Arbabi, for guiding us through this year and remaining patient and positive as we encountered new challenges. We would like to thank our evaluators, Professor Burleson and Pishro-Nik, for introducing new perspectives and providing helpful critiques to make sure NeverLost reached it's potential. We would also like to thank Professor Goeckel and Pouraghily for providing their expertise to validate and improve our various subsystems. Additionally, we greatly appreciated the help of Chris Caron and Wouter Schievink, they were helpful guides in navigating this integrative experience

References

[1] "Surveillance." *National Parks Service*, U.S. Department of the Interior, 15 June 2020, https://www.nps.gov/orgs/1336/data.htm.

[2] . (n.d.). *Meshtastic*. Meshtastic · Open Source hiking, pilot, skiing and secure GPS mesh communicator. Retrieved May 4, 2022, from https://meshtastic.org/

Appendix

A. Design Alternatives

We considered using technologies that would allow us to communicate with satellites rather than using LoRa for our long-range communication. We were quickly turned off of this because of how expensive the components were. Since this project is significantly limited by budget, we could not pursue these technologies.

Larger batteries and solar panels were also considered but similarly, due to budget constraints, we settled for the equipment listed above.

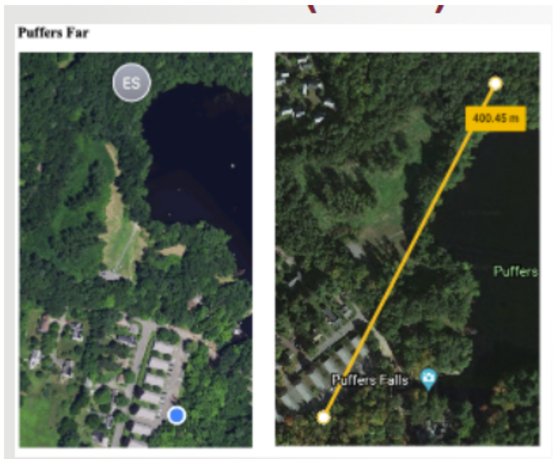
B. Technical Standards

Bluetooth Low Energy - IEEE 802.15.1
WiFi - IEEE 802.11
Frequency Hopping Spreading Spectrum - IEEE 802.11FHSS
License Free 915Mhz ISM Band
Frequency Shift Keying (FSK)

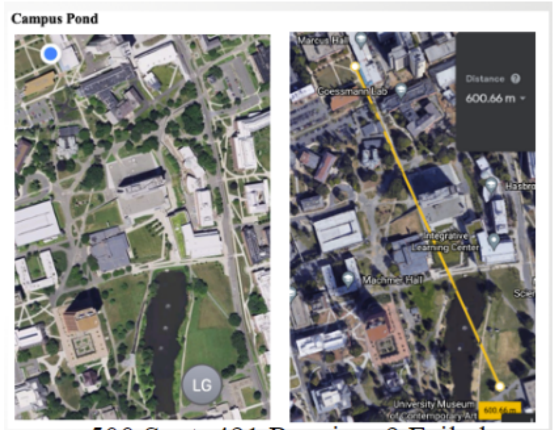
The ESP32 implemented Bluetooth and Wifi while the LoRa module used FHSS and FSK. On the software side, we implemented multiple AWS standards as well as some application and web development methodology

C. Testing Methods

To test the long-range communication we created a loop of 250 messages which were each the 64 bytes to represent the encrypted data that would actually be transmitted. Using this setup we went to various locations and then sent the loop with 500m between the transmitter and receiver. We recorded the number of successful transmissions and the number of reattempts needed to get a success. Images of testing locations are included below.



500 Sent, 476 Receive, 24 Failed
95.2% Success Rate
Avg Duration: 428.63ms



500 Sent, 491 Receive, 9 Failed
98.2% Success Rate
Avg Duration: 421.26ms

To test the power system we measured the total power consumed while running the loop for the range test. This was used to find the amount of power consumed per transmission so that we could plot this and find the amount of power our system would consume.

Testing our solar charging system consisted of making many measurements of output current at various locations and various weather conditions over the course of a day. We used these numbers to plot and find total power produced to run the system. Plot is included in Figure 5.

D. Project Expenditures

	Development	**Production**	
Parts		*Beacon*	*Base Station*
PCB	$42.30	$0.79	$0.62
LoRas	$123.12	$14.44	$14.44
ESP32s	$28.70	$3.60	$3.60
Solar Panel	$29.00	$29.00	NA
Batteries	$58.90	$14.95	NA
Pelican Case	$22.50	$22.50	NA
Misc	$372.59	$22.61	$26.36
TOTAL	$654.61	$107.89	$45.02

Our project was approved for a $227 budget increase to accommodate additional PCB revisions, website and enclosure costs.

E. Project Management

Our team is loosely divided into a software team consisting of Eric Anderson and Shelby Anderson and a hardware team consisting of Louis Gencarelli and Eric Sutherland. As needed, our entire team worked together to successfully integrate all parts of the system. Each team member has also stepped up in a role that may not be their strong suit, especially during the weeks of MDR and CDR to ensure a smooth presentation and demo. Team communication, handling of responsibilities, time management, and mutual support were strong.

F. Beyond the Classroom

As a team, we learned to work together and set specific deadlines and deliverables in order to design and prototype a system.

Eric Anderson: This class taught me a lot about how to use and connect a variety of AWS services. Additionally, I found value in making careful local tests of my programs before running them in the cloud in order to avoid more tedious debugging down the line. I learned how to break large projects into digestible and testable components to ensure the final system performs as intended.

Eric Sutherland: I believe my most important takeaway from this course was how to define a problem and its system specifications. This proved challenging as it was done in a bit of a rush at the beginning of the course before knowing the extent of the work needed to meet each specification. This course also taught me how to truly work in a team, finding each member's strengths and weaknesses and allowing each person to work to their strengths. Time management became extremely important, especially in the second half of the course when waiting for PCB revisions. Allowing extra time for prototyping and completing tests or debugging was crucial. If I was going to do something differently I would have had a better plan for the enclosure before choosing specific components for the PCB. Different style buttons and LED's could have made that design more seamless.

Shelby Anderson: Senior Design Project is the first class experience I had that somewhat emulated what a real engineering role will look like, working on the same project for a year, if not more. One of my biggest challenges, turned to a takeaway, was through debugging. When you are working with such a large, comprehensive system, different aspects are bound to break or run into an edge case that causes the whole system to crash. We initially found it very difficult, and often times frustrating pinpointing the root of many issues. However, debugging and problem solving so often, taught me to maintain composure and move slowly and with purpose. We began changing one thing at a time, working together, and found great success with this method. I also gained valuable knowledge and experience within Java and Application Development. Lastly, I really feel like I was able to apply the knowledge gained in my Security courses to not only formulate, but implement a unique security approach curated to our system. This project constantly had me thinking about how attackers could hurt our system, along with ways to combat that. This mindset is talked about within the security curriculum, but I really think I developed the mindset myself through this project, deliberately trying to think in the same way as an attacker would, in order to protect the integrity of our system.

Louis Gencarelli: Outside the classroom, this project has taught me how to choose certain components and modules based on strict system specifications. I've learnt to read and navigate datasheets, as well as program and wire various MCUs and sensors. A large portion of the project was spent integrating all of the subsystems and modules into one cohesive system. This was more challenging than I anticipated at first when the project was in its early phase. In addition, I feel like I've learned a lot more about PCBs, including how to design them and debug them when a component fails to function properly. Lastly, this course has greatly helped me enhance my debugging and programming abilities in general with embedded systems.

E. A. Author from Westborough, MA (e-mail: edanderson@umass.edu).
S. A. Author from Wrentham, MA (e-mail: shelbyanders@umass.edu).
L. A. Gencarelli from Westborough, MA (e-mail: lgencarelli@umass.edu).
E. S. Author from Charlton, MA (e-mail: epsutherland@umass.edu).