Covert Communications Device

Final Report for ECE 445, Senior Design, Spring 2021

By

Ahmad Abuisneineh (ahmada3)
Braeden Smith (braeden2)
Srivardhan Sajja (sajja3)

TA: Evan Widloski

5 May 2021

Project No. 9

Abstract

The primary purpose of this project is to provide a means for safe and effective communication in highly sensitive military and law-enforcement operations.

It is important for a two-way channel of communication to exist between the base and individuals in the field. Regular communication tools have output which can disrupt sensory awareness of the user thereby rendering them unsuitable for information transmission in these situations. A wearable device that relays messages that are only detectable by the user, and that will not affect their focus is the purpose of this project.

The final output of this project is a small device that can be installed on the human body in a convenient location, such as the upper arm or the outer thigh. The user should be able to access the device using their hand. The device will transmit information to the user through the cadence of haptic feedback (vibration patterns). The user will also be able to transmit information to all other devices in near real-time using a button on the device (pressing the button with whatever cadence is required, e.g. long-short-short). Team selected codes can be established to effectively transmit complex information or instructions.

Table of Contents

1. Introduction

During sensitive military and law enforcement operations like house raids and room clearing, quick intra-team communication is a necessity. Individuals need to be able to quickly receive and relay information, whether that is from an external command station or between members who are not within line-of-sight. An issue that makes typical radios unsuitable for this task is that individuals must speak out-loud to utilize them, removing the stealth advantage in these operations. Likewise, to receive information, the individual must either block some sensory awareness with in-ear radios or use a speaker which will produce external noise. Maintaining silence and the ability to accurately time and communicate actions is essential for the safety and success of these high-risk undertakings.

We created a small, wearable device to complement existing communication equipment. It produces vibrations in place of audible sound, triggered by another user pressing a button on a second device. Its compact design allows it to be kept in a pocket, or with a complementary armband, it may be strapped around the arm, thigh, or waist. The device's internals are relatively simple, divided into four subsystems, illustrated in figure 1.1.

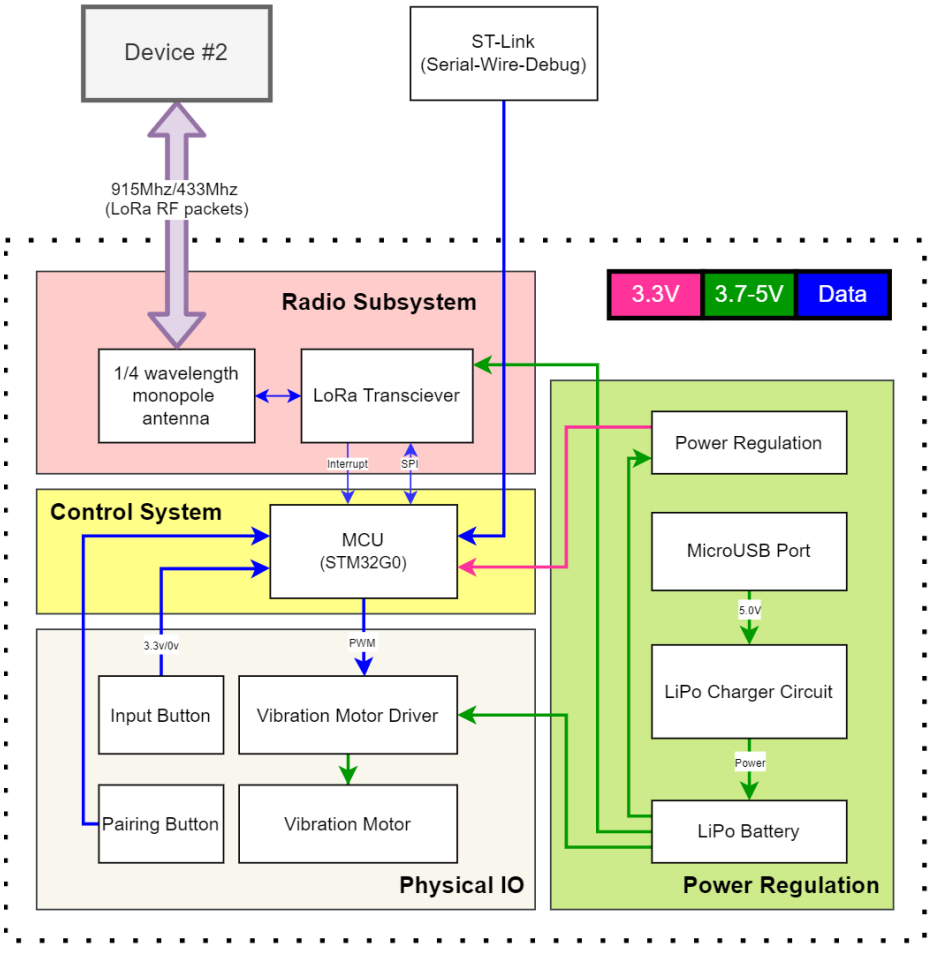

Figure 1.1. *High-level block diagram of the Covert Communications Device*

We divided our project into four major blocks:

The **physical I/O block** handles input from the pairing and communication buttons, and signals the microcontroller unit (MCU) when pressed. It also receives a PWM signal from the MCU, which gets amplified in the motor driver, in order to create the vibrations that the user feels.

The **power system** is a multipurpose unit responsible for charging the 3.7V Lithium-Ion Polymer (LiPo) battery through a micro-USB port, and for regulating the battery output to create a 3.3V power bus which is used as the microcontroller's power. The system bus can be powered from any combination of USB/battery power. For optimal performance, we require that the power system allows for device active usage for a minimum of 1 hour, and passive usage for a minimum of 8 hours.

The **radio block** is responsible for transmitting and receiving encrypted signals to and from other devices. We require our transmission range to be more than 1 km, which is made possible by configuring the modem with specific settings, and including a 7.8cm monopole antenna. We also require the performance of the radio subsystem to be good enough to ensure that a transmitted packet is received and processed by the receiver within 0.4s (300ms for transmission, and 100ms for SPI processing). This requirement is also made possible through the proper configuration of the radio settings.

And finally, the **control system block** is the brain of our device - it controls/responds to the radio system, handles user input, and sends/receives packets and controls the vibration motor. The software and hardware cryptographic functions occur in the MCU. During button presses it records 1.6 second segments of button data which is encrypted and relayed to other devices as vibration. It also is responsible for managing secret keys and pairing devices together.

2. Design Procedure and Details

2.1 Power System

For the power system, simplicity and ease of integration were key components of our design consideration. We decided from an early point that the system would have a battery component, and as part of that, we needed to integrate a capable charger and external power interface. We considered different battery chemistries and settled on 1 cell (1S) Lithium-Polymer batteries due to their high power density and variety of form-factors. We also selected a micro-USB port to charge the device externally, since this is a common interface for handheld devices and the pinout is very simple compared to more modern standards like USB-C. The charger and 3.3V regulator were selected for low ripple and a wide margin in specifications.

In order to select parts and complete the design, we considered approximate current draw for each subsystem and then provided a significant margin of error. We configured the charger to enforce a 500mA current limit to conform to the USB 2.0 specification [1]. These are equations governing the charger output current and our selected resistor.

$$I_{out} = K_{ISET}/R_{ISET},\ K_{ISET} = 510\ to\ 565A\Omega \qquad \text{(Eq 2.1.1)}$$

$$500mA\ = 537.5A\Omega/R_{ISET} \rightarrow\ R_{ISET} = 1075\Omega \approx 1K\Omega \qquad \text{(Eq 2.1.2)}$$

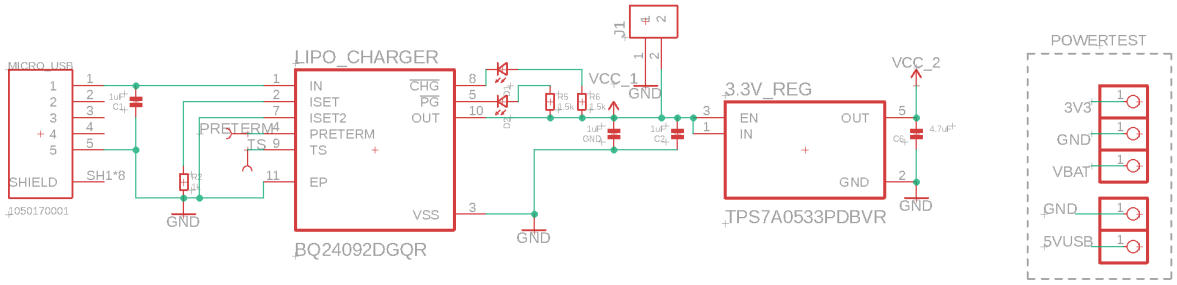

***Figure 2.1.2**. Eagle schematic of our minimal charging & power system*

System	Power Bus	Approximate current draw
Microcontroller	3.3V Rail	~10mA
Vibration motor	System Bus	~84mA
Radio system	System Bus	~120mA
LiPo charger	5V USB	~500mA

***Table 2.1.1.** Estimated current draw for various subsystems used for part and design selection*

We also selected a linear 3.3V, 200mA regulator to avoid the noise and EMI issues that may be present with switching power supplies. Since the regulator only had to drive the MCU, the power loss was minimal and deemed acceptable for our battery life requirements.

We selected a square style Lithium-Polymer battery with a capacity of 1200mA due to its form factor. It fit nicely against the PCB as illustrated in the physical design section and additionally provided more than enough battery capacity as demonstrated in the power system verification.

2.2 Microcontroller

From the start of the project we determined that we needed a powerful microcontroller for implementing the radio interface, software and cryptographic operations. Due to familiarity with industry success, we targeted chips with a 32 bit ARM architecture. We originally selected the STM32F0 line of processors, which would have allowed USB interface without an external clock requirement. Due to chip shortages [2], we were unable to acquire the correct packages. With some additional research & shifting requirements in the cryptography area of the project, we settled on the STM32G0 series, due to their low power consumption, speed, flash, RAM and most importantly the hardware accelerated cryptographic operations [3]. The large amount of GPIO, SPI and UARTs allowed great flexibility if project changes were to occur.

The complexity of this system mostly emerges in the software & cryptography section. Essentially, as seen in figure 2.2.1, the MCU acts as a hub to interconnect systems (radio, vibration motor PWM, button GPIO, SWD, pairing). We can also see two status LEDs that are also driven with low currents to provide some useful user information like mode and operation.

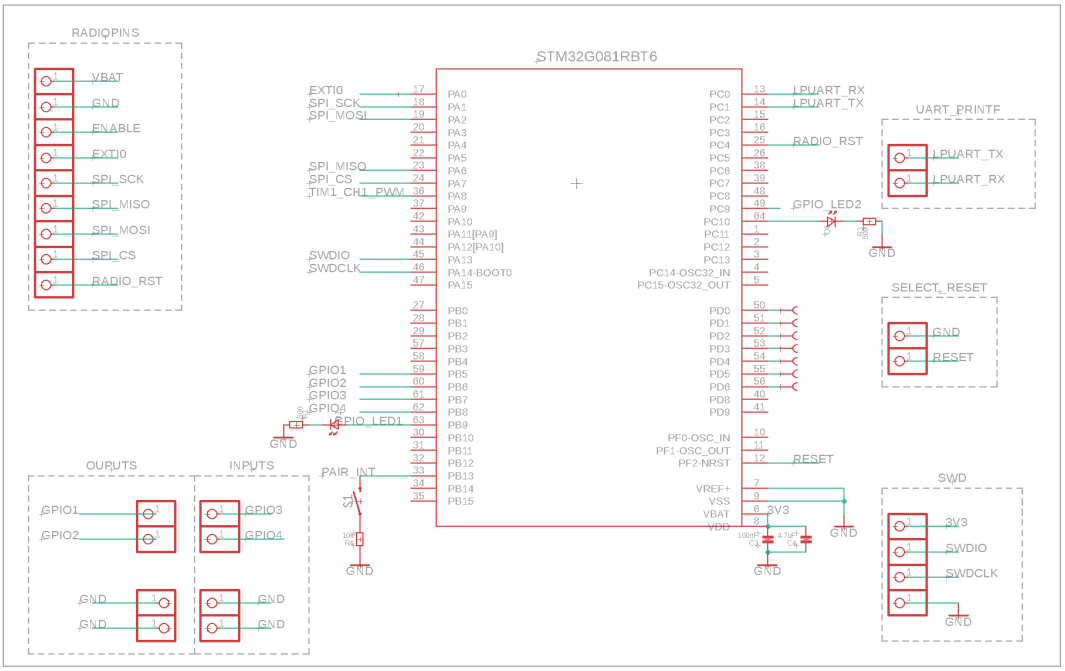

***Figure 2.2.1**. Microcontroller schematic with labeled signals, including some Physical IO components*

The software and cryptography section of the design discusses how the MCU hardware is utilized in more detail.

2.3 Radio System

For our project, we required a radio system that can transmit long distances with minimal latency. After some research, We decided to use the patented technology of LoRa, which stands for Long Range. LoRa enables long distance communication upto 2 km, with non-directional monopole antennas, which fits the use case of this project. Furthermore, this technology is highly reliable for low bitrate transmission, which is ideal for our use case. Since LoRa is widely popular and well tested, we were able to rely on the multitude of documentation and resources that are available.

We chose the RFM9x breakout board equipped with an RFM95 LoRa radio chip for use in our project.

We soldered header pins and wire antennae into the LoRa radio breakout boards for testing with a breadboard. For transmitting at 915 MHz, our antenna needed to be exactly 3in or 7.8cm.

We went beyond the scope of this project to design a fully equipped library for the radio. There are two steps for receiving and transmitting data. First, the device is reset using the RESET GPIO pin. Then, all required registers, including power, operation mode, modem configurations and frequency must be initialized with appropriate values. This is the initialization step, which happens as soon as the device is powered on. After this, based on whether the device is transmitting or receiving, the FIFO register must be written to or read from. The payload cannot be transmitted directly. It is loaded into a 16-bit data buffer (for eventual implementation of AES-128 encryption), which has a preamble, a header and CRC, as shown in figure 2.3.1. For the purpose of establishing a connection and testing basic information transmission, the explicit mode is not used, and thus our final data buffer will contain a preamble and the payload.

The factors that can be configured manually, and directly influence the data rate (DR) are: bandwidth (BW), spreading factor (SF) and coding rate (CR). They are related to each other by the following equation:

$$DR = SF \cdot \frac{BW}{2^{SF}} \cdot CR \qquad \text{(Eq 2.3.1)}$$

Configuration can be done on the radio by setting the registers 0x1D (RegModemConfig1), 0x1E (RegModemConfig2), and 0x26 (RegModemConfig3) [1]. We only used some bits of these 3 registers: 0x1D[7:4] for setting bandwidth, 0x1D[3:1] for setting coding rate, and 0x1E[7:4] for setting the spreading factor.

We tested different values for each of these variables, sending a packet of size 16 bytes. The observed latencies in transmission, along with the different modem configurations are given in table 2.3.1. Signal bandwidth options are discrete, and are not fully under our control, and spreading factors are expressed as base-2 logarithms [4]. Thus, based on our observations, we have chosen the following values to be used for developing our device prototype: 250kHz for the bandwidth, 9 for the spreading factor, and a

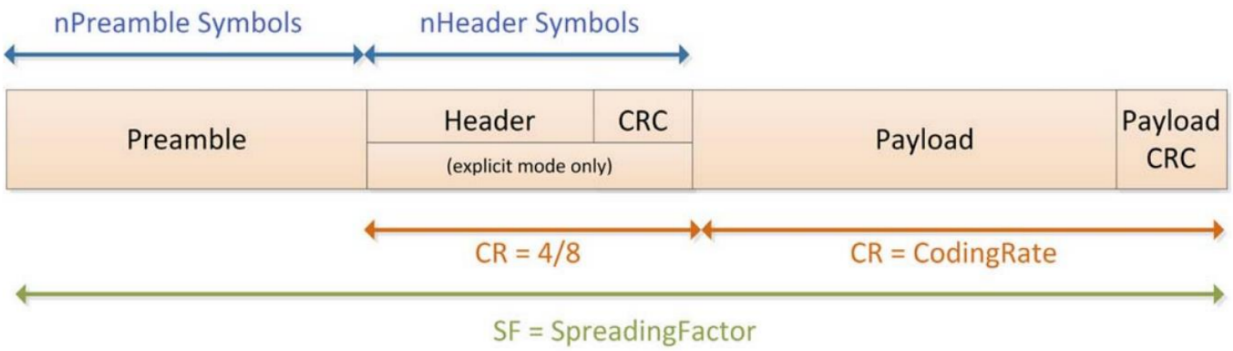

***Figure 2.3.1**. LoRa packet structure, which can be transmitted by RFM95 after required values are set*

Bandwidth	Spread Factor	Coding Rate	Data Rate	Theoretical Latency	Actual Latency during testing
500 kHz	11	4:5	2.148 kbps	59.59 ms	~165 ms
31.25 kHz	6	4:5	2.344 kbps	54.60 ms	~42 ms
250 kHz	9	4:5	3.516 kbps	36.40 ms	~93 ms
500 kHz	6	4:5	37.500 kbps	3.41 ms	~4ms
7.8 kHz	12	4:8	11.000 kbps	11.63 ms	~23 ms

***Table 2.3.1**. Tested values for Average Latency of transmission based on different modem configurations*

ratio of 4:5 for the coding rate. Choosing these values will give us a good balance between maximum range of transmission and data rate while also keeping time required for transmission of a single message of length 16 bytes with SPI under 100ms.

2.4 Physical I/O & Physical Design

We wanted the device to only broadcast a message when the user intended. For that, we went with a regular, surface mount push button. We believed that it had just the right springiness, such that the button would not trigger accidentally. However, we soon realized that the surface area is small, and we would need something bigger. Instead of switching out the button, we 3D printed a plate, which would sit on top of the button, thereby increasing the clickable surface area of the button, making it easier to press. Since the plate was held on by a casing, the casing ensured that the plate would not be hit accidentally. In the future, we might opt for a bigger button, so that we do not need to 3D print any extra pieces, reducing the degree of error in our project.

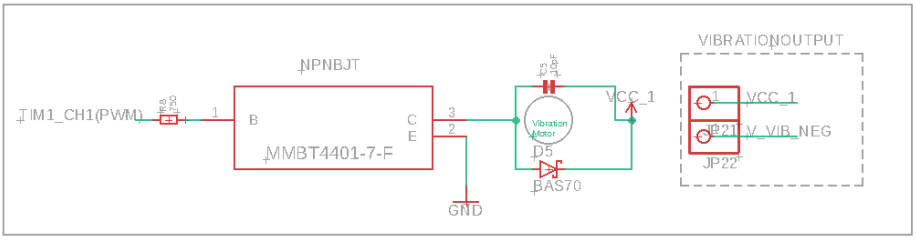

***Figure 2.4.1**. Schematic for vibration motor and related circuitry*

For the output side of the I/O, we needed to drive a vibration motor. We opted to use a NPN transistor as a motor controller. Refer to figure 2.4.1 for a schematic of the motor controller circuit. The signal sent to the transistor/motor circuit was in the form of a PWM wave, which let us adjust the intensity of the vibration motor from the software side. For example- we needed to optimize the motor such that the vibrations could be felt through the casing (higher duty cycle), while keeping it so that vibration motor did not become audible (lower duty cycle). We found a middle ground experimentally, but we do anticipate changing the value based on the final device's casing unit.

The physical I/O and casing design creation went almost hand-in-hand. Since the communication button must be easily accessible to the user, we had to put it on the outside of the case. Similarly, since the vibration motor must be felt by the user, it had to have a special socket in the case, bringing it closer to the user's skin. Due to these criteria, the physical design of the device was designed simultaneously with the design of the I/O unit.

To develop these two units simultaneously, we initially created an AutoCAD file, and imported all of our device components as simplified 3D geometrical shapes (cubes, cylinders, etc). This allowed us to experiment with component placement for the optimal space efficiency. Figure 2.4.2 shows the final decision for object placement in our design.

Once we plotted where each component will go relative to each other, we then took the maximum measurement in each dimension, and created a simple rectangular prism in TinkerCAD to represent the case. In this simple prism, we added necessary holes and mounts so that wires could be thread, the USB port could be accessed, the light pipes could reach the LEDs, and the PCB could be mounted. We then smoothed out the edges of this rectangular prism, and configured it so that it could open and close. This led us to the prototype of the casing. Figure 2.4.2 shows the TinkerCAD .obj file.

The circular hole in the left case component of figure 2.4.2 is the socket in which the vibration motor sits. Since the case is 3mm thick, feeling the vibration motor proved to be a difficult task after 3D printing. We created a small 10mm socket, in which the case is 2mm thick. This improved the ability of the vibration motor to be felt through the casing. However, since the casing is PLA plastic, it acted as a speaker when the vibration motor ran- essentially, the casing amplified the vibrations into sound. For our final design, we would like to add a notch in the casing which will create a tight and secure fit, so that the vibration motor cannot move around and create excessive noise. Not only does the noise lose some sense of covertness, it also takes away from the total energy, which could be vibrations.

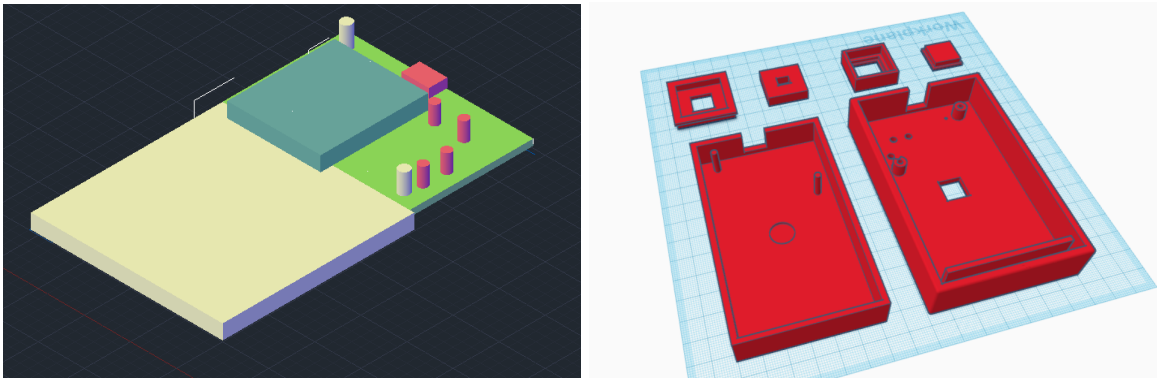

***Figure 2.4.2**. Circuit components designed in AutoCAD (left) and casing protype in Tinkercad (right)*

2.5 Other Design Considerations: Software

We had a couple of guiding principles during the design of the software. Firstly we needed it to be a really robust state machine: reliability was key, the state tracking needed to be logical and the jumps between states needed to be simple (figure 2.5.3). Secondly, we wanted the software to be wholly asynchronous, in our case, that means making every action dependent on interrupts. These interrupts can be external (radio, buttons) or internal and periodic (button recording, vibration playback). This gave our team lots of flexibility for adding additional features and not burdening the performance of the MCU. And finally, we wanted to lean heavily on the hardware when possible, making crypto operations, random number generation and storage all faster (figure 2.5.2).

```
#define MASTER_DEVICE   1
#define DEVICE_ID       1
#define RESET           0
#define NEW_SEQ         0
```

***Figure 2.5.1**. Static defines to configure software flow*

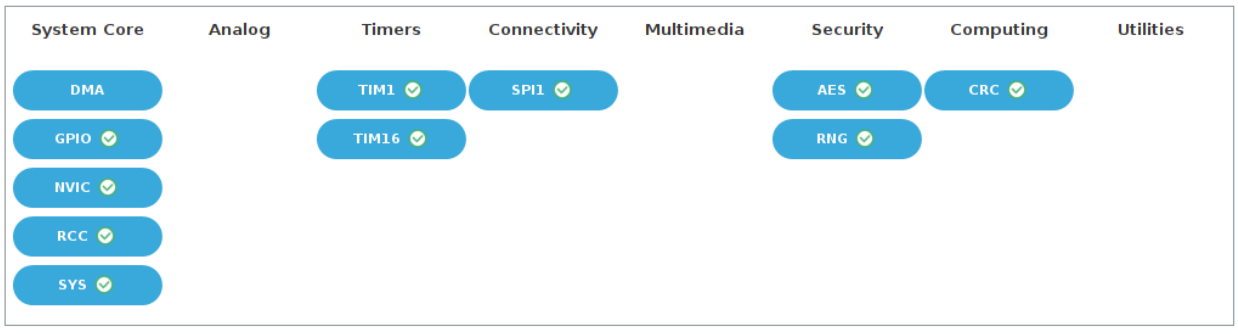

***Figure 2.5.2**. Hardware features used during software development*

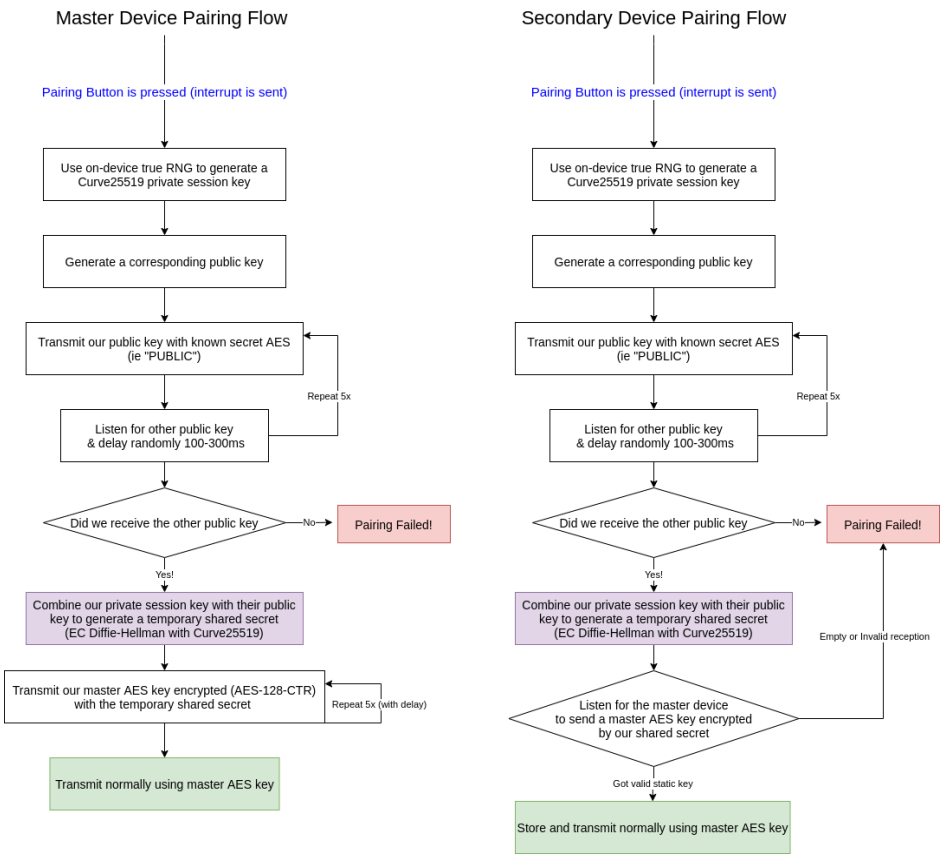

Figure 2.5.3. *Pairing state machine depending on device type*

2.5.1 First Boot & Device Assignment

We start out prior to the first device boot: a singular device within the fleet is assigned a “master” status using static “#define” statements as shown in figure 2.5.1. The software on each device is identical but behaves slightly differently based on this status. Each secondary device must be paired individually with the master device prior to normal communication.

At first boot, every device checks whether it has an assigned or generated “master key”. If it is lacking this “master key”, it uses the true random-number-generator on board the microcontroller to generate and store a new 16-byte key in flash. It also generates a 31 bit sequence number (or recalls it from flash) and increments its sequence number blindly by 2000. This is under the safe assumption no more than 2000 packets are sent between reboots and that less than 1000000 reboots will happen before other devices get power-cycled.

2.5.2 Recording & Playback

Recording and playing back the correct cadence of button presses is an essential part of this project. Without accurate recall, information may be lost unnecessarily. We went through several different approaches to this problem.

We first considered sending a packet with a binary state ie. packet = button is on. This approach had several issues, total latency had to be very small (because it defined the minimum depress time) and it

leaked a lot of information accidentally, because an attacker could just view the number and cadence of packets.

We decided instead to have each packet sent from the radio system embedded a section of time. In particular, we selected a 1.6 second interval from the first time a button is depressed. We sampled the button state every 25ms, embedding the state into a single bit. This inherently prevented debounce issues with the button and allowed for less LoRa transmissions while only costing a minor increase in total latency.

A button press would trigger an interrupt handler in software, which would enable a periodic interrupt to begin firing every 25ms, sampling the button, until 64 samples had been captured. At the end of the sampling, the state machine would signal that the packet containing that data should be encrypted and sent. If the button was still depressed after that interval, recording would continue another 1.6 second capture.

In a similar fashion to recording, when a valid packet holding vibration data was received, the software enables the same periodic interrupt and begins replaying the vibration data. Actively changing the PWM signal to the vibration motor every 25 milliseconds according to the 64 data bits in the packet

2.6 Other Design Considerations: Cryptography

For the cryptographic part of the software, we considered our “threat model”, what attackers we could imagine and potentially stop. We settled on preventing replay attacks and passive man-in-the-middle attackers. In addition to our threat model, we had to determine which cryptographic primitives were essential for pairing and transmission. We settled on a symmetric block-cipher and some form of Diffie-Hellman key exchange. This was important for pairing and secure communication over the radio system. Following industry and government standards we selected AES-128 (Advanced Encryption Standard) as our symmetric block cipher and Curve25519 as our Diffie-Hellman (Public-key key exchange) function. These are both very fast on embedded hardware in comparison to alternatives (like Rivest–Shamir–Adleman [RSA]) and are the backbone to modern day internet communications.

2.6.1 Asymmetric Pairing

The pairing software flow is illustrated in Fig. 2.6.1. All devices have an internal pairing button and only two devices can be in pairing mode at a given moment: the sole master and secondary of choice. To begin the pairing process, both buttons must be pressed within a close time interval. From there, an interrupt occurs on the MCU, jumping it to the pairing code. Each device generates a random private key and then a corresponding public key.

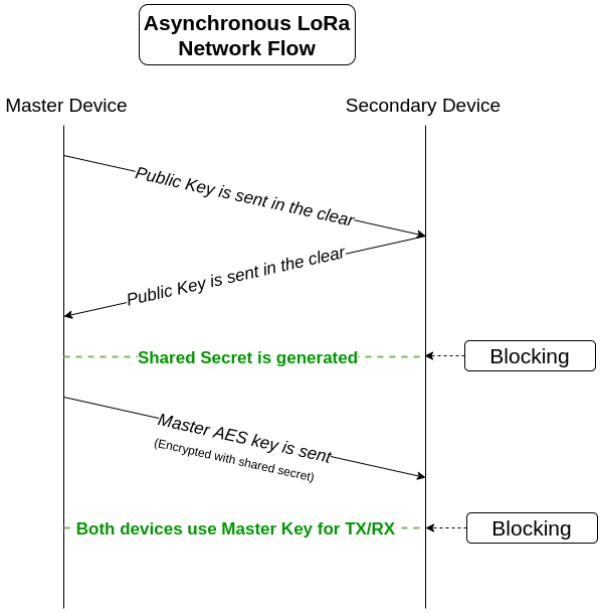

***Figure 2.6.1**. Asymmetric key exchange network flow*

They then exchange their public keys “in the clear” by transmitting five times with random delays, as illustrated in Figure 8. If both devices successfully receive the other’s public key, they can compute an identical “shared secret” using their own private key and the other’s public key. Now that they both have the same “shared secret”, a truncated version of that is used for the AES-128 symmetric encryption. The master device will now send its true “master key” by encrypting it with the shared secret key. The secondary device now replaces its internal “master key” with the one it just decrypted.

2.6.2 Standard Encrypted Communication

For standard AES (a symmetric encryption mechanism) usage, we simply use the “master key” stored in memory to encrypt and decrypt messages. This is accelerated by the on-board AES hardware that is built into the device. AES is a block cipher with a minimum block size of 128 bits, therefore we must create and transmit multiples of 128 bits of content in every radio packet. Since our actual information requirements are very small, we can assume we will send only one (padded) block during each transmission.

Normal communications (of vibration state) use AES-128, a block cipher with a key size of 128 bits. Each packet is embedded with an incremented sequence number, a deviceID, and a packet type preamble. A recipient will only accept messages with a sequence greater than the last one that was received. This prevents what is called a “replay-attack”, where an attacker could directly resend a valid packet without needing to know the contents.

3. Design Verification

3.1 Power System

We specified several simple verifications for the power system, to validate safety features and the ability to supply the correct amount of power to the subsystems. The power system passed our tests without issue, and performed admirably during functional testing. The testing details can be found in appendix A.

We had one verification, to test the battery capacity by discharging it with an external charger, that we were unable to complete due to protection circuitry attached to the battery cell. Since this verification was intended to validate our battery life "high level requirements", we decided to test the system draw instead. With the current draw testing and using even the most liberal estimates of system power consumption, we calculated the battery life with our selected 1200mAh LiPo as follows:

$$\frac{1200mAh}{22mA} = 54.5\ hours\ (RX\ only)\ >>\ 8\ hour\ requirement \quad \text{(Eq 3.1.1)}$$

$$\frac{1200mAh}{220mA} = 5.45\ hours\ (TX\ \&\ Max\ Vibration)\ >>\ 1\ hour\ requiremer \quad \text{(Eq 3.1.2)}$$

Since these estimates far and away exceeded our high-level requirements, we were able to safely disregard the troublesome capacity test. Even 25% of advertised battery capacity would handedly exceed our minimum battery requirements.

3.2 Microcontroller

The microcontroller verifications centered around meeting software minimums for speed -- especially to ensure that we keep the communication & interface latency as short as possible. The cryptographic operations were benchmarked many times to ensure the average time was less than the minimums we considered. All the benchmarking resulted in times that far exceeded the minimums, at worst case, the microcontroller was about 3x faster (shown in table 3.2.1).

As mentioned in the design section, the software is primarily interrupt based. The buttons and radio system both send interrupts to the microcontroller and it is important to handle them as fast as possible. We validated the interrupt handler timing to be <5uS, once again, far exceeding our expectations (figure 3.2.1).

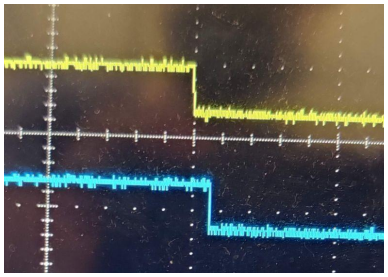

***Figure 3.2.1**. 20µS interval capture of Interrupt timing. Yellow = Interrupt, blue = Interrupt Handler.*

Table 3.2.1. Operations and Durations

Operation type	Total operation time	Average time per single operation
Curve25519 Public key generation Repeated 100 times.	31.377 seconds	0.313 seconds
Curve25519 secret exchange Repeated 100 times.	31.406 seconds	0.314 seconds
AES CTR Encryption of 128 bits (1 block) Repeated 10000 times.	189 milliseconds	18.9 microseconds
AES CTR Decryption of 128 bits (1 block) Repeated 10000 times.	191 milliseconds	19.1 microseconds

3.3 Radio System

The radio unit has two requirements. They are related to transmission range, and transmission distance of the modem configured RFM95 LoRa radio chip.

For the first requirement, we had to make sure that any packet sent from the first device must be received by a second device and processed by SPI within 100ms. For this verification, we set up our apparatus similar to the procedure described in the design document. We connected the first RFM95 radio to the STM32G0 chip, and loaded it with code for transmitting once every 100ms, with the following configurations: 50kHz for bandwidth, 9 for spreading factor, and a ratio of 4:5 for coding rate. Then, we connected the second RFM95 radio to the Arduino code and loaded it with the code for receiving packets of information. An additional verification done was to ensure that after 1.6s of recording time, the recorded packet is received by the second device before the 2s mark. We were able to verify that the packet is in fact being sent and processed by the SPI in around ~93ms.

For the second requirement, we had to make sure that upon sending a packet from the first device, the packet will be received by the second device if the Euclidean distance between the two devices is less than or equal to 1km. There should be no obstructions between the two devices during verification.

We tested this by setting up two different LoRa radio breakout boards. We connected the first RFM95 radio to an Arduino, and loaded it with working code for receiving 16-byte packets of data. Then, we connected the second RFM95 radio to the STM32G0 development board, and set it to transmit once every 200ms. The Arduino is connected to an LED which will blink every time a packet is received. Both radios are set up with the following modem configurations: 50kHz for bandwidth, 9 for spreading factor, and a ratio of 4:5 for coding rate.

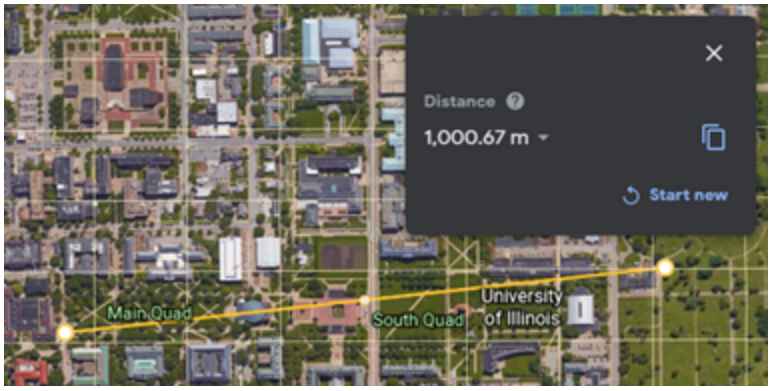

***Figure 3.3.1**. Two LoRa radios placed at points A and B, distanced 1km apart*

We placed the development board at a location on the University of Illinois main quad, and started walking away from it, while ensuring that the Arduino's LED is blinking at a constant rate. I was able to get a strong and reliable signal for the full 1000m. Figure 3.3.1 shows the route taken to verify this requirement.

3.4 Physical I/O

The physical I/O verifications involved preventing interference in the circuit by reducing the back-emf created by the vibration motor, maintaining stealth by reducing the sound of vibrations, and increasing strength of the vibrations, so that the user can feel them through the casing.

To reduce the back-emf, we placed a schottky diode and a capacitor in the vibration motor circuit. Figure 3.4.1 demonstrates the reduction in peak-to-peak voltage after the incorporation of the diode and capacitor. The top figure is the oscilloscope reading before any protection. The peak-to-peak voltage here is 9.1V. The bottom figure is the oscilloscope after using the diode and capacitor protection circuit. There is a significant reduction in peak-to-peak voltage, falling to 5.6V. This created a significant reduction in noise, by nearly 50%.

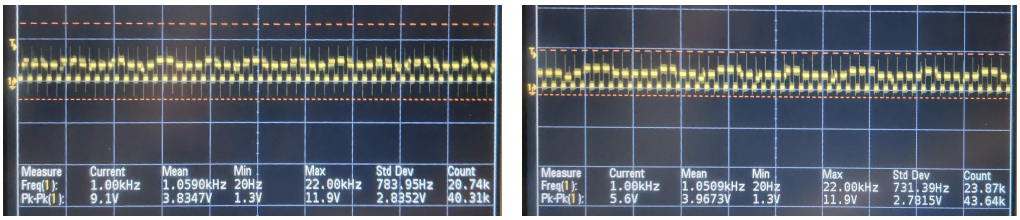

***Figure 3.4.1**. Reduction in Pk-Pk voltage before (left), and after protection circuitry (right)*

4. Costs

4.1 Parts

The total price of all the device components, and the price breakdown for each, for prototyping is detailed in table 4.1.1. The cost of parts for each device is approximately $37-40, ~$20 of which is from the radio system and another $10 for the battery. Purchased in bulk, pure PCB BOM price could be brought to ~$5-6.

We also need to factor in the cost of the casing- each case, with about 30% fill density, uses about 72 grams of PLA. Currently, a single 1kg roll of PLA sells for ~$20 on Amazon, so a single case costs about $1.44 to 3D print (not including the cost of electricity to run the printer). If we change the fill density to make the case sturdier or weaker, the cost of printing may increase or decrease.

***Table 4.1.1**. Parts List, Quantities, and Costs*

Part No	Supplier	Description	Price Per Unit	Quantity	Net Price	Bulk Pricing
NUCLEO-G071RB	Digikey	*Discovery Board*	$10.99	1	$10.99	n.a.
STM32G081RBT6	Mouser	*Microcontroller*	$3.85	5	$19.25	$2.38
ST-LINK/V2	Digikey	*ST-Link*	$22.60	1	$22.60	n.a.
455-1704-ND	JST Sales America	*JST Connector*	$0.17	5	$0.85	$0.06
1201	Digikey	*Vibration Motor*	$1.95	3	$5.85	$1.56
1010	Adafruit Industries	*Buttons*	$5.95 per 15	1	$5.95 for 15	$4.76 per 15
SMMBT3904LT3G	Microchip Technology	*Motor Controller*	$0.50	2	$1.00	$0.04
258	Adafruit Industries	*Battery*	$9.95	2	$19.90	$8.86
296-41204-6-ND	Texas Instruments	*LiPo Charger Chip*	$1.36	2	$2.48	$0.65
MMBD101LT1GOSDKR-ND	ON Semiconductor	*Schottky diode*	$0.29	5	$1.45	$0.05
TPS7A05	Texas Instruments	*3.3V LDO*	$0.62	4	$2.34	$0.55
WM1399DKR-ND	Molex	*MicroUSB Female port*	$0.83	5	$4.15	$0.35
1528-1667-ND	Adafruit Industries	*Transceiver*	$19.95	2	$39.90	$19.95
n.a.	Digikey	*Misc. Resistors + Capacitors + LEDs*	$10.00	1	$10.00	n.a.
4269	Adafruit Industries	*Antenna*	$0.95	2	$1.90	$0.76
Total Price:					**$138.27**	

4.2 Labor

A typical salary for a UIUC BS Computer Engineering graduate is $96,992/yr, or about $50/hr. On average, we each worked on the different components of the device for about 10 hours per week, which gives an estimated cost of labor of about $12,500 for each teammate, or $37,500 overall.

$$\frac{\$50}{hr} * 10\frac{hrs}{week} * 10\,weeks * 2.5 = \$12,500 \quad \text{(Eq 4.2.1)}$$

Overall, depending on the profit level that we are trying to reach, and the cost of labor, the retail price would be in the $50 to $100 range. We believe that this is a reasonable price for such a device.

5. Conclusion

5.1 Accomplishments

By the end of this semester, we produced three working prototypes of our device. The range, latency, and battery life of our device was on par with our goals -- performing better than we wanted in the first place. Overall, we believe this was a very successful semester, and a successful project as well.

5.2 Uncertainties

We do not have any major uncertainties with our project. However, we are considering new use cases and scenarios for our device. For example, it could be used as a portable nurse pager in hospital rooms. We are trying to expand our horizons and explore new scenarios in which this device could make a difference.

5.3 Ethical considerations

Lithium-ion batteries can be a fire hazard if punctured or misused. We went with a lithium-ion polymer battery, because they have been shown experimentally to have greater durability than traditional batteries. We also selected batteries with built-in protection circuitry, and chose a charger that was reputable, configured with a 500mA current limit, to maximize safety for the consumer.

Radio frequency transmissions can be hazardous or illegal - that is why we selected a radio system that operates in a license-free ISM band (at 915MHz). This band is low-energy and cannot be of any harm to humans or the environment. We also selected a module that is FCC pre-approved for integration [4].

As mentioned above, we build our software with an emphasis on security and privacy, integrating 128-bit AES encryption within every radio broadcast, complying with IEEE ethics standard #1 to the best of our abilities [5].

5.4 Future work

Before we have a final product launched, we have a list of goals we would like to achieve.

We would like to include an on/off switch for the device. The current setup only allows for power on/off by connecting and disconnecting the battery. Obviously, a switch would be more efficient for a user.

We would also like to use the USB port for reprogramming the device, instead of only charging. Thus, if we ever need to make a software update for already-deployed devices, the devices could be updated via a computer. The current setup makes use of the jumper pins on the PCB, which a user would not be able to configure without additional equipment.

We also plan on including a battery monitoring system, so a user can see how much charge is remaining in the battery. This would be a useful feature for the user, so that they know if the battery needs to be recharged soon.

Some changes may be necessary to improve the device. We might experiment with new button types, a internal PCB antenna (over the existing monopole), a new method to attach the vibration motor (socket, instead of tape), and maybe a new USB interface (USB-C over micro-USB).

And finally, there are some things that need to be removed before we launch a final product. We would like to remove the radio transceiver's breakout board, and solder the transceiver module directly to the PCB, to save room in the casing. We also want to shrink the casing, removing empty space, to create a more compact design overall.

References

[1] *What are the Maximum Power Output and Data Transfer Rates for the USB Standards?*, 01-Jul-2020. [Online]. Available: https://resources.pcb.cadence.com/blog/2020-what-are-the-maximum-power-output-and-data-transfer-rates-for-the-usb-standards#:~:text=In%20general%2C%20the%20specifications%20for,volts%20from%20each%20standard%20output. [Accessed: 06-May-2021].

[2] G. D. Vynck, "What you need to know about the global chip shortage," *The Washington Post*, 01-Mar-2021. [Online]. Available: https://www.washingtonpost.com/technology/2021/03/01/computer-chip-shortage-explainer-qa/. [Accessed: 06-May-2021].

[3] "STM32G0 Series," *STMicroelectronics*. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32g0-series.html. [Accessed: 06-May-2021].

[4] "3072," *DigiKey*. [Online]. Available: https://www.digikey.com/en/products/detail/adafruit-industries-llc/3072/6005357?s=N4IgjCBcoLQBxVAYygMwIYBsDOBTANCAPZQDa4ArAEwIC6AvvYVWSAMwAMA7Cw0A. [Accessed: 06-May-2021].

[5] "IEEE Code of Ethics," *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 06-May-2021].

Appendix A Requirements and Verifications Table

Requirements	Verifications	Verified?
Control Unit 1. AES-128-CTR encryption and decryption on 16-byte blocks each must occur in <10ms	A. Include the official STM32 cryptographic library + enable hardware AES for MCU in STM32 CubeIDE B. Write to perform AES-128-CTR encryption and decryption on a random piece of stack memory 10,000 times C. Record the times for both operations relative to the clock speed D. Use the serial wire debug breakpoints to view time/clock cycles elapsed for the operations	Yes
Control Unit 2. Public key generation and key exchange using Curve25519 library in STM32 crypto must occur in <1 second taken separately	A. Include the official STM32 cryptographic library B. Write a two programs to time and do the following 100x: a. Generate random bytes for private key and execute public key generation b. Generate a shared secret from the created secret and a hardcoded public key C. Take the average times recorded by these programs	Yes
Control Unit 3. Ensure the STM32 can receive external interrupts and context switches to respond to them in <50µS.	A. Get two oscilloscope probes and place the oscilloscope into its highest capture rate with a rising edge trigger B. Configure the STM32 to enable rising edge external interrupt on an exposed pin, and internally pulled-down output on the other C. Write code that, when it receives an interrupt, it drives the other pin high D. Place scope on both pins, pull the interrupt high E. Measure the latency between the rising edges on the oscilloscope	Yes
Power Supply Unit 4. LiPo Battery: True capacity, from being fully charged (4.2V) to discharged (3.7V) should be ≥800mAh	A. Use a lithium polymer charger with constant current charge and discharge capabilities (ISDT Q6) B. Charge lipo to full (4.2V) -- discharge it at 500mA with a cutoff voltage of 3.65V, record displayed capacity. C. Now charge LiPo to full (4.2V) again at 500mA rate, record the displayed capacity	No (See power system details above)

Power Supply Unit 5. LiPo Charger: LiPo charger should have protection against accidental shorts of exposed JST pins	A. Use a current limited mode on a lab power supply, set 5V, 600mA limit B. Drive the USB power pins via exposed breakout on assembled PCB C. Measure the output pin of the LiPo charger with an oscilloscope D. Short the JST connector pins E. Ensure the output pin drops to zero until power is cycled and that the 600mA limit on the power supply is not hit	Yes
Power Supply Unit 6. LiPo Charger: LiPo charger should be capable of providing 300mA throughout its output voltage range (3.65-4.25V), while being powered by a 5.0±0.1V 500mA current limited source	A. Prior to installation. A benchtop power supply should be set to 5.0V with a 500mA current cutoff B. The LiPo charger should be configured appropriately with passive components to set the proper mode and charge rates C. A standard discharged (3.7V) LiPo battery should be plugged into the output pins D. A multimeter should be placed in series between the output pin & the battery E. Check if the current output exceeds 300mA F. Otherwise, add an artificial load via a resistor to ground in parallel with the battery. Continue to lower the resistance until the multimeter reads 300mA G. At 300mA, use an oscilloscope or multimeter to measure the voltage is above 3.65V	Yes
Power Supply Unit 7. 3.3V Regulator: Regulator should be capable of supplying 3.3±0.1V at 50mA from an input of 3.7V at 400mA	A. Prior to installing the voltage regulator, place it on a breadboard, use a benchtop power supply to provide 3.7V as input B. Place a resistor of 50-66 Ohms between 3.3V output and ground C. Use a multimeter to ensure the 3.3V output does not fall below 3.15V under load.	Yes
Radio Unit 8. Upon sending a packet on one device, it should be received on the secondary device and sent via SPI in <100ms	A. Use two 3.3V/5V logic level devices (Nucleo development board + Arduino are both options) and have each connect to a transceiver with SPI B. Setup both devices with LoRa libraries, use selected modem configuration, set one of them to TX every 200ms C. Have the transmitting device drive a GPIO pin high when it finishes transmitting D. Have the receiving device drive a GPIO pin high as soon as it receives a packet via SPI E. Attach an oscilloscope to each GPIO pin and set up single-shot capture	Yes

	F. Compare the time differences between each pin's rising edge over a sample size of five	
Radio Unit 9. Upon sending a packet on one device, it should be received on the secondary device if paired and euclidean distance between devices is less than or equal to 1 km with no obstructions between them	A. Use two 3.3V/5V logic level devices (Nucleo development board + Arduino are both options) and have each connect to a transceiver with SPI (at maximum output power) B. Setup both devices with LoRa libraries, set one of them to TX every 200ms C. The RX device will blink an LED everytime it receives a packet D. Walk the TX device 1km away within line of sight E. Use a video recording to ensure the RX device blinks at least once per second (it is okay to miss packets)	Yes
I/O Unit 10. Vibrations generated by the motor should have high enough amplitude to convey the user information without the user facing any difficulty, but should be low enough to not alert targets	A. Power the vibration motor with 4.2V, to make it run at maximum battery voltage B. Install a decibel meter application similar to "Sound Meter" on a smartphone, to measure the decibel level of the motor while running C. Place phone microphone 1m away from the vibration motor D. Check with the decibel meter to see if decibel range is between 0 and 7 decibels compared to the existing noise floor	Yes
I/O Unit 11. The addition of a small capacitor and schottky diode should reduce the back-EMF voltage spikes generated by the motor down to a negligible level (<50% its original value)	A. Drive the vibration motor from 4.2V, using a signal generator. Attach an oscilloscope probe to the motor B. Observe the voltage spike generated by the motor during the transition from 4.2V to 0V C. Use a single-shot capture on the oscilloscope and measure the amplitude of the voltage spike D. Attach the diode in the reverse orientation and capacitor in parallel and redo the previous test (see Figure 5) E. New measurements should be less than 50% of the original value	Yes

Appendix B PCB Design and Device Prototype

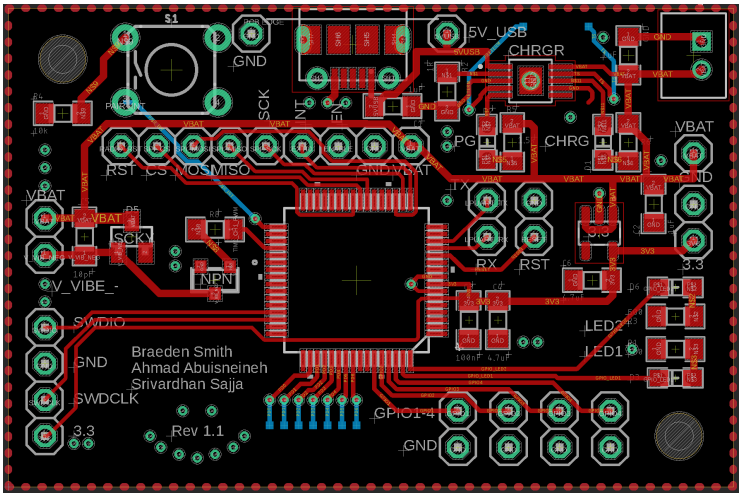

***Figure B.1**: Final PCB design made using Autodesk Eagle®*

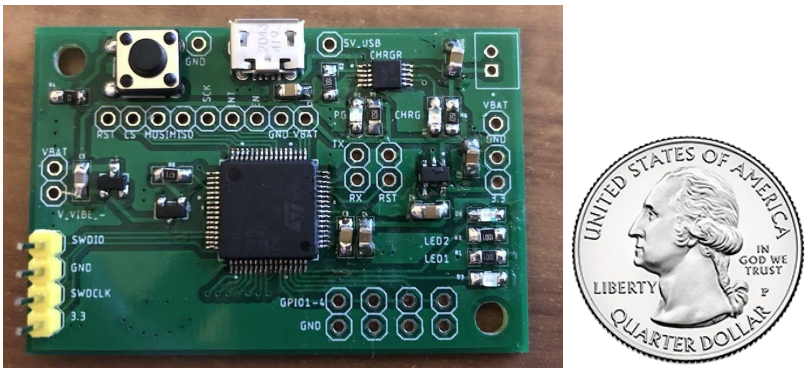

***Figure B.2**: Final tested PCB (Quarter imposed for scale)*

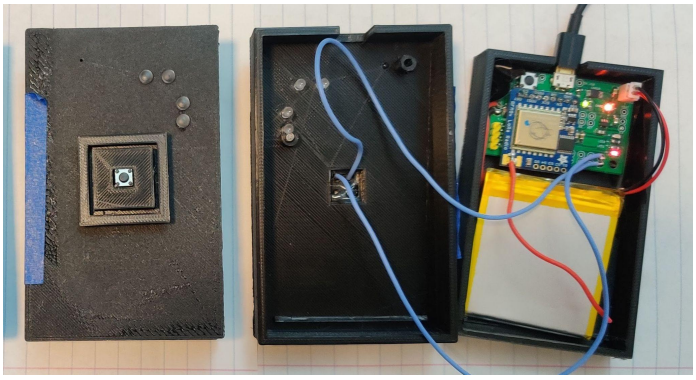

***Figure B.3**: Final Device: Unopened casing with exposed button (left)*
Opened casing with PCB & battery visible (right)