Field Radio

The University of Central Florida
Department of Electrical Engineering and Computer Science

Dr. Lei Wei Senior Design I: Spring 2022
Group 5:

Brian Taylor (Electrical Engineering)
Elier Bermudes (Computer Engineering)
Daniel Sypioe (Computer Engineering)
Noah Madison (Electrical Engineering)

Sponsored by and in association with Amateur Radio Club @ UCF

Figure 1. UCF Seal

Table of Contents

List Of Tables

List Of Images

Executive Summary

Since the invention of radio communications in the late 1800's, its applications have reached a wide array of disciplines ranging from physics, telecommunications, radar, amateur radio and much more. With the rise of digital processing and computers in the mid to late 1900's, radio technology became even more accessible with the advent of devices such as modern HAM radios, portable radios (long range handheld and car radios), as well as software defined radios [21] [27].

With guides and technology becoming more user friendly every decade, people can now enter the field of amateur radio communications with relatively little to no prior experience in electrical engineering, signal processing, and mathematics. In order to engage in amateur radio communications though, people must complete an amateur radio licensing examination. Upon completion of the examination you are now allowed on certain bands and wavelengths specified by the FCC and can engage in hobbyist radio, volunteer emergency services, and much more.

Some of the downsides with modern day radio communication, specifically software defined radio, is that the set up cost is very high and may deter those who are just entering the field. In addition to this, when it comes to portable software defined radio options you are limited to the final design of the manufacturer and oftentimes they are installed with proprietary software that cannot be changed or replaced. Due to this, the Amateur Radio Club of the University of Central Florida has designated us with the task of creating a portable software defined radio system that is modular with common components as well as being open source and upgradeable so that future engineers can develop the system further.

In order to achieve such a task we designed a software defined radio based on the Raspberry Pi and Lime SDR mini, alongside complementary components which cost no more than $500 to produce. The device features a speaker, microphone, and push to talk button which will enable the consumer to listen to their favorite AM/FM stations as well as communicate with others on FCC regulated wavelengths. It also features a touch screen to visually display the necessary functions and features of the radio. This includes separate pages to access regular listening, communications, a waterfall display, a settings page, as well as a signal bar on the right hand corner of the screen. In the final stages of prototyping, measures will be taken to make the product water-resistant to sweat and mild water droplets allowing the device to be taken outdoors without risk of damage due to weather or accidents.

The most critical aspect of our device is the combination of our embedded system, radio receiver, and amplifier. The Raspberry PI and Lime SDR must be integrated seamlessly and in a way that is computationally efficient in order to reduce any disruptions to service. The amplifier on the other hand must be selected carefully since a number of factors can affect its performance such as the input/output impedance, heat dissipation, and the type of filter it uses.

1. Project Description: Handheld SDR Radio based on LIME

1.1 Project Background

1.1.1 Description

This project i a user-friendly handheld software-defined radio. Per our initial research, the LimeSDR mini and LimeSDR are both good candidates to serve as the base for our project, and we used additional hardware to add various features and make it more user-friendly.

A software-defined radio is a radio communication system where radio features which would otherwise be handled by individual pieces of hardware are otherwise implemented by software written on a computer. This technology although not new has had massive gain in popularity due to new IC Fabrication advances dramatically reducing the cost. This new availability of low cost SDR transceivers is what inspired this project.

We have the following customers and sponsors:

UCF (Customer): Senior design project to be delivered to demonstrate competence in engineering for degree completion

Amateur Radio Club (Customer/Sponsor): Provides funding and resources for the project in exchange for the deliverable and source code to be given to them at the end of the semester

Amateur Ham Radio Community (Customer): Will have access to the source code and design to be able to recreate or purchase their own version of what we will be working on

1.1.2 Motivation/Goals

The reason that we worked on this project is that there is currently no open-source design for a modular, portable, low-cost, and user-friendly SDR available to the HAM radio community. This is problematic because a lot of the most prominent handheld radios available in the market have closed-source designs, and are often made with shoddy materials. These generally do not meet FCC guidelines and are a big issue for the community as a whole.

We have fixed the aforementioned lack of an open-source handheld SDR by using the LIME to develop a baseline that future hobbyists, developers, and students can build off of to add additional features to. This was at the request of the Amateur Radio Club, our Sponsor. The main goal of the project was to meet all of the functional requirements while still allowing for expandability and community reproducibility. so that the optional requirements could be introduced and the original feature perfected at a later point. Because of this less than traditional motivation our design approach will try and match. This means a large portion of our design process is not in a vacuum but rather in consideration with that open source community objective.

Basic Goals

The following features are absolutely required for the final project to work. We successfully met all of our basic goals. If it is missing any of these, it should be considered unsuccessful:

- Receive and transmit FM (Frequency Modulation) signals
- Within budget ($500 maximum production cost)
- Single T/R antenna
- Display
- External Controls
- Open Source Software Design

Advanced Goals

The following features are very important for the final project to have, and we strove to meet all of them before aiming for any of the stretch goals, unless the goals could be worked towards in parallel. If the project meets all of these goals, it deserves an A:

- Low production cost
- Adjustable output power level
- Long battery life
- Configurable
- Multipurpose
- Easy to use
- Portable
- Fully Programmable from a computer
- Modular
- Memory presets for FM use

Stretch Goals

The following features are optional, but the design of the base product supports these being added as either software expansions or hardware plugins. We were able to have a waterfall display. If it includes any of these, it should be considered a great success:

- Touchscreen
- Receive SSB AM (Amplitude Modulation) signals
- Automatic repeater signal discovery/scanning
- Store either demodulated voice or I/Q (Quadrature) samples to a microSD (Secure Digital) card
- Expanded UI (User Interface) through phones via bluetooth, USB (Universal Serial Bus), or HTTP (Hypertext Transfer Protocol) over wifi
- Waterproofing against rain
- Battery level Indicator
- Satellite communication tracking features
- Waterfall display via spectrum scope

- APRS (Automatic Packet Reporting System) support including: GPS (Global Positioning System), and time sync (should be in the form of a hardware plugin)
- Expanded transmission support for 900Mhz at 1W-4W
- Additional encoding and decoding modes, including AM FM DFM APRS DSTAR Fusion DMR DRM codec2 SSB
- Receive NOAA (National Oceanic and Atmospheric Administration) weather radio signals and support for alerting with SAME (Specific Area Message Encoder)

1.1.3 Demonstration Plan

The goal of our demonstration was to show off as many of the goals that we have satisfied in a short amount of time and with no uncertainty on the part of any stakeholders. Many of the goals could be proven to have been met in our documentation, and thus these goals will not be demonstrated. This includes an open-source software design, proof of modularibility, and low production costs.

Other goals may take too long to demonstrate, such as programming the device from a computer. Finally, some goals may be so obvious that they do not have to be specified, such as the fact that our project has a display and a single antenna. Thus, our demonstration was something that shows off the following goals:

- Receive and transmit FM signals
- Receive AM signals
- Adjustable power level
- Memory presets for FM use

To show this off quickly and easily, we will connect to various nearby FM radio stations which we have set on the device ahead of time. Additionally, we will transmit a signal over FM/AM bands and use a separate radio to receive those signals. Lastly a separate RF power meter was used to measure the output power.

1.1.4 Related Work

One of the most powerful digital radios on the market today is the RFinder Android Radio retailing for $1,299. It combines an android smartphone with a powerful transceiver capable of outputting 4 watts of RF (Radio Frequency) transmission power. Via the RFinder you are able to connect to an EchoLink node and the device will auto configure its settings in order to provide the best quality TX/RX transmissions on DMR and FM stations.

The RFinder will also allow you to connect to 3G, 4G, and WiFi if in the event you are unable to connect to an EchoLink with support from ATT, Verizon, and Google mobile services. Since it is powered by Android, you are able to download and install supported apps that enable you to use the push-to-talk feature on the device for transmissions. With repeaters in almost all countries, the RFinder is used in many environments and applications ranging from remote industrial worksites, search and rescue operations, and amateur radio uses. The

main drawbacks of the RF finder we found were cost and lack of open source upgradability.

1.1.5 Risks

There are various risks we dealt with that are either general risks for any senior design project, or risks specific to this project. We came up with ways to mitigate these risks as much as possible.

Due to both the size of the group and the length of time that this project will be worked on, we assumed there would be a point where a group member may face unforeseen circumstances and be unable to complete their portion of work for a given period. Fortunately, our group consisted of an even split of two Electrical Engineering majors and two Computer Engineering majors. Also the Radio club itself has multiple members with a significant amount of expertise in the subject matter. Therefore, because we all did our due diligence when it comes to documenting what we are working on, there was always somebody else on the team available with enough overlap in knowledge to complete a team member's part.

Another significant risk was part availability. With the global chip shortage, many electronic parts were in high demand and were difficult to obtain. To help prevent this from affecting the timeline of our project completion, the Amateur Radio Club @ UCF ordered some of the expensive or hard to obtain parts that we needed to complete the project, including a LimeSDR. Also, Multiple backups for each system required to meet the aforementioned goals are discussed in the part selection section. Lastly, a significant amount of necessary components are already owned by the radio club and if necessary we were given permission to use their part library since the final product will be going to the club regardless. These mitigation approaches should allow for enough variability where this concern can be mitigated significantly.

A frequent issue that we have seen cited by our predecessors in Senior Design have been that their project was difficult to complete due to miscommunication or lack of communication between the group members and the stakeholders. This miscommunication could potentially be with the sponsor or potential customers. In our case, the Amateur Radio Club served as both a sponsor and a customer, so as long as we keep the channels of communication with them open there should be no issues. Additionally, one of our group members is an officer in the Amateur Radio Club, so they will have constant contact and opportunities for communication with our primary customer and benefactor. Finally, one of our group members has communicated directly with chip manufacturers and got assurance that they are willing to work with students to make sure we have the chips to meet project deadlines and goals. The approaches should be able to minimize the aforementioned communication and mismanagement issues prevalent in many senior design projects.

1.2 Requirements and Deliverables

Because this is a sponsored project The Amatuer Radio Club at UCF has consulted and had input on several requirements for the deliverables:

RF requirements

- Support out of the box for FM RX/TX, and AM RX for configured amplifiers
- Able to transmit and receive on multiple bands on a single antenna
- Support repeater interaction features: CTCSS standard subaudible tones (88-200hz) and tone squelch (RX/TX); half duplex operation (TX/RX offset, offset polarity (+ 0 -)), touchtone generation

Portability

- Have a transmitting power output of 3-5 watts for a minimum of 25 minutes
- Able to remain on standby demodulating FM signals for a minimum of 5 hours
- Easily sunlight readable screen
- Charge through USB at 5 volts
- Have multiple field configurable and modular amplifiers at one time for different frequency bands

Programmability

- Entirely open source and free for adaptation by the radio community
- Completely and easily programmable including reusable SDK and documentation on all code
- Able to connect to a computer and be reprogrammed over USB or WIFI in the field

Spurious Emissions/EMI

- Filter stairstep radio synthesizer output to prevent spurious emissions
- meet emi, modulation, Drift, and frequency accuracy as defined by the FCC
- Discussed further in related standards under constraints

Easy to Use

- Has built in Microphone and Speaker
- GUI for controller and setup of radio in the field
- GUI documentation allowing for easy adoption
- Connectors, fasteners, and components that do not require uncommon tools: E.G. should use common components no apple style design
- Documentation on common hardware use case for repair

Configurable and multipurpose

- Matches community standards and expectation for cost and complexity allowing for widespread community adoption
- Amplifiers should be modular and fairly interchangeable without compromising noise characteristics
- Uses well documented open source libraries
- Supports common connector types

1.2.1 Bands Table

This table shows the different goals that we have for transmission bands.

Type:	Bands
Minimum	2m/146Mhz
Base	Dual band (2m/70cm) switched
Stretch	2m/146Mhz, 70cm/440Mhz, 1.25m/220Mhz, 0.333m/900Mhz,

Table 1. Bandwidth Transmission Specifications

1.2.2 Engineering Specifications

This table shows the different engineering specifications which we will be attempting to accomplish with our product. Demonstrable specs will be indicated by an asterisk and highlighted, and these specs will be shown during the showcase.

Type:	Minimum Spec:	Base Spec:	Stretch:
Transmission Delay	< 10 sec	< 5 sec	< 2.5 sec
Receive Delay	< 10 sec	< 5 sec	< 2.5 sec
Output Power*	3 Watts for 25 minutes	5 Watts for 25 Minutes	5 Watts for over 30 Minutes
Demodulation*	FM RX and TX	Min + AM TX/RX	Base + APRS + packet Radio
Repeater Functionality	CTCSS standard subaudible tones (88-200hz) and tone squelch (RX/TX);	Min + (TX/RX offset, offset polarity (+ 0 -)), touchtone generation	Base + standby smart station scanning and waterfall display + duplex operation
Battery Life	3 hrs standby	5 hrs standby	6 hrs standby
Size	8 x 5 x 4 inches	7 x 4 x 3 inches	6 x 3 x 2inches
Manufacturing Price	Under $600	Under $400	Under $200
Controls*	Serial	Button	Touch

Table 2. Engineering Specifications

1.2.3 Additional Specification

Minimum

On device Gui + Display + speakers + microphone + EMI Standards met (see Requirements)

Base

Minimum + hot swappable amplifiers + pc connectivity

Stretch

Base + touch screen + local hotspot for configuration + client side pc gui

1.3 Diagrams

1.3.1 Software Block Diagram

The following diagram provides a generalized layout/order of software functionality. The receiver will have 2 modes Tx, and Rx before a mode can be accessed the modulation scheme and frequency must be selected. From there it diverges as the diagram outlines.

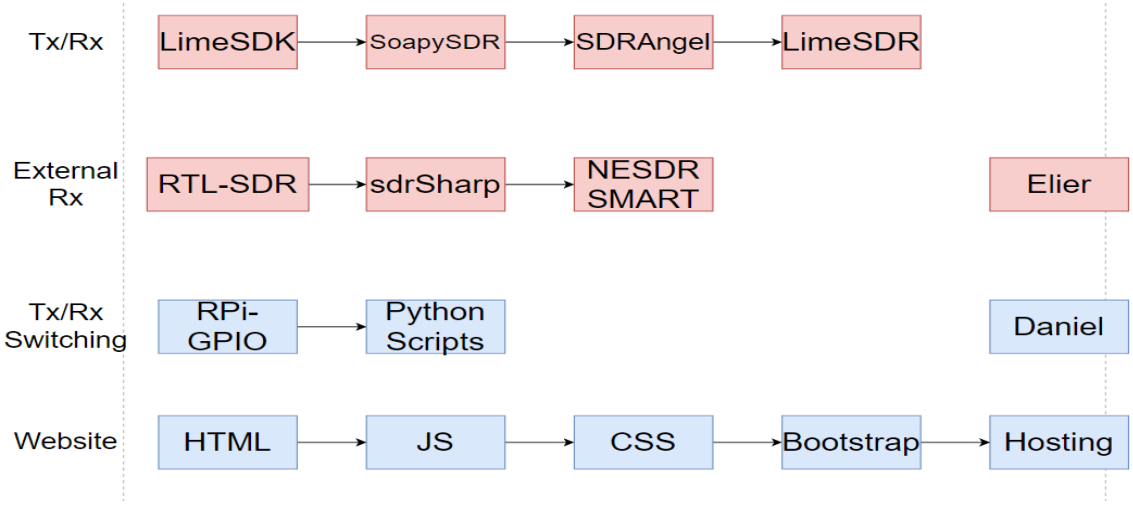


Figure 2. Software Block Diagram

1.3.2 Hardware Block Diagram

The following figure gives a generalized diagram of all of the features and how they should interact at a hardware level. The embedded system is further elaborated on in the digital system section. The battery control circuit, although necessary for functionality of the system, will not be a part of the actual handheld and will be external to the system. This diagram in particular is for a high abstraction explanation of different features that will exist and how they are to interact with other project features and systems. It is not meant to be an exhaustive description of every component, more of a summary.

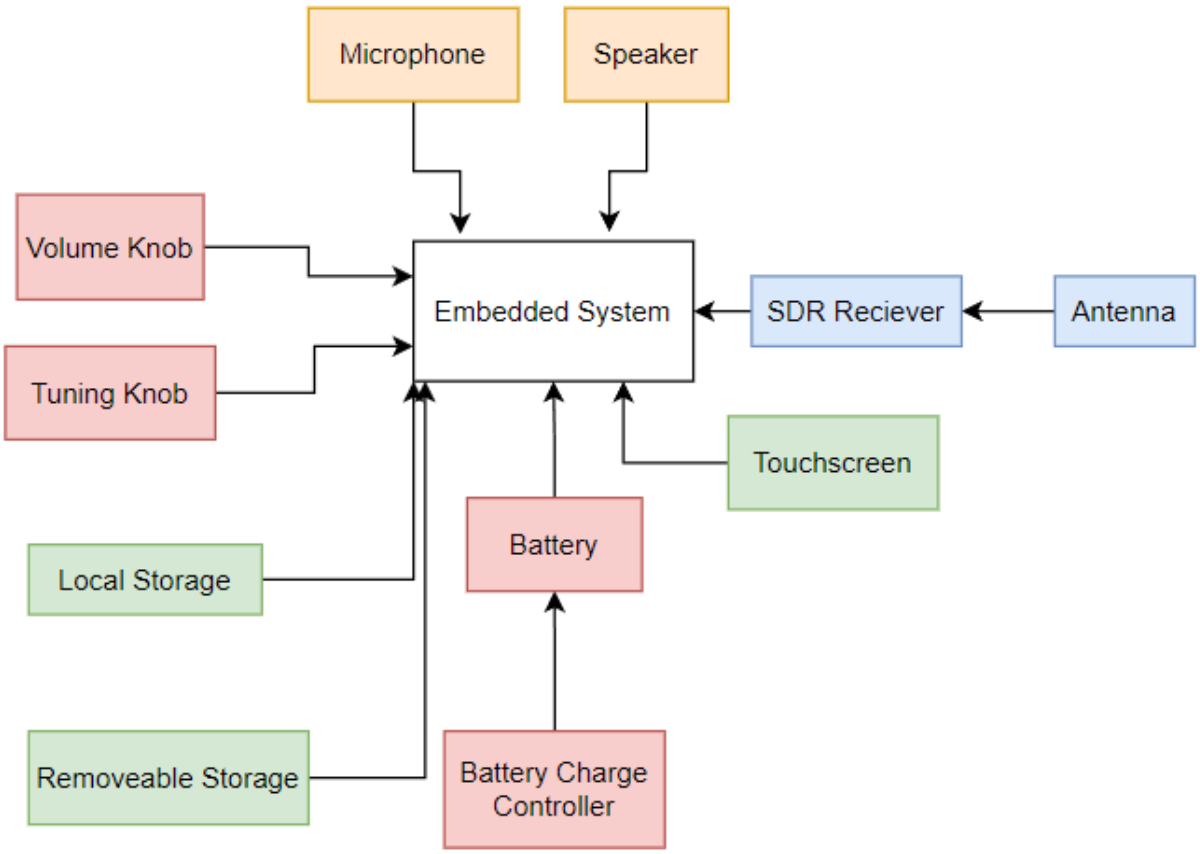


Figure 3. Hardware Block Diagram

Legend

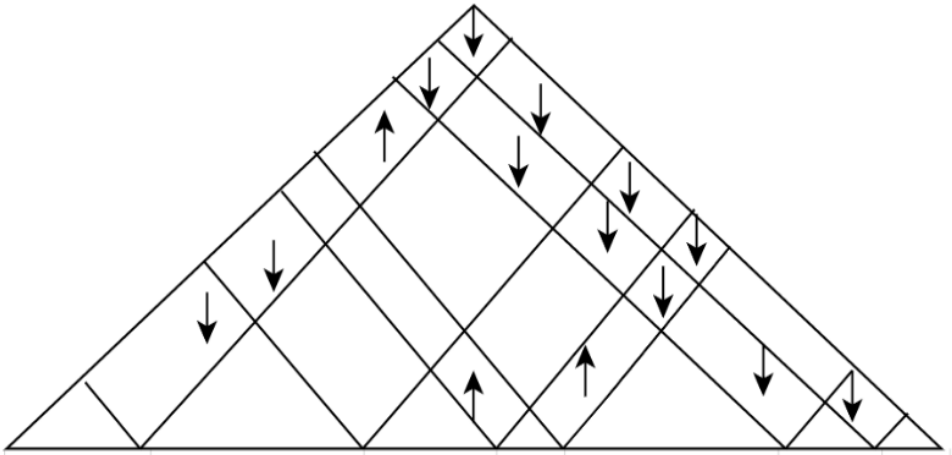
- **Blue - Elier**
- **Green - Daniel**
- **Red - Brian**
- **Orange - Noah**

1.3.3 House Of Quality Diagram

The following diagram is a house of quality representing correlation between requirements and features. Each feature is marked with a symbol denoted by the Legend representing the correlation amount. This Figure is useful for denoting the importance of each feature of the final system and how they will be prioritized. It also serves as a reference to us for what features should be optimized and what correlations/interactions these optimizations will create.

Legend	
↑↑	Strong Positive Correlation
↑	Positive Correlation
↓	Negative Correlation
↓↓	Strong Negative Correlation
+	Positive Polarity
-	Negative Polarity

Figure 4. House of Quality Legend



		Sound Output	Water Resistant	Output Power	Range	Radio Accuracy	Dimensions	Cost
		+	+	+	+	+	-	-
Portability	+	x	↑↑	↓	↓	x	↑	↓
Ease of Use	+	↑	↑↑	x	↑	↑↑	↑	↓
Battery Life	+	↓↓	x	↑↑	↓	x	↓↓	↓
Sound Quality	+	↑	x	↓	x	↑↑	↓↓	↓↓
Cost	-	↓	↑	↓	↓	↓	↓	↑↑
Targets		+90 dB	Resistant to sweat/mild water	5 hr battery life	1 mile	outputs without lag and static	7" x 5" x 3"	$500

Figure 5. House of Quality Diagram

2. Research and Part Selection

2.1 Technology Comparison

For each part of our system, we have created a table to compare the different options that we considered, as well as stated which part we selected. The selected

2.1.1 SDR Receiver

Software Defined Radio (SDR) is a versatile tool and by far the most important module on the entire system. Proposed uses for the system as a whole are local radio, emergency stations, air traffic control, and even signals from the International Space Station. These uses will require a wide range of radio frequencies, channels, and demodulation capabilities. Because of the wide range of performance metrics of the SDR this part had to be carefully considered as it will have a massive effect on the system as a whole. Some metrics looked at which are discussed more in depth in the part selection section are: filters, frequency range, output power, and FPGA processing power.

Luckily, SDR receivers have become widely accessible and affordable in recent years with hobbyist models such as the NooElec NESDR Mini 2 costing as little as $25. Higher end SDR receivers such as the LimeSDR, HackRF One, and portable SDR receivers cost $299. There are also many windows and multiplatform SDR software programs such as the sdr Console V3, SDR Sharp #, and SDRplay SDRuno that work with the SDR receivers that allow you to cycle through the radio frequencies and perform additional functions. [7] [30]

Feature	LimeSDR Mini	CaribouLite (MHz)
Frequency Range	10MHz - 3.5 GHz	Channel 1: 779-1020 or 389.5-510 Channel 2: 3-6(GHz)
RF Bandwidth	30.72 MHz	2.5 MHz
Sample Rate	30.72 MSPS	4 MSPS
TX Channels	1	2
RX Channels	1	2
Transmit Power	10 dBm	14 dBm
Interface	USB 3.0	SMI (GPIO powered)

Table 3. LimeSDR Mini vs CaribouLite

When making this decision, we were beholden to what our sponsor has requested. If the LimeSDR Mini had proved incapable of performing well in the aforementioned proposed use case, then we would have suggested that the CaribouLite had been used. However as our Sponsor has a preference for the LimeSDR Mini, as long as we are able to show that the LimeSDR is capable of performing, then it should be the one that is used.

After analyzing the specs on the LimeSDR Mini, we have decided that it is more than capable of doing what we need it to do, and because of that we have selected the LimeSDR Mini over the CaribouLite. One possible issue is the supply chain. Due to recent stocking issues the LimeSDR Mini may not be obtainable. Because of this a remade version with a slightly faster and more available FPGA was made; this is still just a minor version change, meaning it's a drop in replacement, and is supposed to be stocking soon.

If the supply of LimeSDR Minis remains low due to stock issues coming into the month of May, we will pivot and instead switch to using the CaribouLite instead of the LimeSDR Mini. The sponsor has agreed that this may be unfortunate but necessary. The CaribouLite can meet our goals but significantly limits the upgradability and experimental prospects.

2.1.2 Embedded System

This decision was a complex one, as we had to prioritize dimensions, price, and overall capabilities all in equal weight. As two of our largest requirements for this project are the price and size, we chose the Raspberry Pi Zero 2W. Obviously, It is a better choice than the Raspberry Pi Zero W due to having stronger overall performance and more available stock at only a $5 increase in price. Another good reason for choosing the pi was compatibility. Although the drivers are available for linux we know for a fact the Raspberry pi series has been used with the LimeSDR before. This Prior known compatibility is a significant plus because all USB transceivers modules are not equal so the possibility of incompatibility is not small for unknown products.

Similar to the LimeSDR Minis, there are supply chain concerns regarding the Raspberry Pi Zero 2W due to high consumer demands combined with low supply. Because of this, we have planned to pivot to an Embedded System similar to the Raspberry Pi Zero 2W if we are still unable to purchase a Raspberry Pi Zero 2W. The system we have chosen as a potential replacement is the Omega2, as it has a similar pin count and a USB 2.0 slot which would be more convenient to use than the Micro USB port on the Pi zero for structural reasons. The main and obvious drawback is the lower speed and lack of known compatibility so the first choice by a long shot is the PI. However, since this is a real world project we are willing to switch to whatever is required. The flexibility is the only way we can combat supply chain issues and make sure the project meets requirements

	Raspberry Pi Zero W	Raspberry Pi Zero 2W	Omega2
SoC/SIP	Broadam BCM2835	RPi RP3A0 with Broadcom BCM2710A1, 512MB RAM	MT7688 SoC featuring 580 MHz MIPS CPU
CPU	Single-core Arm11 @ 1 GHz	Quad-core Cortex-A53 @ 1 GHz (overclockable to 1.2 GHz)	Single-Core @ 580MHz
GPU	VideoCore IV	VideoCore IV	N/A
Memory	512MB DDR2 PoP	512MB DDR3 wire-bond	16MB Flash 64MB DDR2 DRAM
Storage	MicroSD Card	MicroSD Card	Supports MicroSD or USB Swap
Video	Mini HDMI	Mini HDMI	N/A
Audio	Mini HDMI	Mini HDMI	N/A
Wireless	802.11 b/g/n WiFi 4, Bluetooth 4.1 LE with PCB antenna	802.11 b/g/n WiFi 4, Bluetooth 4.**2** LE with PCB antenna	Dual mode 2.4 GHz 802.11 b/g/n wifi with 2 dBi directional chip antenna
USB	Micro USB OTG port	Micro USB OTG port	USB 2.0
Expansion	40-pin GPIO header	40-pin GIP header	32-pin GPIO header
Power Supply	5V/1.2A	5V/2.5A	3.3V/240mA
Dimensions	65 x 30 x 13 mm	65 x 30 x 13 mm	42.9 x 26.4 x 9.9 mm
MSRP	$10	$15	$25

Table 4. Embedded System Comparison

2.1.3 Low-Level Software

	Windows 10	Raspberry Pi OS	Raspberry Pi OS Lite
Weight	Heavy	Medium	Light
Ease of Use	Very easy	Medium	Difficult
Ease of Development	Difficult	Easy	Medium
Open-Source	No	Yes	Yes
Size	32GB	8GB	4GB
Virtual-Machine Friendly	Yes	Yes	Yes
GUI	Yes	Yes	No

Table 5. Low-Level Software Comparison

This decision was a simple one, and the comparison between different Operating Systems was mostly done as a safeguard to make sure we had considered other options. As we are developing on a Raspberry Pi Zero, some variant of Raspberry Pi OS was obviously the correct choice. Due to the low performance power of the Raspberry Pi Zero 2W, we chose Raspberry Pi OS Lite. If our development cycle for this project was longer, we may have chosen a more specialized OS with better performance for our specific use case. However due to the nature of Senior Design and its limited development time, we chose to go with a tried and true OS.

2.1.4 High-Level Software

	C++	Java	Python
Speed	Fast	Slow	Slow
Ease of Use	Difficult	Medium	Easy
Community Support	Low	Medium	High
Memory Usage	Low	Medium	High
Developer Familiarity	Medium	Medium	High

Table 6. High-Level Software Comparison

Similar to the Operating System decision, the choice of which high-level language to use for development comes down to which we can use to minimize our development time as much as possible. We chose Python because it is an extremely powerful language able to quickly and easily implement complex features through extensive use of external libraries. Additionally, the two members of our team who will be working on the software the most have expressed a personal preference for Python, and due to this it is the language we will be developing on.

There is an argument to be made that for our use-case, it may be better to develop in C++ to have our system run faster. While this might be true, this increase in speed comes at the cost of development time. For this project our list of basic goals is fairly small, while we have an extensive list of stretch goals. Due to this, it is best to minimize the time it takes to implement features so that we could implement more features overall, even if it comes at the cost of some performance speed. This lack of processing speed should not be a significant drawback and if need be we are willing to incorporate c++ dlls for processing intensive software modules.

2.1.5 Antenna

	Baofeng UV-5X3	Nagoya NA-320A	Diamond HT Antenna
Power	10 Watts	10 Watts	10 Watts
Price	18.99 for 2	20.98 for 1	32.14 for 1
Bandwidth	Adequate	Adequate	2m/70cm only
VSWR	Lower than 1.5:1	Lower than 1.5:1	Not rated
Rating	4.5/5 out of 121	4.5/5 out of 18944	5/5 out of 11
Size (inches)	17.7	17.7	15
Connector	SMA	SMA	BNC

Table 7. Antenna Comparison

Antenna are easily replaceable parts of a radio system, and thus this decision holds little weight. Due to this, we simply chose an Antenna with a low price and high reviews. That being Antenna 1, The Baofeng UV-5X3. This also comes in a two pack which is a plus for demonstration if both radios use the same antenna. As mentioned in the technology comparison a lot of the performance metrics associated with antennas are not verifiable without purchasing an significantly expensive tested antenna. So price may not mean

quality. Also If finding stock of this Antenna proves difficult, our Sponsor is able to provide an Antenna for us.

2.1.6 Audio Amplifier

	PAM8403	TDA7052
Amplifier Class	Class D	Class D
Supply Voltage	2.5-6V	3-18V
Current	16ma	4-8mA
Power (8 ohm load)	1.8W	1.2W
Channel	Stereo	Mono
Input Impedance		100kOhms
Frequency Response	20kHz	20kHz
Voltage Gain	24dB	38-40dB

Table 8. Audio Amplifier Comparison

The speaker system we selected came with an audio amplifier integrated circuit solution that features the PAM8403 Class D audio amplifier. Compared to designing our own audio amplifier this component comes pre-soldered and costs less than purchasing all of the components separately. Since radios can operate in mono playback mode our option if we wanted to design our own would be the TDA7502.

2.1.7 RF Amplifier

We selected multiple switched amplifier modules for each band with the same characteristics similar to the discussed 5W 433MHz Metal RF Power Amplifier Modules. For each band's amplifier the same approaches and performance metrics will be considered. But each band will not be explicitly discussed in detail because of the fact that they are all made using the same MIC and that is subject to change as more of the stretch goals are attempted. All of the amplifiers are made by the same manufacturer so the un documented characteristics are assumed to be the same for each band. Some of the important differences between the amplifiers are the supported bands that they are capable of working with, as well as the size, availability, and price.

The main decision was influenced by the massive benefits to cost, size, versatility and efficiency. This choice is also aligned with the open source goal in that it allows for repurposing of the overall system without changing the schematic and reprinting the PCB. The users can just order a new amplifier in a

band needed and start using it immediately. However, due to possible issues the specific amplifiers may change after testing. The overall plan will be multiple amplifiers each tuned for individual band requirements.

	RS-UVPA 5W UHF/VHF Power Amp	RF Power Amplifier 20M-512MHz	5W 433MHz Metal RF Power Amplifier Module
Type	MultiBand Tune	WideBand untuned	Switch Single Band
Heat Dissipation	Small Heat sync plus optional fan	Heat Sync	Heat Sync
Impedance	50 ohm	50 ohm	50 ohm
Power	5W	5W	5W
Filter necessary?	Has built in Filters	Yes	Has built in Filters
Availability	2-3day shipping	30 day shipping	Said 15 day shipping but that was 20 days ago.
Modularity	To Large for our needs	Smallest option	Small but would need multiple
Price (USD)	129	36.88	Around 45 for all bands covered
Bands	2m 1.25m and 70 cm	20M-512MHz	70 cm

Table 9. RF Amplifier Comparison

2.1.8 T/R Switches

There were a lot of different T/R switches available to use in our system. Given our requirements, we chose these 3 to begin with for comparisons because they all met the power requirements of our system. The MASW-008955-TR3000 is an appealing option due to its DC capabilities making it easier to work with, however almost all of its other functional characteristics are worse. Due to this, we will not be choosing the MASW-008955-TR3000 unless its DC capabilities are found to be necessary after more testing is done.

The SKY13414 and SKY13588 are both significantly more appealing options for our use case and when selecting parts we quickly knew we would be using one of these. Both of these have low insertion loss and high isolation, making them compatible with the LimeSDR and our use case. Between the two, we decided that we would be using the SKY13588.

Considering the current supply chain issues, if we are unable to find stock of the SKY13588 we will instead be using the SKY13414. The reason we are not using the SKY13414 in the first place is due to the power capabilities of the SKY13414. When looking at the datasheets for the system its power capabilities made us concerned that it would not be able to work as consistently and reliably as the SKY13588.

	SKY13414	SKY13588	MASW-008955-TR3000
Protocol	Direct GPIO	Direct GPIO	Direct GPIO
Type	SP4T	SP3T	SP3T
Max RF reverse power	DigiKey Says 70 dBm but the datasheet says 33dBm	70dBm	42dBm
Max RF power Throughput	DigiKey Says 37 dBm but the datasheet says 27dBm	39dBm	35dBm
Isolation	31dB	40dB	20dB
Insertion Loss	0.45dB	0.45dB	0.6dB
Frequency Range	100MHz-3.8GHz	100MHz-6GHz	DC-3.5GHz
Technology	Absorptive and Reflective	Reflective	Reflective
Price	1.35	1.24	1.74

Table 10. TR Switches Comparison

2.1.9 ADC

When deciding on an ADC, we had to compare the communication protocol as well as the number of bits and channels on the device. Due to the relatively low cost of ADCs overall, the cost of the actual ADC was barely a concern as none of the ADCs we found ever got close to costing even as little as $5. Additionally, we are more concerned with the number of measurement channels than the resolution of those channels. Due to this, we prioritized the number of measurement channels above the number of bits. Additionally, communication protocol was not a concern at all as the Raspberry Pi Zero 2 W is

capable of communicating over both I2C and SPI protocols with hardware and if need be bit banging anything else.

Due to a combination of all of the aforementioned criteria, the ADC that we chose was the MCP3008. It was the device with the largest number of channels that we found, while still being cheap and working at the same voltage range as the rest of our system. In addition, all of the ADCs which we compared are extremely popular products with widely available software libraries, meaning that even if we decide to switch later, the software of the system will not have to be significantly modified and concerns of support will be non-existent if adjusting to a different ADC is necessary.

	ADS1115	MCP3008	ADS1015
Protocol	I2C	SPI	I2C
Voltage Range	0-5v	0-5v	0-5v
#Bits	12	10	16
#Channels	4	8	4
Technology	Delta-Sigma	SAR	Delta-Sigma
Price (USD)	1.50	3	2.50

Table 11. ADC Comparison

2.1.10 Accelerometer

	Adafruit ADXL335	HILetgo MPU-6050
Input Voltage	5V	5V
Dimensions	0.75”x0.75”	0.5”x0.75”
Acceleration Range	+3G	+16G
Weight	0.52oz	0.63oz
Bandwidth	50Hz	50Hz
Price	$16.17	$3.33

Table 12. Accelerometer Comparison

The compared accelerometers both have high ratings on amazon and are compatible with our current design. Since the accelerometer is a stretch goal, we will choose the cheaper option and upgrade if in the event we have budget to spare. On top of that, the HiLetgo option also comes in a pack of 3 so we will

have more to spare if one is dead on arrival. In addition, our team members have prior experience working with the HiLetGo accelerometer and have found it reliable and easy to use. So because of the significantly lower price, ease of implementation/design and better size and acceleration specs, we chose the MPU-6050.

2.1.11 GPS

From a purely functional and price standpoint the VK2828U7G5LF is the clear winner. However because this is an extra stretch goal/feature the NEO-6M has significantly more support and as discussed previously we are not trying to reinvent the wheel. This is just a stretch goal so the cost can be considered less important as it will not be included in the manufacturing costs and will be a separate add on module. In conclusion, although the Neo-6M and BN-880 both have large pre-existing community support, at half the price the NEO-6M is a clear winner.

	NEO-6M	BN-880	VK2828U7G5LF
Protocol	UART	UART	UART
Popularity	This is by far the most popular option with multiple guides and tutorials	This is another popular option with some examples and guide for previous uses with the Pi platform	This option has the least Support with no evidence of previous implementation
Supported GPS bands	L1 and L2 Dual Band	L1 and L2 Dual Band	L1 and L2 Dual Band
Voltage	3.6-5V	5V	3.3V
Size	27.6mm*26.6mm	28mm*28mm	25mm*25mm
Update Rate	1-5Hz	10Hz	10Hz
Clock Accuracy	unrated	unrated	0.5 PPM TXO
Additional Feature	none	Accelerometer, Altimeter, and Compass	Compass
Price (USD)	12	25	10.30

Table 13. GPS Comparison

2.1.12 Case Materials

When it comes to materials there are countless options, but given our need for durable water resistant materials our choices are limited. When comparing PETG and Polypropylene, PETG is more durable while also being cheaper and easier to print with. When it comes to PETG vs PLA, the two materials are extremely similar with PLA being easier to work with as most 3d printers are tuned for it specifically. Due to this, we will be using PLA during development due to how common, widespread, cheap, and easy to use this material is. During production however, we will switch to PETG as it is more suited to our product's needs since it is possible the radio will be outdoors for a not insignificant amount of time and in the sun often, which has been shown to be a condition PLA is unable to hold up in, over long time periods.
The highlight means used for testing

	PETG	Polypropylene	PLA
Strength	Medium	Medium	High
Durability	High	Low	High
Price	Low	High	Medium
Ease of Use	High	Low	High
Water Resistant	Yes	Yes	No

Table 14. Case Materials Comparison

2.1.13 Battery

The choice between Lithium Ion batteries and a different kind of battery would be determined by many factors including: our ability to recharge our device, how long the device will last, and how large/heavy the device will be. With non-rechargeable batteries, we are forced to allow the user to have easy access to the battery compartment to be able to replace the batteries whenever they run out. However by choosing to use Lithium Ion batteries, we are allowed to prevent access to the battery compartment and instead have a port on the device to recharge the batteries. This choice just gives more physical design versatility as well as better SWaP characteristics.

Lithium Ion batteries do come with the downside of a significant price increase for the development and manufacturing costs of the system, as we would be including the cost of batteries in our Bill of Materials, as opposed to normal batteries where that cost could be ignored as it would be the User's responsibility to provide batteries. This downside was deemed acceptable.

Characteristic	Lithium Ion Batteries	Other types (Lead Acid)
Longevity	Extreme longevity, very low self-discharge rate with relatively low maintenance	Does not require any maintenance at all, and is very reliable. Long life cycle, and can withstand inactivity but lower overall lifespan due to a very limited cycle life.
Charging speed	Extremely fast in comparison to other battery types. Can fully charge in 2.5 hours.	Extremely slow charge rate. Full saturation can take up to 16 hours.
Voltage Capacity	Boasts a very high voltage capacity, meaning it will last longer when charged.	Low specific energy, and low power density. Poor weight to energy ratio.
Stability	Environmentally friendly and generally more stable. It is not temperature restricted and can be stored both uncharged or charged. Long periods of use will not generate heat.	It is not environmentally friendly and must be stored in a charged condition to prevent sulfation. Different versions have different restrictions, flooded versions require watering.
Weight	About 70% lighter than lead based batteries. Weight to energy ratio is extremely favorable for handheld products such as this project.	Lead is heavier in comparison to alternative elements (even nickel iron cells) and due to its low specific energy has little to show for its higher weight.
Cost	Has become cheaper with time, but originally was more expensive. Averaged $132 per KWH in 2021 and $101 per cell.	Simple and inexpensive to manufacture. Low cost per watt-hour, best value power per KWH. Lead can be recycled and reused in new batteries.

Table 15. Battery Composition Comparison

2.1.14 Touchscreen

The JniTyOpt 3.5 Inch Display was the clear choice due to its unique ability to natively support 3.5mm Audio Output. This would allow us to significantly cut down the size and costs of our Speaker as we would be able to choose one with a 3.5mm Audio Jack, which are by far the cheapest, smallest, and most common kinds of speaker. Another contributing factor to the decision was the ease of mounting. The chosen screen has pre-made mounting holes and

more easily fits with our physical design goals. These two factors alone were enough to influence our decision. [28]

	JniTyOpt 3.5 inch Display	**iUniker Raspberry Pi Screen**
Refresh Rate	60Hz	60Hz
Screen Size	3.5 inches	3.5 inches
Resolution	480x320	480x320
Video Support	HDMI	GPIO
Audio Support	3.5mm Audio Output	None
Touchscreen	Resistive	Capacitive
Power	Micro Usb	GPIO
Power Consumption	130mA/5V	5V
Weight	9.1 oz	2.12
Dimensions	3.46 x 2.99 x 0.47 in	3.94 x 1.97 x 1.18 in
Price	$28.99	$29.99

Table 16. Touchscreen Comparison

2.1.15 Speaker

	GPIO	Bluetooth	HDMI	USB	3.5MM
Price	$13	$16	$30	$20	$8
Size	Medium	Medium	Very Large	Medium	Small
Availability	None	Average	Average	Average	Average
Ease of Use	Medium	Difficult	Difficult	Easy	Easy
Power Source	Pins	Independent	HDMI Port	USB Port	3.5MM Port

Table 17. Speaker Comparisons

There are many different options when it comes to audio output. We can immediately rule out using converters to output audio through the HDMI port as this would require too much space. A speaker that connects to the GPIO pins

would be nice for multiple reasons. However this was ruled out because we have a lot of other devices that will be heavily using the GPIO pins making it significantly more difficult to use than any other connection type. This could also lead to software issues down the line; if too much GPIO latency is introduced the device may not be functional. Bluetooth was quickly ruled out at least partially due to required power however this feature may be added just for convenience as a secondary option.

This leaves us with deciding between a USB speaker and a 3.5mm audio speaker. Unfortunately the Raspberry Pi Zero 2W does not have a built-in 3.5mm audio jack port, so we would normally be forced to select the USB speaker with the USB hub. However, the touch screen that we selected has a built-in 3.5mm audio jack port, so we will be able to go with a 3.5mm speaker. This is the best option by far due to it being the cheapest, smallest device, while still being readily available and very easy to use. [22]

2.1.16 Microphone

	GPIO	USB	3.5MM
Price	$7	$4.50	$8
Size	Small	Tiny	Tiny
Availability	Low	High	Average
Ease of Use	Medium	Easy	Easy
Power Source	Pins	USB Port	3.5MM Port

Table 18. Microphone Comparison

Similar to audio output, there are many different choices for audio input. In this case, the GPIO-based solution would not take up all 40-pins, so it is not as bad of a choice as it was in the case of audio output. However, due to the reasons discussed in the speaker section, if it is possible we would prefer to keep as many of the GPIO pins free for future expandability as we can. Because of this, we are left with two compelling options: either a direct USB or 3.5MM solution.

There is no cheap and simple solution that would make the 3.5mm audio jack input work. This is mainly because the Raspberry Pi Zero 2W does not have a slot for 3.5mm audio, and the screen that has a 3.5mm audio jack port is output only. So in order to incorporate a 3.5mm we would need another module, but since a usb hub will already be available the USB option is more convenient.

Because of the aforementioned convenience and price benefits, we chose to use a USB microphone. [29]

2.2 Part Selection (30-40 pages)

2.2.1 SDR Receiver

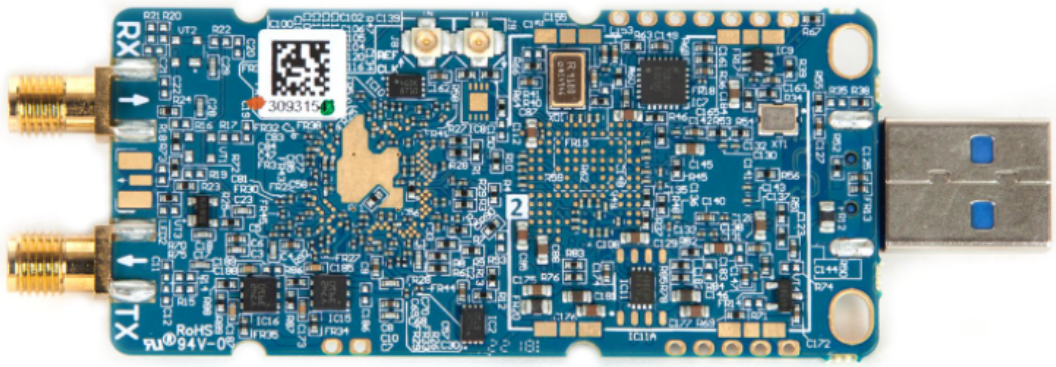


Figure 6. LimeSDR Mini

The SDR selected will play a critical role within the system itself, as our RF capabilities are completely limited by the capabilities of our SDR Receiver component. As the sponsor wishes for the project to be based on LIME technology, the preferred SDR receiver we will use to design the project will be the LimeSDR Mini. Using an SDR Receiver generally means an SDR system will be required to handle significant amounts of digital signal processing, this would require a powerful and fast external FPGA. The FPGA on the Lime mini is the Altera Max 10 which is a significantly powerful FPGA and a major selling point. On top of this the SDR module needs a good RF front end; the Lime SDR has a decent one with a preamp providing up to 30 mW of output power and adequate RF switching capabilities. This SDR is also Duplexed and supports a Wide band, which Supports our project goal of Versatility and bandwidth as described in the goals section.

The LimeSDR Mini features the same radio transceiver as the full sized LimeSDR model at a fraction of the size, cost, and weight. The main differences between the two boards are that the LimeSDR Mini is a two channel device instead of a four channel, as well as having a slightly less impressive FPGA. One benefit for our purposes is the inclusion of SMA antenna connectors rather than micro U.FL antenna connectors. This allows us to use hot swappable antennas and amps without an adapter. As the current supply chain shortage continues, we felt it critical to consider backup plans for this component given that it fulfills one of the most critical roles possible within the project. The LimeSDR Mini

features an RF frequency range from 10MHz to 3.5GhHz, and uses two separate channels for TX and RX with multiple different matching networks connected to the RF transceiver. These separate connectors for TX and RX allow for the possibility of Duplex operation and significantly Easier T/R switching for half duplex.

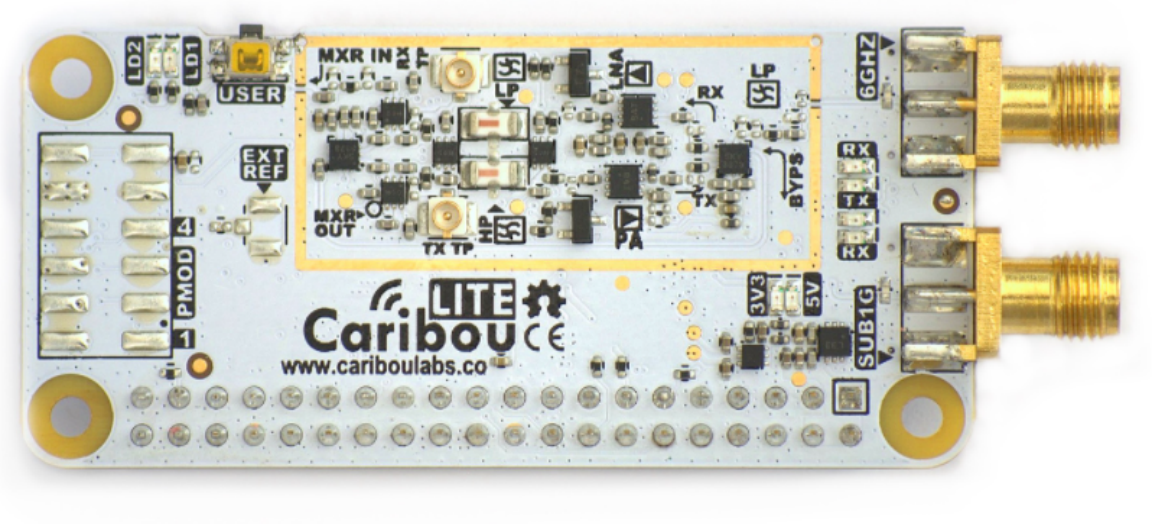


Figure 7. CaribouLite SDR

For this purpose, we also considered a Caribоulite SDR receiver as a secondary option to the SDR receiver. It also stands as a high bandwidth TX/RX SDR that is designed to be a highly compatible radio HAT for the Raspberry Pi platform. The CaribouLite is the most affordable still open-source SDR evaluation platform as well with TX capabilities. Also it is designed to complement the current SDR climate by being scalable and standalone with the uncommon capability of simultaneous dual channel software control. Aside from cost, another main benefit is availability. The Caribоulite Does have a guaranteed ship date of may 15 which makes it a good backup if the lime has supply chain issues.

Unique to the CaribouLite, is it's utilization of the SMI present on all 40-pin versions of the Raspberry-Pi platform, this allows for an impressive data exchange rate of ~500 Mbits between the HAT and the RPI interface chosen (such as the raspberry pi zero 2 w). This interface is especially useful for its accessibility with Linux applications, as it is programmable from almost any Linux application with Broadcomm's API.

The statistics comparing the LimeSDR mini to the Caribоulite show that while the LimeSDR mini sacrifices a bit of transmit power, it operates with a much higher sample rate and analog bandwidth than the Caribоulite, and has significantly higher hardware capabilities as a result. Although not completely necessary for the current defined goals. The main goal of open source versatility makes the LimeSDR a significantly better choice. ("LimeSDR vs LimeSDR Mini")

2.2.2 Embedded System

The embedded system is a very critical part of our project design. Our SDR handheld radio will only be as strong as the computational strength of our embedded system and SDR components, so if this were to be the weak link in comparison to the LimeSDR, we would be limited to the strength of our embedded system. Primarily, the embedded systems role in this SDR radio stands as the primary computer for any non-RF related task expected of the project, the “master” of any communication occurring within the project itself. For example, the microphone audio input, decryption, storage and display would all be the simultaneous job of this embedded system to take the microphones input and perform whatever task the user would like to do with said audio input coming from the microphones (I.E. if the user wishes to store whatever the microphone hears as data, the embedded system would accomplish this goal by writing to an SD card).

The embedded system has the added requirements of size restrictions (many computers can be automatically thrown out as they are too large, this project is designed to be handheld and cannot have a computer larger than our system restrictions) as well as power restrictions (we cannot exceed a few watts as this would threaten battery life as defined in our design expectations). Due to these restrictions, the primary candidates for our embedded systems include smaller, single board based computers featuring serial communication protocols and multiple digital/analog GPIO pins available for components to be attached to (preferably with power capabilities).

The chosen embedded system we will design the SDR Radio around will be the Raspberry Pi Zero 2W. The Raspberry Pi is a small and powerful single board computer that uses the Linux operating system and comes with multiple embedded systems. This more than fulfills many of the required and optional features of the portable radio. We specifically chose the “Raspberry Pi Zero 2W”, as the primary versions of the Raspberry Pi (such as the Raspberry Pi 4) are often extremely power hungry and draw an amount of current that would not be suitable for our needs. A good example of this is the Raspberry Pi 4 maxing at 7 watts, while the smaller and cheaper Raspberry Pi Zero 2W maxes at 3 watts. Via the Raspberry Pi’s onboard GPIO modules can be plugged directly into the GPIO in the form of a Pi Hat such as a touchscreen, battery holder/charge controller, or GPS module. Onboard USB ports and a 3.5mm audio port allow for the SDR receiver to be connected as well as a speaker and microphone.

Shown below is a picture comparison of the Raspberry Pi Zero 2w vs the Raspberry pi 4, showing the overwhelming size difference between them. This is another factor we had to consider, and ultimately influence our decision [43].

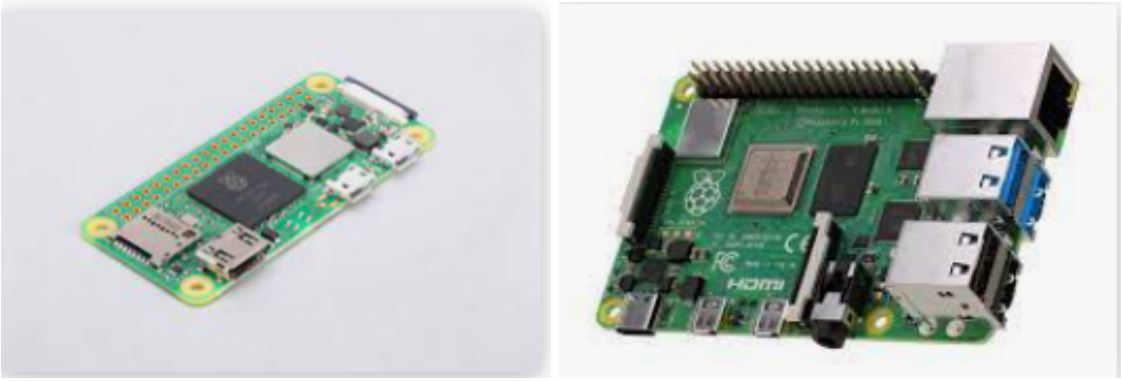


Figure 9. Size comparison of Raspberry Pi models [44]

We also considered looking at other options such as the Onion “Omega2” computer, potentially as a backup to the Raspberry Pi should we run into supply chain issues as Raspberry Pi’s are considerably difficult to find in the current climate. The Omega2 is very close in statistics to the Zero 2W, requiring 3.3V GPIO in and using USB 2.0. It also has multiple drawbacks including: non removable limited storage and low computational power with a clock speed of about 400 MHz compared to the Pi's 1 Ghz. With these drawbacks this option is also $10 more expensive making it significantly worse than the Raspberry Pi Zero 2 W. However, it is a suitable alternative with high levels of community software support should supply issues cause this backup plan to become necessary.

The Omega2 is shown below and is significantly smaller than the Zero 2W, which would benefit our project in the interest of being handheld. However this size difference isn't much and we decided the margins in which it is smaller than the Zero 2W does not make a definable critical difference.

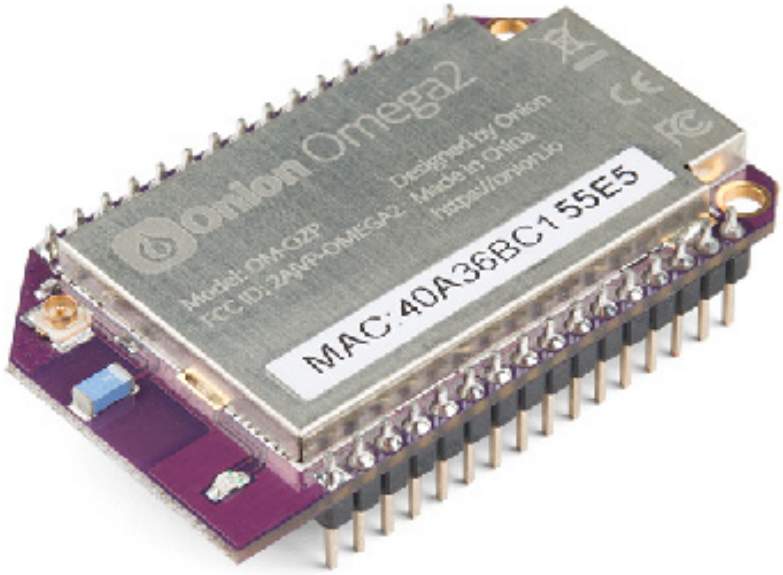


Figure 10. Omega2 [40]

To compare the Pi zero and Onion2 As shown by the technology comparison chart: there were significant downsides with the lack of HDMI video

output. This would mean we are required to use their closed source OLED expansion boards which requires access to all32 pins or a workaround PCB system would have to be designed by us. Their board may leave us without options for GPIO and it would limit the processing power significantly as the HDMI transceiver would not be done with built in hardware. This would take a significant amount of time to work out. Most likely requiring us to design our own audio and video encoder interfaces, which would directly increase the size of our PCB and thus drastically affect our overall system (since we are not creating our own TX amplifier pcb size is important to overall size). For these significant drawbacks we plan to keep the Omega2 as an absolute last resort to the Raspberry Pi Zero 2W.

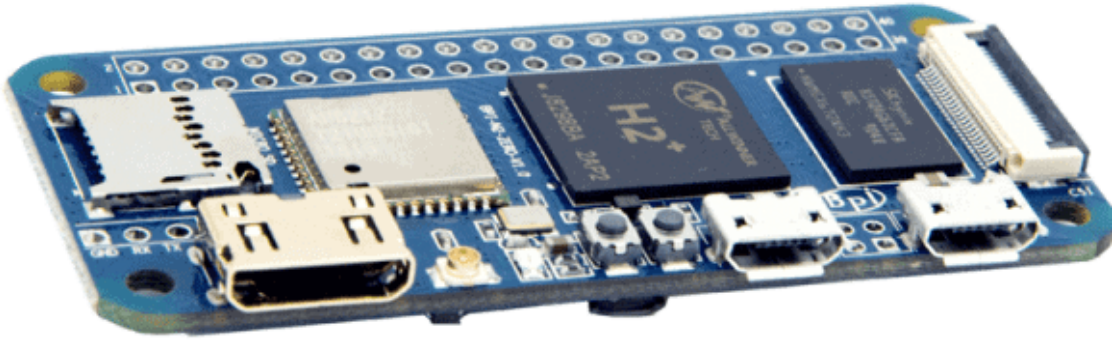


Figure 11. Banana Pi BPI M2 Zero [53]

The Banana Pi BPI M2 Zero is another alternative to the Raspberry Pi Zero 2W that could be considered. It boasts an extremely similar form factor while still having all of the same requirements that we need, such as mini-hdmi output, micro-usb port, micro sd card port, and built-in wifi and bluetooth. This would be just as good as the Raspberry Pi Zero 2W hardware wise. The reason that we would choose the Raspberry Pi Zero 2W over this is that there is more software support specifically for the Raspberry Pi Zero 2W as it is a more popular system than any of its rivals. [53]

2.2.3 ADC

An ADC will be necessary for the measurement of the volume/gain combination knob, the battery voltage and the voltage going into the TX amplifiers controlling the output power. This is a mostly simple part with only the supply voltage, number of channels and communication protocol mattering. We mainly chose this part based on the availability of libraries and popularity. Since we have 3 voltages to measure the min number of channels is 3 but it is possible we may need to measure more after testing. Because of this the more channels

the better. The ADC we chose was the MCP3008 for reasons described in the Technology comparison.

2.2.4 Accelerometer

A stretch goal that we have as a group is to make the device as versatile as possible. To do this we can add an accelerometer component that interfaces with the Raspberry Pi and can update the system on the user's acceleration and velocity in real time. If in the event the user is stationary or moving at a pace similar to walking or running, all features can be enabled. This is mainly required for APRS. For anything over running speeds, a warning message can be displayed on the screen which prompts the user to allow low resolution GPS estimation.

Size:

Accelerometers are very small components that oftentimes are no larger than a quarter. Due to this the only changes we would need to make would be the necessary mounting brackets as well as space to feed the wires.

Price:

This is an optional feature therefore the lowest priced accelerometer available should be chosen without compromising integrity of the data.

Additional Features:

Most common accelerometer modules utilize the standard IIC (Pronounced eye squared see) communication protocol or some similar interfacing technique. Some accelerometers come with additional features that include a gyroscope sensor, temperature, compass, and data logging capabilities. For our project scope though, a standard accelerometer is enough and because this is for stretch goals the end user has the ability to easily modify the component as they see fit.

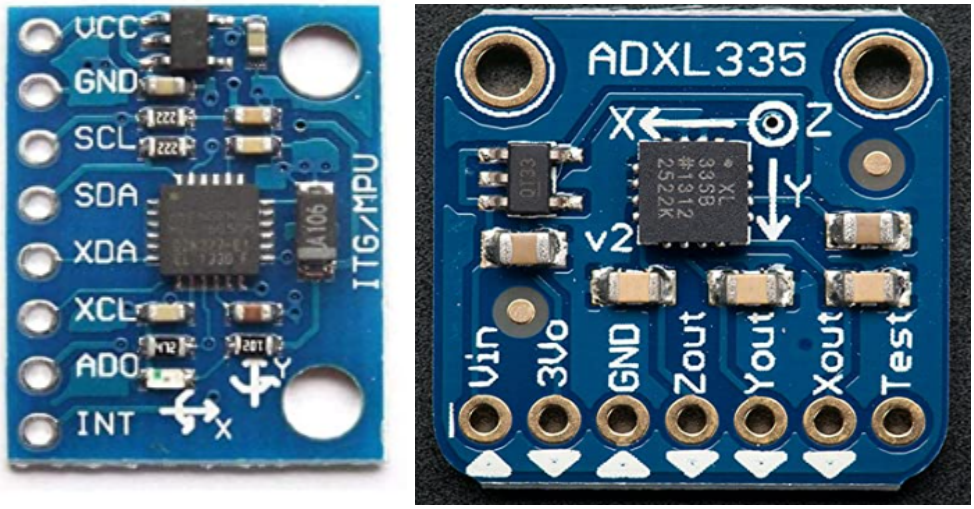


Figure 13. Accelerometer modules (HiLetgo MPU-6050), (Adafruit ADXL335)

2.2.5 GPS Module

A GPS module is a necessity for the APRS repeater functionality goal. Although this is a stretch goal the best option should still be chosen. There are multiple things to consider for the GPS module including: Size, Price, Supported Bands, Update Rate, Clock Accuracy, and communication protocol.

Size:
Since this is supposed to be an add on a possible addition to the frame and size requirements may be needed. However ideally it could be internally integrated with no additional space required other than antenna cutouts.

Price:
Again since this is an additional feature it does not necessarily have to meet price requirements but due to the open source repeatability goal/requirement the lowest cost option should be chosen.

Supported Bands:
GPS currently has 3 bands denoted by L1, L2, L5. L5 is relatively new and won't have any non-military modules supporting it. Each band not only gives us the maximum possible updates per second but also the amount of time it takes for the GPS to Acquire the constellation. Because the L2 constellation is so much newer and thus faster any module chosen will preferably use this band.

Update Rate:
Each individual GPS has a different update rate which is the number of updates in position per second. This is generally limited by the module itself but sometimes it is linked to the constellation. The APRS standard has no requirement for GPS update rate so a reasonable update rate should be considered to generally not affect APRS performance. But, since we want the versatility described in the requirements this could potentially be on a plane meaning the update rate should be as fast as possible.[19] [62]

These are some GPS modules which we have considered:

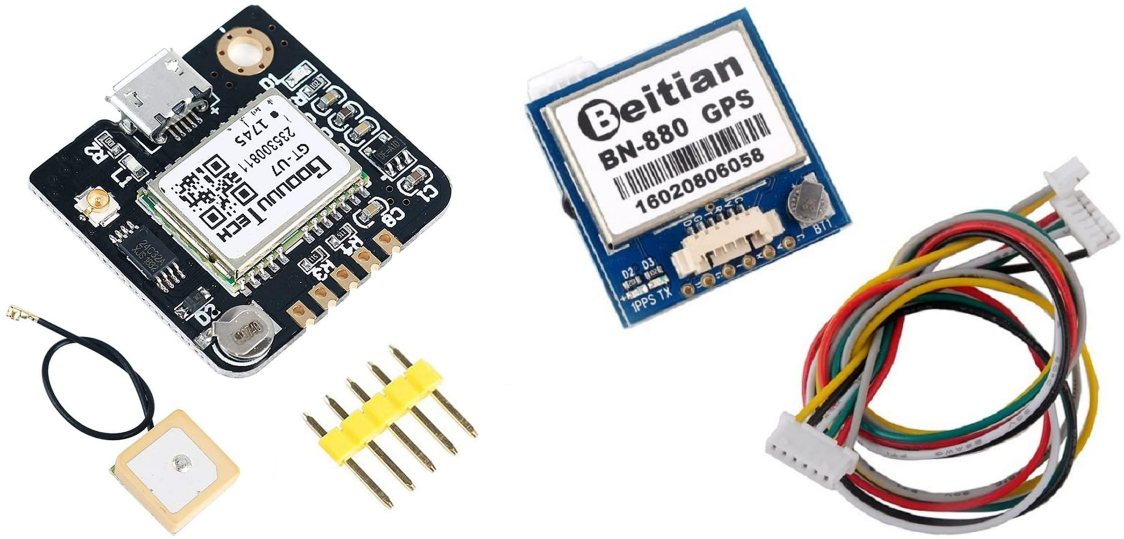


Figure 15. GPS Modules [20], [18]

2.2.6 Case Materials

For our system to be water-resistant we have to have all of the electrical parts enclosed in a case, as having them exposed to the elements would make them vulnerable to being shorted by any droplet of water. When it comes to prototyping, one of the best solutions available to us would be 3D printing a case. UCF has a 3D printer available for public use to all of its students as long as we bring our own materials, but we would still need to decide on a material based on if it meets our requirements.

Price:

The material for our case is a low-priority part of our system as it would likely be changed once the design is out of the prototyping stage, and 3D printing is unlikely to be employed if this product were even to be manufactured at a large scale commercially. Additionally, due to the small size of our product any difference in material costs are unlikely to have a large effect on the overall costs of our system as we likely would not be buying a large amount of 3D printing material.

Shock-Resistant:

Our radio system would likely be often used outdoors and while a user is moving, so it is extremely likely to be dropped. Due to this, we would like for our case to be durable and shock-resistant to a reasonable degree. This would be to minimize the damage on any internal electrical components. Another possible solution if needed for durability may be a hybrid of flexible and rigid components. This could be something like a ninja flex bumper on the edge if necessary. This would also need to be attached to the hard shell so the possibility of gluing to the material. So both of these factors will be considered in the shock resistance metric shown in the technology comparison section.

Water-Resistant:

Due to the nature of 3D printing, any case we make is likely to have inferior water-resistance compared to something coming out of a factory. This is just due to the physical layered nature of the FDM 3d printing process. To make up for this, we should choose a material which is known to be particularly water-resistant when used for 3D printing to help make up for this gap. However this may also not be water resistant enough so other options will be considered like sealing and painting. Because of this paintability which could directly affect water resistance will also be considered in the same vein.

Temperature-Resistant:

The device would likely be used in a wide range of temperature extremes ranging from sub-zero temperatures in northern US states during winter to temperatures over 100 in Florida during the summer. Due to this we would like to have a material which can withstand various temperature extremes. This criteria may be more difficult to fulfill than the others due to the way that materials work, so our goal would simply be a material that can withstand temperatures from 60F to 80F indefinitely. Ranges beyond this would be good but not necessary. The main reason this is considered is longevity. If this system is placed outside even in a weatherproof box, thermal expansion cycles could eventually cause cracks to form. [59]

Here are some materials we considered:

Figure 16. PETG [55]

PETG

PETG is a semi-rigid material that has very good impact resistance. Its surface is softer than other similar materials so it is prone to wear, however it does have good water resistance as it is often used to manufacture water bottles. The material is also good at dealing with rapidly changing temperatures.

Some of the negatives of PETG are that the surface sometimes has thin strands coming off of it due to errors in the printing process. This happens commonly but is not a significant issue for our purposes. Due to the nature of PETG we will likely not be using it for our prototyping process, but may suggest printing with it be considered when the final product is being manufactured.

Figure 17. Polypropylene [54]

Polypropylene

Polypropylene is yet another water-resistant material. Compared to PETG, this material is weaker, although it does support higher temperatures. The higher temperatures that Polypropylene supports are not necessary for our project, and it is much more expensive. The main positive of this material is that it is lightweight and does not wear down easily. In addition, the finish on Polypropylene is smooth and the material itself has relatively high impact resistance.

Some of the negatives of Polypropylene is that it is prohibitively expensive compared to nearly any other 3D printing material that we looked at, which would significantly increase our development costs if we were not able to find a way to only pay for the exact amount of the material that we used in our design. Additionally, the material often warps during printing so multiple prints might be needed. Overall, due to the positives of this material being relatively unimportant for our needs, and the negatives being too large to handle, this is not great material for our purposes.

PLA

Figure 18. PLA [56]

Polylactic Acid, AKA PLA, is an extremely popular material in desktop 3D printing. It is common, cheap, durable and easy to print Because of the wide range of positives, extrusion-based 3D printers generally print mainly PLA. PLA is a fantastic material for the majority of use cases and the majority of printers are optimized for it. From an ethical ecological standpoint PLA is derived from crops such as corn and sugarcane, making it renewable and most importantly biodegradable.

Some of the downsides of PLA are that it has low heat resistance and does not hold up well against sunlight. These 2 factors make it unsuitable for our needs as the product would likely be used outdoors very often. In addition, the filament has low impact resistance and can often break from becoming brittle over time. Despite these downsides, due to the cost and ease of use, this material will be the main choice for prototyping. Depending on testing results, this material may be considered for production, but we would likely use something that better meets our needs such as PETG instead. However PLA is a fantastic prototyping material and will be used extensively for that purpose.

2.2.7 Low-Level Software

When defining low-level software, what we mean is the operating system that we will use on our embedded system, which will likely be the Raspberry Pi Zero 2 W. When choosing an OS, we are looking for each of the following features:

Lightweight:

Whatever embedded system we end up choosing will likely be extremely weak, because for that decision we are focusing more on power optimization, footprint, price, and supported features. Because of this, what little processing power we have should not be wasted on the OS.

Simple:

The software for this project will likely not start to be designed until May, at which point we would likely already only have 3 months to finish the project. Because of this, we want an OS which is extremely easy to use and program in, as we have to go from complete beginners in it to having our project done in just a few months due to the short timespan of the summer semester.

Open-Source:

One of the major selling points of our project will be the open-source software design. While it may be considered technically acceptable if we built our open-source code on closed-source software, it would be more true to the intention of the stakeholders if the OS that we decided to use was also open-source, so that all of the code could be viewed by someone for free.

Small:

Our project has very low requirements in terms of storage. Most of the files that we will store on the radio will be the compiled code which will be running on the system, as well as the OS it is running on. Because of this, it is very likely that our OS itself will take up a majority of the file storage space, with some OSes taking up 4GB, 8GB, or even 32GB. So we would like to have a small OS so that we can buy less storage.

Virtual-Machine Friendly:

Some of our group members will be outside of Orlando for the duration of the project, and will likely face difficulties when it comes to being able to access the hardware that we will be testing on. Because of this, it would be helpful if the OS we decided to use is easily virtualizable through VMWare or other similar programs. This way the team member who is not in Orlando can still contribute.

Considering all of what was discussed above as being factors that we are prioritizing, here are some of the options we considered:

Figure 19. Windows 10 Logo

Windows 10 is currently one of the most popular OSes on the market and is used in businesses and homes everywhere. Additionally, all of us have experience using it and would not have trouble navigating it. Unfortunately, Windows 10 would not work for our use case at all, and this consideration serves more as an example of everything that could be wrong with an OS for our purposes.

Windows 10 is not lightweight at all, as there are many programs which will attempt to run or update in the background such as OneNote, Windows Updater, etc. Windows 10 is also not simple to run embedded firmware code on, despite being very simple to develop for. This is due in part to the desktop environment, as well as lack of access to the UNIX Command Line Environment. Additionally, Windows 10 is not open-source, as Microsoft's business model revolves around selling Windows 10 licenses to businesses. Next, Windows 10 is an extremely large installation, coming in at around 32GB due to the OS reserving space for updates. This would mean that each of our devices would need 64GB of storage to also be able to store our executable code. Finally, it is not even possible to run Windows 10 on a Raspberry Pi Zero 2 W. This is because the Raspberry Pi Zero 2 W and Raspberry Pi 1 both run on ARMv6 CPUs, while Windows 10 requires ARMv7 CPUs. One last positive of Windows10 is that it is easily virtualizable.

In conclusion, Windows 10 is one of the worst possible choices that we could make when deciding what OS to run on the Raspberry Pi Zero 2 W. This goes to show that even the best products can fail to fulfill your requirements based on your use case. [16]

Figure 20. Raspberry Pi OS Logo

Raspberry Pi OS is considered the “default” option when it comes to installing an OS on a Raspberry Pi. It is optimized specifically for the Raspberry Pi series of devices, so there is a guarantee of a minimum level of functionality with them. It meets many of our requirements, but there are a few trade offs compared to some other options.

The Raspberry Pi OS is more lightweight than Windows, as most Linux distributions are usually more lightweight since it is up to the user to decide a lot of the programs that they will be installing, as opposed to Windows which comes with a lot of bloatware. The OS is very user-friendly and easy to understand, especially since some of our team members have previous experience using it. The OS is open-source since it is based on Linux, and only requires a minimum of an 8GB microSD card, 1/4th the amount of space that Windows 10 needs. Lastly, the Raspberry PI OS is virtualizable.

The Raspberry Pi OS is a great choice for Raspberry Pi’s in general, but it has some trade offs which will be considered in the Raspberry Pi OS Lite section.

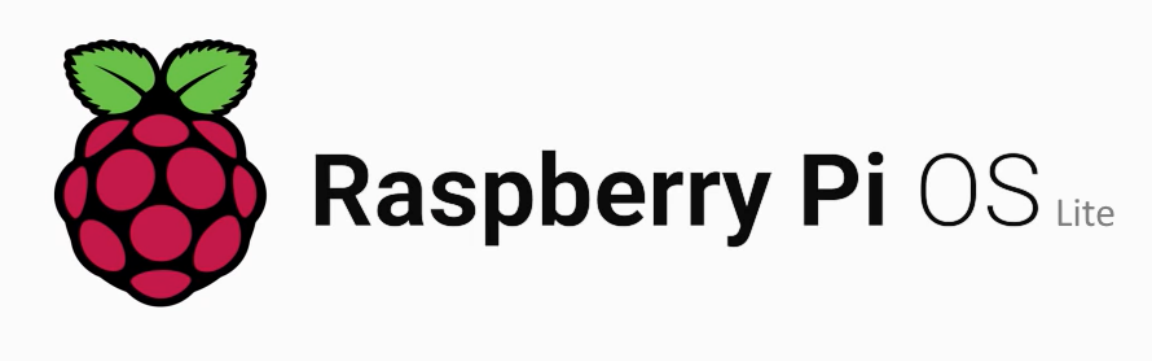


Figure 21. Raspberry Pi OS Lite Logo

Raspberry Pi OS Lite is an even more lightweight version of Raspberry Pi OS which is designed specifically for weaker Raspberry Pi devices such as the Raspberry Pi Zero 2 W. Similar to the OS it is based on, this OS meets many of our requirements but has trade offs compared to the full version of Raspberry Pi OS.

The Raspberry Pi OS Lite is even more lightweight than Raspberry Pi OS, as there are even less programs running in the background taking up processing power. The OS is also open-source, and requires only an 4GB microSD card, 1/2th the amount needed for the full version. Additionally, it is easy to virtualize.

The Lite version is unfortunately not very user-friendly as it does not ship with a Graphical User Interface (GUI) installed by default. Without a GUI, most programming would have to be either done on a Command Line Interface (CLI), or written on a separate computer then moved onto the Raspberry Pi Zero 2 W. Additionally, with the current costs of storage devices going from an 8GB OS installation to a 4GB OS installation results in a negligible cost decrease. [41] (“Raspberry PI OS Lite vs Desktop: comparison between the 2 distributions”)

Conclusion: I believe development for this project should be done on the Raspberry Pi OS Lite. Although the full version may have a GUI, the finished product will need a lightweight OS and transitioning from one OS to another is bound to cause problems throughout development. It would be easier to develop a finished product first on the lite version of the OS, then not have to transition after software has finished development and testing.

2.2.8 High-Level Software

For our high-level software, we have a plethora of options when it comes to programming languages available for us to use when developing on an Debian-based operating system such as C++, Java, Python, and many more. Each language comes with their own set of pros and cons when it comes to development and end result.

C++ is a very fast language that has been in use for several decades and therefore has a wide range of development applications across many platforms as well as tons of community support and engagement. Several functions such as the waterfall and sampling will be much more efficient to operate via C++ as opposed to Python and Java, but the main drawback is development complexity. C++ does not have as high a level of abstraction as Python and Java, therefore we will have to create our own functions and libraries in a language with a fairly complex syntax. Alongside this, our group members have more experience with other languages, so it does not seem feasible to learn and develop with this programming language within the given time frame for the project.

Java is a better option than C++ due to its superior community engagement and accessibility for other developers since it is one of the most widely used programming languages for application development on the enterprise level. Java comes standard on the Raspberry Pi Operating System as well as having libraries such as PI4J which enables communication between the embedded system and its GPIO pins. It has been noted that there are some compatibility issues with the java libraries when it comes to accessing the gpio on some raspberry pi models since most are developed based on the Raspberry Pi models 3 and 4, but there are plenty out there to choose from that can work for our specific tasks. There are also several built in libraries such as JavaFX which lets developers easily and efficiently design graphical user interfaces.

Python Programming Language

Figure 22. Python Logo

Python is native to the Raspberry Pi Operating System and is a very powerful and developer friendly high level programming language that will be the foundation of our Software Defined Radio program. Python has a plethora of libraries that will be useful when creating complex features of the program such as the graphical user interface, communicating to the low level software and operating systems, and interacting with the GPIO of the Raspberry Pi.

One drawback of using python is that we may have to be conservative in both the time and space complexities of our program since the model of the Raspberry Pi that we are using only comes with 512MB of memory. Most of the memory will be allocated to the operating system and additional drivers needed for the touchscreen/modules therefore lightweight libraries should be used wherever applicable. Therefore we should prioritize any processes that are related to signal processing/analysis between the LimeSDR and the Raspberry Pi as well as sound I/O so that communications can occur in real time with limited delay.

Open source development is an important selling point of our project therefore we will provide code and documentation online for the Amateur Radio Club to use at their discretion. For this task, we will use the Github repository framework to manage our documentation, handle version control, and also collaborate with each other for remote development. There is also substantial community engagement and documentation on multiple forums and discussion boards which satisfies our requirement of being open source as long as we are within guidelines and copyright regulations.

GPIO Interfacing

Using the RPi.GPIO library we will be able to communicate to the GPIO pins of the Raspberry Pi via Python . Even though the library does not support i2c, serial communications, and PWM, it is capable of being read for a simple voltage divider circuit to check to see if the pin is active or not. This can be useful in multiple applications such as a physical push to talk button, a power button, and much more. There are other libraries that include SPI such as the spidev library that can be useful for getting readings from our battery level monitoring circuit as well as a microphone circuit for communications.

Graphical User Interface (GUI)

In order to create the graphical user interface for the software defined radio, a graphical library is needed. One of the simplest and most lightweight

graphical libraries that Python has to offer is the Tkinker library and comes standard to Python. Most of the library's built-in classes are implemented in the C language so they should be much faster than other libraries that are solely designed based off of Python or Java. Using this library we can create a cross-platform window that displays all pertinent information and functions to the SDR program. In order to maximize performance and efficiency we will be grouping similar functions into separate selectable windows so that higher level functions such as the waterfall/panadapter can run without being limited by background applications.

Audio Interfacing

For audio there are several file formats and methods we can utilize within Python and its available sound based libraries. When we receive the audio samples from the frequencies that we connect to it will be in the form of a stream of information that must be processed. We can simply convert the input audio stream into a .wav file and use a library to play the audio. This will require a number of libraries including pyaudio, wave, and sys.

Using the standard sys and subprocesses library, we can also access the low level software of the operating systems command prompts. Via these functions we are able to modify elements of the Raspberry Pi itself including the bluetooth and microphone recording if in the event a usb microphone is utilized.

On the Raspberry Pi operating system, to record audio the command arecord and its functions can be used to produce a .wav file which can be accessed via the aforementioned wave library. Using the subprocesses library we are also able to change the volume of the system.

2.2.9 Antenna

The antenna is the easiest component of the RF frontend. The antenna must only meet 6 criteria: power, price, frequency, VSWR, durability, and size. Since for RF transmission we are not trying to break any new ground; the classic 1/4 wave whip antenna will be perfect. However since we also want versatility a multiband antenna is preferable.

Power

The requirements specify that output power should be 5 Watts for the three main bands. This is the power output of the amplifier, not the EIRP: It would be impossible to have a perfect radiation efficiency. Since antennas are cheap and easy to manufacture, this should be easy to meet with common off the shelf antennas.

However, we do have the issue of return loss vs radiation efficiency. A common issue with cheap off the shelf antennas is that the VSWR will be low (<1.5) but the EIRP will also be low meaning a high return loss even but the antenna will not be radiating efficiently. This is caused by a large internal absorption of rf energy causing the power to be converted to heat. Since this would not be published on the description of any antenna that is not significantly

over budget due to testing antenna we will decide based on user reviews for this specification.

Wavelength

The wavelength of our bands obviously cannot be adjusted as they are the required amateur bands specified by the system requirements. The fraction of the wavelength is an important metric for determining antenna efficiency since this antenna will be multi-band; the middle band will always have the optimal antenna length and efficiency whereas the upper and lower end of the band will be less optimal. However, this generally should not affect antenna performance too much according to the experience of the radio club faculty advisor.

Frequency

The required bands were 144MHz-148MHz, 222-225MHZ, and 420-450MHz at an output power of 5W. One thing to note is multi band antennas may lose efficiency in different bands. The actual impedance of the antenna is also an important metric as it defines what we can use for the amplifier and in a perfect the entire system would have the same impedance. So, to cut down on impedance, the entirety of our design was decided to be 50 ohm which is generally an industry wide standard so everything should be easier to find and build for at this impedance.

VSWR

The Voltage Standing Wave Ratio is a measure for antenna efficiency that specifically gives a good idea of the ratio of power into the antenna to power radiated. This has to do with impedance matching characteristics at a specific frequency. More specifically it is a measure of the voltage ratios between different sections of the system. So a difference in impedance causes a voltage difference leading to reflection. These reflections are the main way this characteristic is measured.

VSWR is by far the most important characteristic of an antenna because what isn't transmitted can be reflected and can cause damage to amplifiers and possibly even to the SDR. This is due to reflected power heating up the final amplifier transistor causing it to degrade and possibly break down over time. An ideal antenna would transmute 100% energy the amplifier provides. This is of course not reasonable. Using other handheld transmitters as references, the VSWR should be at least less than 2 the lower the better.

Price

The price for this component was budgeted for was 30 dollars but since one of the main goals is for it to be a low-cost DIY transceiver the lower the cost the better. Since most isotropic ¼ wave whip antennas are extremely simple the antenna must only meet the other characteristics. The antenna we landed on was $19 dollars, meets all the criteria and has good reviews.

Final Decision

We considered 3 different choices. The Diamond HT Antenna, Nagoya Antenna, and Baofeng Antenna. We settled on the Nagoya Antenna, and the comparison between them can be found in the technology comparison section. [13], [37], [63].

Figure 23. Final Antenna ("Nagoya NA-320A Triband HT Antenna 2M-1.25M-70CM (144-220-440Mhz) Antenna SMA-Female for BTECH and BaoFeng Radios")

A note on directional antennas:

Directional antennas allow for an increased receive and transmitter power with a reduced antenna aperture. For this application we decided that they are not necessary. However eventually if someone wanted to use this design for fox hunts a directional antenna would be ideal. With this in mind the entire system will be designed to allow for any 50 ohm antenna in the band. To accomplish this baluns and common connectors will be used. So although we are not using a directional antenna we are making it as easy as possible to use one.

The 900 MHz stretch goal:

The 900MHz band stretch goal would require another separate antenna; the efficiency of a single antenna is reduced not only for the number of bands but also for the distance between the bands. So an antenna that could handle all bands would either be a very well-tuned multiband antenna that would be costly or would have non ideal VSWR characteristics. Therefore, for this stretch goal should we try and attempt it an entirely different antenna would be required.

Baluns

There is a possibility of an unbalanced impedance match. Due to this, a balun might be employed. A balun is like a tuner for the VSWR. It allows the

impedance at different frequencies to be tuned; this may be necessary especially if a wide band amplifier is used. A balun is essentially a transformer that allows one side to work with anything because of the essentially zero impedance. The other side will always be at 50 ohms because it is isolated.

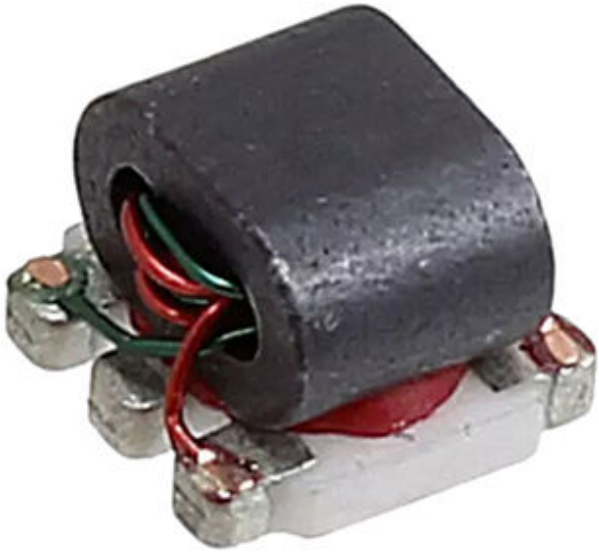


Figure 24. Balun[4]

Connectors:

Since the two most preferred options by the radio club are the LimeSDR Mini and the Caribou Lite Raspberry Pi hat, the antenna we will be using for the radio will have a coaxial RF SMA that will allow the entire system to use consistent connectors reducing cost and complexity. Since SMA is also a handheld radio industry standard parts for it are also easier to source. The not only goes for the Antenna but every RF coax connector in the system will be SMA.

Figure 25. SMA Connector [32]

SMA connectors are generally rated for 500 cycles so for this case it may be a good idea to use an SMA-to-SMA adaptor to add to life of connector so the adapter wears out not the top connector these adapters will most likely be what is connected to the case of the transceiver as well since the internal components will use SMA too. Because of this we have also picked out adapters to act as something that can wear down instead of the built in connector. This should not have any effect on VSWR or impedance matching.

Figure 26. SMA Adapter

2.2.10 Amplifier

The requirements for output power are 5 Watts/ 37dBm/ 7dBW possibly with an adjustable gain. The Output of the Lime Mini is at around 15 mW depending on the band. Because of this we not only need a power amplifier stage, but also a preamp stage. The amplifier stage should be able to handle the required bands of 144MHz-148MHz, 222-225MHZ, and 420-450MHz at the desired power outputs. Amplifiers are tricky, most operate on and are tuned for only a small bandwidth.

In a perfect world the transceiver would be able to send and receive on all frequency bands within the SDRs capabilities at the 5W output without changing anything. However the main limiting component for this will always be the amplifier. Since this is such an important part of the RF front end multiple solutions and implementation methods were considered.

Heat dissipation

Regardless of VSWR and impedance matching the amplifier will have internal uncontrollable resistances which will always generate heat. To extend the life of the amplifier good heat dissipation is a requirement. This allows for better performance and prevents damage. For this reason, an amplifier with a good heat sync will be ideal.

Impedance

As with the antenna the impedance system wide was chosen to be 50 ohm. This impedance will also be matched as best as possible to the SDR and the antenna. Since these amplifiers are untested and may use high tolerance components an impedance mismatch great enough may require a balun transformer to prevent significant reflections. However this not only adds to cost and complexity it also reduces Radiated power due to additional resistive losses in the balun.

Modularity

The amplifier should be a relatively straight forward modular component to replace since the SDR has such a wide band it would be a shame not to plan for parts being swapped to allow the use of the entire SDR. Because of this the amplifier should be accessible in the case and should use SMA connectors to allow for easy replacement. This is more a physical design aspect so it will be discussed further in its respective section we just need to note the size cannot make this impossible.

Price

This is obviously one of the most important amplifier characteristics influencing the choice between working options. The amplifier will also be a costly part no matter what the choice so minimizing cost is the goal.

The 900 MHz stretch goal

Currently the best option is multiple switched tuned amplifiers. This means the 900MHz stretch goal would require an extra amplifier and extra time tuning and testing. Because of this the current plan is to get the main bands working then if time allows purchasing and testing the 900MHz transmission amplifier. However this would not affect the ability to receive 900MHz that would only need an antenna which the radio club already has.

Wide Band Amplifiers

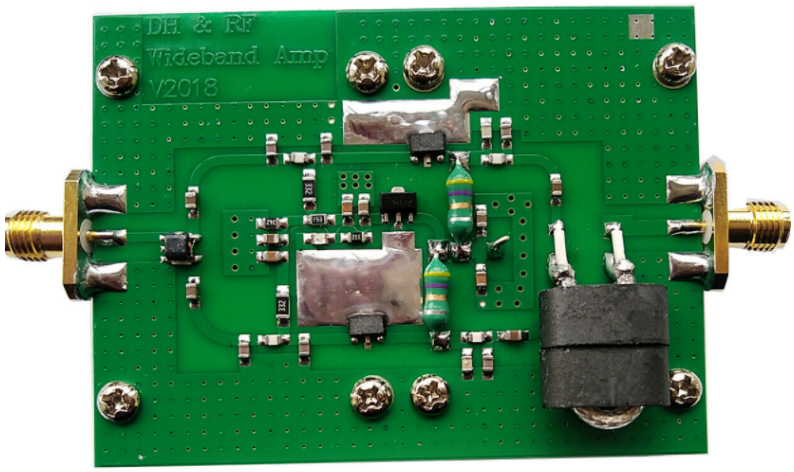


Figure 27. Wide Band Tuned [46]

Wide band untuned with switch filters

The need for tuning is clear, without tuning the amplifier would amplify noise outside of the band and possibly become non linear due to out of band noise power. The impedance of an untuned amplifier is also problematic since it is nearly impossible to ensure perfectly flat impedance and gain over a wide band. One possible option was a wide band untuned (non selective) amplifier could be used with multiple filters for each of the bands. This may be an effective yet inexpensive option; the main draw would be the necessity of a wideband high adjustable gain preamp. Because of the differing insertion losses of each component including the amplifier switches and filters the preamp gain would have to be set for each band. This would not be characterizable without testing since more of the cheap modules within our budget do not provide tests for things like this. Because of this added complexity, uncertainty and testing we decided against this option. [17]

Multiband tuned

Another option would be a wideband amplifier tuned to add our specified bands. This would be incredibly hard to do without switching and would be cost prohibitive. One option was found which is out of budget but more concerningly undocumented. Spending 20 bucks on a single band amplifier is one thing because of the low cost ease of testing and relative simplicity of the design. Spending 130 on a similarly undocumented untested and unknown amplifer is out of the question. Performance is not guaranteed and this could put the project significantly over budget should it not work out. For this reason we decided against this option.

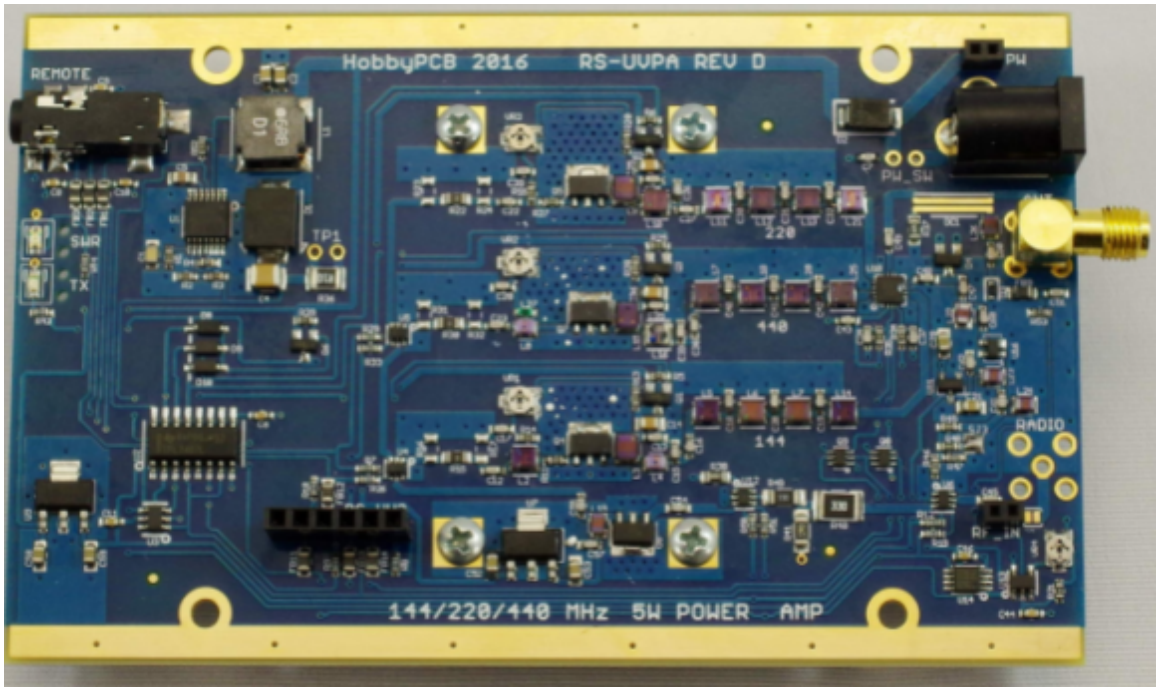


Figure 28. Multiband Tuned [47]

Smart tuning

This may be beyond the scope of the project but because of the sheer versatility of the SDR a fully software configured self-tuning routine is possible. It is possible to run through several steps and fully characterize the S domain characteristics of the amplifier and antenna combinations from here the harmonics and reflections can be predicted by something like a Kalman filter this would allow the performance of the transceiver to rival that of precision tuned single frequency transmitters.

The main drawback of this method is the necessity of an RF attenuator which was decided against in the RX conditioning section. Another drawback is the sheer complexity of something like this. This could be considered research paper level stuff because of this we decided against it for now, but this may be a stretch goal down the line. [12] [15]

Multiple Switched Tuned Amplifiers:

This is the old way of doing things. It is tried and true and will work. The main drawbacks are size, cost, and versatility. Another important drawback would be the necessity of control circuitry for the amplifiers since we still need switches. We have come to the conclusion that the benefits of this method significantly outweigh the drawbacks. This method's benefits are smaller power loss, selectivity, and reduced harmonic distortion.

Tuned amplifiers are basically filters and amplifiers in one. Harmonics get reduced because only a narrow band gets amplified. They are selective, meaning noise outside of your band is not going to be amplified; This helps with SWR, effective power, and reflections. This is important because the output of the lime SDR may have harmonics at high frequency due to the stair step characteristics of synthesized rf although additional filtering may be necessary.

This is obviously the ideal method from an electrical specification's standpoint. So we decided for each of the required bands a single dedicated amplifier would be selected, hopefully giving the best output characteristics. This also allows for cost optimization should one amplifier not meet standards.[46], [45].

0.1W (20dBm)
Working Voltage: 5-7.2V (DC)

Buying vs Building our own Amp

There is of course the option of building our own. Building an RF amplifier at the frequencies relatively straight forward, even at 900 MHz the wavelength is about 33 cm or 13 inches. So in order to prevent large amounts of interference the max length of a trace should be 1/17 of the wavelength or about .75 inches which should be doable if right after the amplification the signal gets passed through a coax connector. Generally there are 2 ways to go about this, firstly MICs (monolithic ICs). Most of these are going to be designed to be in the 5G /phone range so finding one that works in the necessary bands will be a challenge. This option is also expensive. The main upside is the ease of implementation and low noise. Good examples of possible components are

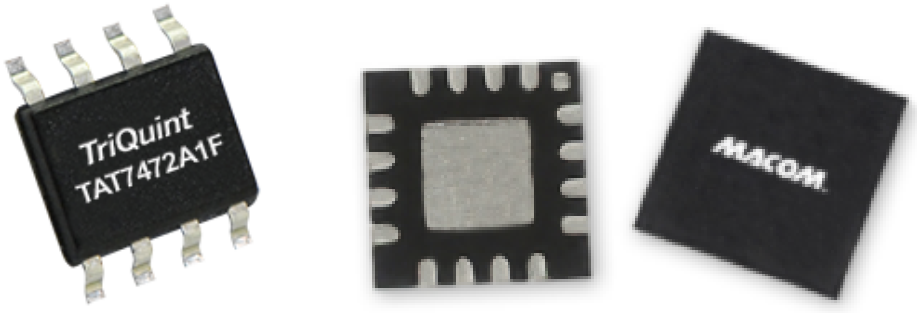


Figure 30. MIC Amplifiers

This of course is not the only way to go about building an amplifier; power transistors still work comparatively well; these are what the above modules are

made with the main issue we could come across is noise; the TAT7472A1F has a 3dB noise figure. For these designs the noise figure would be completely based on the circuit design. As a group we agree that we don't have the rf experience necessary to know all the tips and tricks when designing so building is off the table. However, using a single amplifier puts a lot of trust in an unknown cheap amplifier. [33] [34] [35]

Power Supply

The chosen amplifiers need a voltage of 7.2 voltage to output the required 5W. One important thing to consider is power supply noise. This needs to be carefully prevented from the noise of the switch and the internal noise should be kept outside of the bands and the desired IF frequencies and their harmonics. This will be accounted for in the power section.

Preamp

Figure 31. WideBand Controllable Gain LNA [2]

Another thing to consider is intermediate amplification. Because the output of the sdr is so low (about 15 mW) it may not be able to meet the minimum input power for the 5-watt power amplifier. This is also where variable gain could be applied since the SDR does not allow for it. However since variable gain was a stretch goal performance will be the first priority. [61], [24].

2.2.11 Rx Conditioning

The plan is to make the system Half duplex. Although full duplex is possible with the chosen sdr that requires bulky filters for each band adding complexity and size. So, since for our application duplex is not necessary we decided against it. Half duplex will still be a challenge. The output power of the system is 5 Watt which means the voltages could be up to 20 volt peak or 40 volt PkPk. For obvious reasons these voltages could damage the front end of the sdr if proper isolation is not set up.

Because of the importance and complexity, multiple RX protection circuits were considered including Rf limiters, circulators, Duplexers, and T/R Switches. Another thing to consider is the input signal. DSP can do alot but some things are always easy and more effective to implement in hardware for this multiple RX

signal conditioning options. These options were Squelch, noise filtering, and preamps.[17], [15], [12].

RF limiter

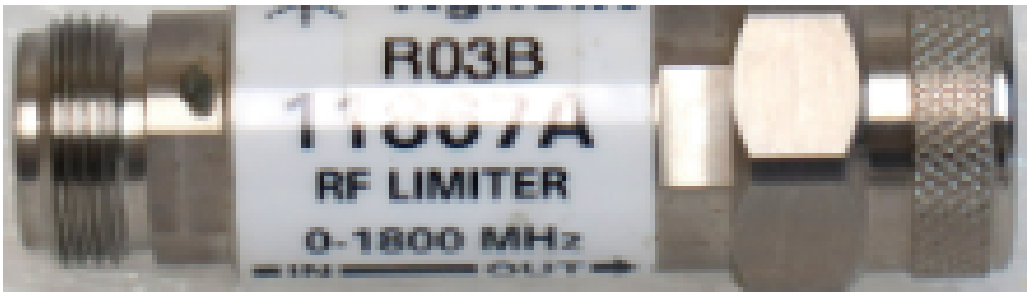


Figure 32. RF Limiter

This component would basically be a variable attenuator limiting the RF passed through to the SDR. The main benefits would be less control circuitry and software control needed to protect the SDR. This is because these are automatic so protection is guaranteed and there is no way to mess this up with misconfiguration. Another key benefit is failure: generally these components fail as an open circuit whereas a digitally controlled rf switch could fail closed or in an unknown state. Another great benefit is low insertion loss. The highest insertion loss I could find was only 0.1 dB: This will sound better when put into the context of the other options.

There is another benefit from a growth standpoint. Because one of the main goals for this project is future open source growth and building the most versatile platform possible, this option may be ideal for it. It would allow for the SDR to receive an attenuated version of what it is sending; this would allow for precise software tuning as discussed in the amplifier section.

There is one major function drawback to this option and that is a non flat response. Meaning at certain frequencies in the Limiters band there may be more or less power throughput. This would have to be accounted for potentially lowering the power throughput just to protect the SDR. Another possible issue from this is volume. At different frequencies the SDR may just be completely silent whereas at other bands the audio output could be dangerously loud. This would require a pre configured software squelch to just even out the non flat response.

All in all, this sounds like the perfect solution from a purely functional point of view. Which to be fair it is, but its drawbacks significantly outweigh the benefits. The main drawback is price: The cheapest Pin diode I could find was 30 dollars which depending on the design our limiter may need 2 diodes. Another drawback is heat; rather than completely block the power, the RF limiter acts as a variable resistor and dissipates a significant amount of power. This power would have been transmitted. Related to the heat issue, another drawback is size. The above picture is 4in long and 300 dollars.

Because of the drawbacks this method was ultimately decided to be a backup at best.

T/R Switch

One possible solution depending on the sdr Tx/Rx may be a RF switch. This would disconnect the receiver when transmitting and the transmitter when receiving. This has a higher insertion loss then an RF limiter but is the cheapest option. One main drawback is the necessity of control signals from the PI. But because these are Cheap Simple and offer a significant amount of control over transmission type we decide these were the best options. Plus the size is significantly better as it is just the size of a small IC.

For digital communications there are generally 2 options for RF switches. The first being RFE MIPI and the second being protocol-less direct digital. Because of the required speed and the lack of hardware support for RF MIPI on the PI Bit banging that protocol would be a significant waste of processing power. Because of his protocol, GPIO is the only option.

For RF switches there are also 3 isolation modes. Absorptive, Reflective, and Hybrid. The Absorptive provides a 50 ohm impedance path to ground for the connections that are switched off. This mode would not be ideal because we would lose a lot of transmitting power in the current configuration. The next mode would be reflective: this mode basically turns the switch off modes to an open circuit which would be dangerous if the power had nowhere to go but since we need the power to radiate through the antenna then this is the configuration we need. Lastly there is hybrid which turns everything reflective but also has the option to turn the antenna connection to an absorptive load preventing amp damage when used. Although this mode is nice it's not necessary. This is by far the best option.

Circulator

Figure 33. Circulator

This is probably the worst option from a cost and size standpoint but it was included specifically because of the additional future potential. A circulator is a 3 port device using 2 ferromagnetic elements. Basically it allows for direct cancellation on one port due to phase mismatch and output on the other. RX would not receive the TX signal at all. This would have issues with insertion loss

and possibly noise but not a significant amount more than any other options. Another issue would be Size and frequency selectivity. This option main would need 1 for each band. This obviously affects the size with each being roughly an inch square. And at around 30 dollars each for our power requirements it is just not feasible.

The main reason for going with this would be for the possibility of software defining duplexing. This is a moderately new method that uses a SDR to achieve nearly perfect duplex without large filters. Basically this method has the SDR produce a symmetrical out of phase signal that can self cancel if summed with itself. This is the holy grail for handheld radios on feature alone but for now the complexity, cost and size are too much for this project so we decided against it.

Squelch

A common feature in many radios is squelch. This feature allows the radio to suppress the received demodulated audio, basically completely cutting off weak signals. A possible way to implement this in hardware would be a variable attenuator. This may be necessary if the received power of something is higher than the receiver can demodulate. Or in other words the receiver is clipping. Another use for this feature is noise deadening but this feature should generally be implemented in software with other noise gates to allow for better audio options.

Filtering

The RX input may have multiple different noise sources, for example a clock in the circuit has a specific harmonic or the LNA is particularly sensitive to the frequencies generated by a car starter. Although the SDR itself will be a shield there are multiple place noises like this could be picked up so after testing a filter may need to be added to the second iteration of the pcb.

Possible LNA

The SDR has a built-in LNA but because the switch that was chosen has an insertion loss of .6-1dB, an additional low noise preamp may be necessary. Any additional conditioning/filtering will add loss to the system. So based on the final performance given the switch in the loop the decision for adding a second LNA to the sdr input will be made, if necessary, after testing. This would not be ideal because of the added noise, complexity, size, and cost.

2.2.12 Battery/Battery Charger

The preferred battery lifetime for the device is five hours and the battery would have to power the Raspberry Pi, the LimeSDR Mini, and any peripherals that are added to the system. For the system, the most versatile battery type we could use would be the lithium ion battery. Lithium Ion based batteries come in many form factors such as a complete portable power station solution or a combination of lithium batteries and a battery charging module. The battery

charging module would charge the batteries using a 5V input source as well as provide information on the voltage level of the batteries which we could then use to calculate the battery life percentage.

We believe lithium-ion batteries are better for our system due to the fact that they possess a higher energy density, voltage capacity, and lower self-discharge rate than any other rechargeable chemical-composition of batteries, meaning this battery can be recharged more on average without burning out and are significantly more stable than alternate compositions. The only reason we would want to choose a lead acid based battery or any other type of battery would be purely for cost, as lithium ion batteries generally cost slightly more than competition.

For the purposes of our project, we have a number of factors to consider when choosing how to implement these lithium batteries. We require a big enough power bank to be able to power this radio for up to 5 hours, constantly powering multiple 5V outputs such as the Raspberry Pi Zero 2W or LimeSDR mini (optionally 3.3V for the LimeSDR mini, depending on which mode we intend to use) for both low power modes (when device is not being actively used) and active modes. These batteries must also power the TX amplifiers and RF switches, which are also currently rated for 7.2V and 5V respectively, obviously requiring to be able to handle the current load produced by all of these components as well.

All in all, the battery system must be able to sustain both 7.2V and three 5V connections for over 5 hours with varying current loads. To accomplish this task, we opted to select two Panasonic's NCR18650B configured in parallel with each other, as connecting in parallel allows us to increase the overall current capacity and amp-hour capacity without increasing the voltage going into the system.

Figure 34. Batteries

This product is available both on amazon and orbtronic for immediate purchase. This power bank of two batteries in series will be sufficient for our watt hour needs and the batteries themselves possess built-in qualities such as PCB protection for overvoltage and overcurrent protection. This battery has the drawback that it cannot be allowed to discharge completely, but we can prevent this problem within our system by turning off the system before the battery is completely discharged. While the battery also requires consistent use to be of

worth, we hope that this project will not encounter long periods of inactivity (such as multiple months of inactivity) or will require new batteries. [57]

For the casing of the batteries, these particular panisonics are a very standard size of 18650 which is very slightly larger than AA battery standards, but have plenty of battery holders available on websites such as digikey or mouser for this purpose. We do not have high requirements for this battery holder, it simply must have the standardized metal prong V+ and V- connections for each battery to hold them in series and be secure around the base of the battery holder.

Most battery holders are very cheap (generally under $10) and the amount of CADding work to design our own far outweighs the price of simply choosing an already created product for this specific purpose. As both of the batteries will directly connect to a power regulation PCB anyway, the holder needs no capabilities to regulate power itself, just simply hold the source of it. As for the accessibility of the batteries, we plan to attach this battery holder to the **back** of the radio, which means if we make the back detachable, then anything sitting on it will also become detachable, thus making the batteries easily accessible and changeable for the consumer.

The battery case we chose has part number “BK-18650-PC4” and is produced by MPD (Memory Protection Devices). It was chosen for its 2 cell size and through-hole PCB capabilities, which lines up with our needs perfectly for the purpose of the battery holder. This case is shown below.

Figure 35. Battery Case [5]

This product is massively in stock on digikey and can be found at the link provided in the references section.

2.2.13 Touchscreen

One of the required features of the Amateur Radio Club’s SDR Radio project is the ability to display information and control features within the software. To do this we will be implementing a touchscreen display that will be connected to the Raspberry Pi Zero. There are many different kinds of touchscreens available for us to use in a wide variety of formats and features.

Ideally the touch screen should be water-proof or at the very least, water-resistant since the SDR Radio's primary application environment is outdoor uses. Resistive touch screens usually perform better than capacitive touch screens due to their double layer screen technology that is used to detect user input; but for prototyping purposes a capacitive touch screen would work just as well as since they utilize a glass screen which is resistant to small amounts of water. We may also develop a waterproofing feature for the entire device which will also aid in the performance of the touch screen module.

Within our price range for the touch screen module, our selection is limited to models that are 3.5 inches large with a fairly low screen resolution. When selecting our touch screen module, one aspect to note is the connection type from the touch screen to the Raspberry Pi Zero. Many touch screen module options that are compatible with the Raspberry Pi can only be connected to the system via soldering the touch screen module directly to the 40-pin GPIO on the Raspberry PI. Once this is done, the Raspberry Pi's GPIO cannot be used for anything else other than operating the touch screen module so this would only be a viable option if in the event we are certain that we will not be needing the GPIO.

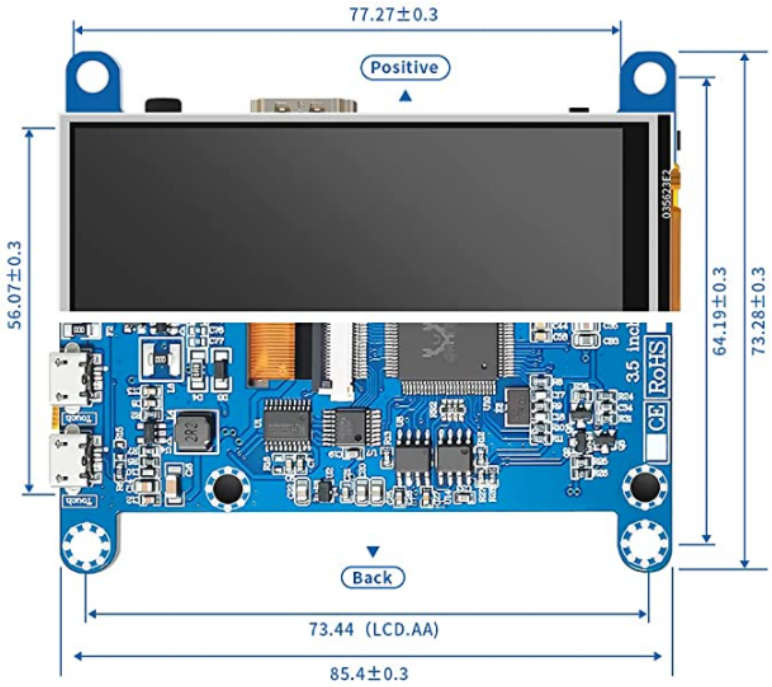


Figure 36. Touch Screen [28]

One of the best options available within our price range is the JniTyOpt 3.5 inch Touchscreen Display available on Amazon.com for $28.99. This touch screen comes as part of a plug-and-play package which includes: the touch screen module, an HDMI cable, HDMI to micro HDMI adapter, micro USB cable, copper mounting screws, and a stylus. Onboard the touch screen PCB are several useful I/O features including: a 3.5mm stereo audio output, a power

button in order to turn off the backlight within the display, and two buttons that act as backlight/volume shortcut controls.

Compared to the iUniker Raspberry Pi Touchscreen, a module whose design is common for Raspberry Pi Zero compatible 3.5 inch touch screens at the same price range, the technical specifications are relatively the same. The main difference between the JniTyOpt 3.5 inch Touchscreen Display and others on the market is that instead of using HDMI for video and audio connection, most others rely on using the full GPIO of the Raspberry Pi without the inclusion of the 3.5mm audio output.

2.2.14 Speaker

The kind of speaker that we decide to go with depends on the device we end up choosing. This is because different kinds of devices will have different audio output capabilities which will affect how we decide to output audio. These could include (in order of difficulty): rerouting GPIO headers, outputting audio over bluetooth, splitting an HDMI signal, connecting to a USB port, or connecting to a 3.5mm audio jack. Because there are so many different options, we have looked into one option for each variant.

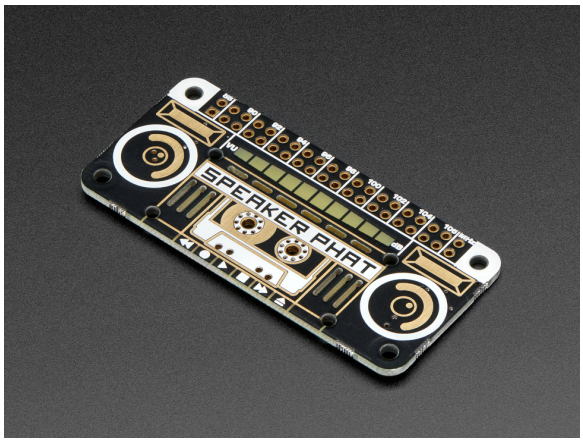


Figure 37. GPIO speaker adapter [42]

Rerouting GPIO Headers

The Pimoroni Speaker pHAT for Raspberry Pi Zero retails at $13 and is designed specifically for the Raspberry Pi Zero. If chosen would be an extremely easy to implement solution for audio output. The main problems with this device are that it has been discontinued, so finding one for cheap would be difficult, and it uses all 40-pins of the Raspberry Pi Zero. This would make it more difficult to attach other devices which would also use the GPIO pins. Therefore this device, and any other device which uses a lot of GPIO pins, is a non-viable option.

Figure 38. Pimoroni [14]

Audio over Bluetooth

This bluetooth audio speaker kit from Amazon is being sold at $16. A bluetooth speaker would make it so that we are not hampered by as many wired connections from the Raspberry Pi Zero to the audio output. Unfortunately, bluetooth speakers would use up a lot more power, while also requiring their own power source. Thus this kit, and any other bluetooth speakers, are non-viable options.

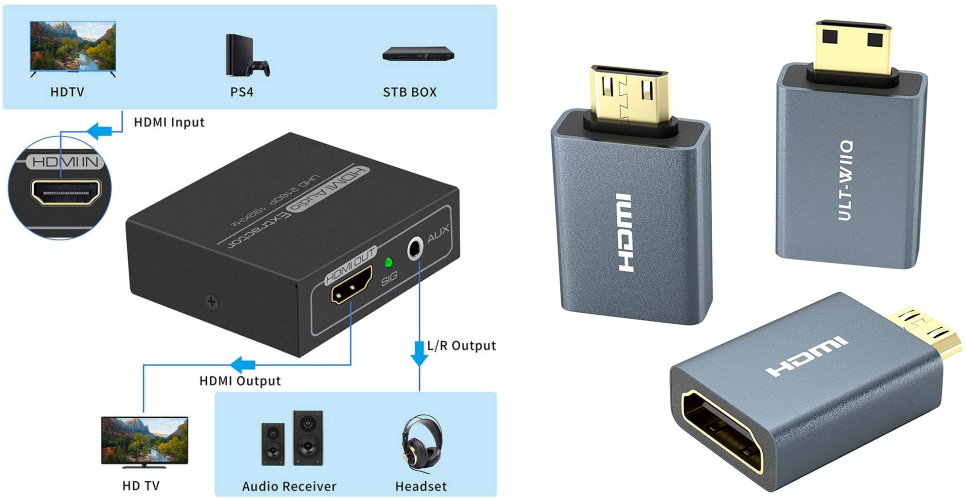


Figure 40. HDMI options [22], [36]

HDMI Signal Splitting

The Raspberry Pi Zero comes with a Mini HDMI port equipped to output both video and audio. If we wanted to make use of that port, we could use a mini HDMI to HDMI adapter, then plug an HDMI cable into that adapter and an HDMI splitter, and finally output our audio through a device connected to the 3.5mm headphone jack. The major downsides to this solution include the extremely high

cost compared to the other solutions, as we would be spending $20 on just the adapter and splitter before even factoring in the cost of the speaker. Additionally, this would take up a lot of space for a relatively unimportant feature.

The major positive to this solution is that it would likely be easy to implement on the software end (assuming sending a blank video signal is easy). The other positive is that it still allows us the option of outputting video through the mini HDMI port. If this solution were only to be used for the speaker, I would consider it completely non-viable. If however on the display end of things we decided to go with a display that requires an HDMI signal, this may be worth considering.

Figure 42. USB Options [25], [31]

Connecting to a USB Port

Using a USB-based device as a speaker would be useful for multiple reasons. Unlike the mini-HDMI port, which requires both an adapter and a splitter, it is very easy to find a cheap Micro USB to USB A Hub for as little as $7. Then on top of that, USB speakers are very common devices as they are used outside of development, so there are a lot of options for what to get. Finally, USB devices are usually extremely easy to find the appropriate software for in terms of drivers, and the extra USB ports from the HUB might be useful for other external devices. Of the options listed so far, I would choose a USB Hub + USB Speaker by far as it comes in at around $20, would have a small footprint on size if we chose a good combination, and would be very easy to handle on the software end.

Figure 44. Speakers [50][11]

Connecting to a 3.5mm Audio Jack

The easiest, cheapest, and simplest option would be to use a speaker that connects to a 3.5mm audio port. This first example comes in at only $8. The biggest downside to this is that the Raspberry Pi Zero 2 W does NOT have an on-board 3.5mm audio jack, so this option would only be available if a different embedded system were chosen. Fortunately, the touchscreen that we have selected DOES come with a 3.5mm audio jack that supports audio out, so we will be able to select a 3.5mm speaker. We have chosen the Degraw speaker as the Tangxi shipping time was much longer than the Degraw speaker.

2.2.15 Microphone

The kind of microphone that we decide to go with depends on the device we end up choosing. This is because different kinds of devices will have different audio input capabilities which will affect how we decide to input audio. These could include (in order of difficulty) rerouting GPIO headers, connecting to a USB port, or connecting to a 3.5mm audio jack. We have considered an option for each variant

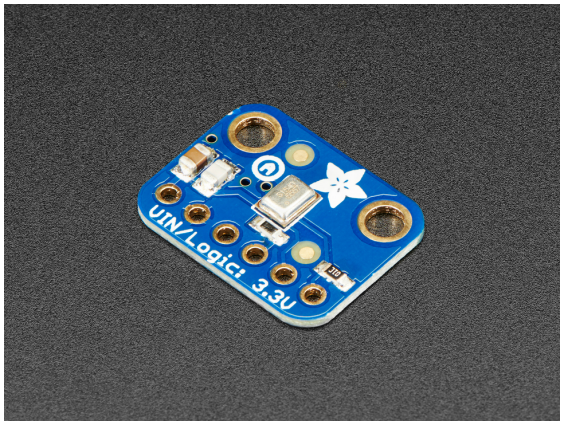


Figure 45. GPIO Microphone [3]

Rerouting GPIO Headers

This microphone retails at $7 and is designed specifically for the Raspberry Pi Zero. If chosen would be an extremely easy to implement solution for audio output. The main problem with this device is the fact that it uses GPIO pins, as many of the devices we may end up wanting to implement may take up all 40 GPIO pins. If the GPIO pins are not in high demand, this would be an excellent solution as it is extremely small, cheap, easy to find, and easy to implement.

Figure 46. USB Microphone [29]

Connecting to a USB Port

Using a USB-based device as a microphone would be useful for multiple reasons. USB Mics are very common devices as they are used outside of development, so there are a lot of options for what to get. Additionally, they are extremely cheap, as the microphones shown above are sold for $4.50 each. Finally, USB devices are usually extremely easy to find the appropriate software for in terms of drivers. Unfortunately, as the Raspberry Pi Zero 2 W is very limited in terms of USB ports, we would require a USB Hub to effectively make use of this. Additionally, for this specific USB microphone it would be difficult to get clean audio from the user as we would have to have the device exposed. Overall, this option is worth considering, especially if due to a decision in terms of a different part we are already forced to use USB ports.

Figure 47. 3.5mm Microphone [9]

Connecting to a 3.5mm Audio Jack

Similar to the USB port based microphones, this 3.5mm Audio Jack based microphone is very cheap, simple to use, and is low cost at around $11. Unfortunately, the Raspberry Pi Zero 2 W does NOT have an on-board 3.5mm audio input jack, so this option would only be available if a different embedded system were chosen.

3. Design Constraints and Standards (10-15 pages)

3.1 Constraints

Constraints are determined by the team to be the most significant logistical challenges to the project's success but also serves as a guideline throughout the research, design, prototyping, and development phases of the project. The factors outlining the constraint categories were first determined by the Senior Design administration and then refined to be applicable to our project with the Amateur Radio Club's end goals in mind as well as our project requirement specifications, availability of components, and other circumstances. By creating these constraints we are able to refer back whenever a roadblock is met to decide the best course of action.

The key constraints that we will focus on during the duration of our project are design, economic, time, manufacturing, sustainability, ethical, health, safety, environmental, societal, and political. Some constraints work hand in hand and therefore can be viewed as categorically similar such as the health and safety constraints or societal and political constraints.

Design Constraints

Design constraints are factors that either negatively affect our project design or provide us a challenge, both of which will add to our knowledge and skills within the field of radio technology and computer/electrical engineering. Many of the constraints presented throughout the design section affect which components we will ultimately choose as well as provide us a basis as to what the physical characteristics of the device will be.

The physical design constraints of our project comes from the very nature of our device. Since the device is meant to be portable, it cannot be too cumbersome to carry both in the hand as well as in a carrying case such as a backpack. It also should not be excessively larger or heavier than current portable software defined radios. In addition to the size parameters of the device, many applications of portable software defined radios are held in an outdoor environment and therefore the product needs to be durable to a certain degree. This should include: remains undamaged after any minor fall, screen is resistant to mild water droplets, water resistance to rain, and stability of mechanical components after extended use.

The particular components we choose are necessary but ultimately arbitrary since there are many different combinations of parts we could use in order to accomplish our task. Nevertheless, many of the components we need to create the device generate a fair amount of heat energy since they are performing rigorous tasks on small motherboards and have to be placed near another component that is also generating heat. Notably these are the processor

of the embedded system, the RF amplifier/switches, as well as the processor and the screen of the touch screen.

Normally, we would use heatsinks on any microprocessor that is generating heat, but in doing so we will run into a number of problems. If we put a heatsink on all the processors the size of the device will increase drastically and the placement of the heatsinks might interfere with each other. If we simply get smaller heat sinks, then the amount of heat we are dissipating will not justify the additional cost. Even implementing a heatsink on the RF component proves difficult since it is a relatively high wattage device in an extremely small package. The heatsinks for this type of component are usually made from larger amounts of metal leading to increased size and weight; oftentimes up to five times the dimensions than the original component. The only way we can overcome this challenge is to selectively place our heatsinks as well as providing any mechanical advantage we can, for example, the addition of non-obstructing vents to the casing of the device.

Another key aspect of our project is that it is to be open source. Due to this, we are aiming to keep the use of more obscure or outdated programming languages at a minimum so that we reach a wide target audience of entry level programmers and amateur radio enthusiasts. For example, many students and engineering professionals know the C language well, but someone who is just entering the field of amateur radio will most likely start with Python or a similar object oriented programming language and they will likely be deterred if they have to learn a complicated language to upgrade their device.

Economic and Time Constraints

One of the main constraints during Senior Design 1 and 2 determined by our team is managing our time in accordance with the duration of the spring and summer semesters. Everyone in this team works at least twenty hours a week, so efficient usage of time is a major obstacle to the success of the project. Instead of having sixteen weeks for research and preparations in a fall semester and fourteen weeks of design and corrections in the spring with almost a month of break between terms, we have fourteen weeks during the spring semester for the first half of the project and roughly twelve weeks in the summer for the second half with less time separating the two semesters. To overcome this we are going to have to purchase our components and start testing them earlier than we normally would as well as have frequent meetings with ourselves and our project advisors/consumers to ensure there will be no unexpected challenges.

Although funded by the Amateur Radio Club, the total cost of components should not exceed five hundred dollars. This will include our prototyping and development costs as well as shipping. Outside of project funding, in order to develop and test the project some team members may be required to obtain an Amateur Radio License. Costs of the license examination range between fifteen and fifty dollars depending on what level of certification needs to be achieved. To alleviate the cost of components during our prototyping phase, we will use the facilities available to engineering students at the University of Central Florida as

well as the lab spaces of the Amateur Radio Club. This will give us access to a wide set of high-quality tools and through hole components necessary when completing the breadboard development phase.

Manufacturing and Sustainability Constraints

The second most important constraint is component availability. The components must be widely available and mostly unaffected by the current semiconductor shortage. For this reason popular parts should be used. In addition to this, many major PCB manufacturing companies receive an influx of orders during the end of each semester due to the amount of students from colleges and universities around the world ordering parts for their senior design projects as well as all the orders they would normally accept from other companies and manufacturers. This leads to a shortage of components that already adds to the global microprocessor shortage and delay in order completion.

When taking into account sustainability, we look to existing devices available on the market. Many standard handheld radios and portable software defined radio solutions rely on disposable batteries and thus are not sustainable for two reasons: the cost of constantly purchasing replacement batteries and the waste generated by the disposal of such batteries. Therefore we have decided to implement the rechargeable battery feature found in higher end portable radio models. By choosing the best component within our price range within each category, we ensure that the device will have a long life without needing to be repaired constantly.

Ethical, Health and Safety Constraints

In regard to health and safety constraints, most of our challenges lie in our power delivery system. Due to the use of 18650 Lithium Ion Batteries, we must ensure the safety of the user by implementing measures that protect the batteries from any of the following scenarios: overvoltage, undervoltage, short-circuit, overheating, being submerged in water, and excessive humidity. If these conditions do occur the batteries are at risk of explosion which can result in extreme physical injury and damage to the surrounding area. The flame produced by the battery is a special kind of electrical fire that can be extremely dangerous without the aid of a class D fire extinguisher. (Brooks)

Ethically, our constraints are mainly due to radio and software regulatory practices and responsibility falls on both us as the developers and the end user of the device. Users are expected to use our device within compliance of FCC regulations. Regardless of this, if in the event highly restricted wavelengths are not blocked by the radio receiver, we will be implementing limitations to ensure a user cannot accidentally enter confidential radio space. Designing the software component of the software defined radio will also require the usage of various third-party extensions and programming language add-ons. We will need to ensure that we are following best copyright practices as well as obtaining all necessary licensure to use any data, figures, and components.

Environmental, Societal, and Political Constraints

Since our device is designed for indoor and outdoor usage, the environment does not provide many constraints while designing our project. Radios are robust devices capable of being used in the most remote areas imaginable and during the harshest natural disasters. Although the environment itself cannot provide any major challenges we can test for within the allotted time frame of the project, pollution creates an interesting issue.

One major constraint to our project is the disposal of Lithium Ion batteries. They are considered hazardous waste and therefore must be brought to a certified electronics recycler instead of being disposed of in residential or commercial trash and recycling bins. Lithium based batteries themselves contain less toxic components than other traditional battery types but lithium mining practices contribute negatively to the environment in forms of pollution, deforestation, and carbon dioxide emissions. To alleviate this we plan to use highly reputable 18650 Lithium Ion batteries that are long lasting, durable, have a high number of recharges, and are efficient so that they do not need to be replaced as often as other forms of battery technologies. [60], (Brooks).

Many of our societal and political constraints arise from the laws and regulations of engineering and radio technology. Each state has their own laws governing the ownership and usage of radio equipment and broadcasting therefore we as developers must ensure that we create a device that is within these legal limitations.

3.2 Related Standards

Standards are the guidelines that govern any reputable device and are implemented by companies, governments, and non-profit organizations in order to keep technology safe and accessible for all that use it. In addition, standards can describe common hardware features and communication protocols shared between devices. In the following sections we will describe the standards that we have imposed upon ourselves so that we can give the Amateur Radio Club a desirable finished product as well as standards set by our professors and organizations that provide us a challenge and the credentials to pass this course.

3.2.1 FCC Standards

Must comply with transmitting noise and power standard set by FCC CFR Title 47:part 97
Including but not limited to:

- Angle modulation must have a modulation index less than 1.
- No non phone emissions shall exceed the bandwidth on phone quality emissions using the same modulation scheme
- The total bandwidth of side band emissions shall not exceed A3E emission quality which is full quality dsb amplitude modulation of voice in

low frequencies with spurious side band amplitude at a maximum of what is defined in FCC CFR title 47 part 95.635 unwanted radiation
- The baud rate of radio teletype emissions must not exceed that which is defined in part 97 for each band
- The meaning or intention of any transmission not use for direct control of a satellite as defined in part 97 must not be obscured
- All components must meet EMI standard defined in fcc part 97
- The last constraint we considered is the requirement of amateur licenses by the FCC to transmit on amateur bands one of the group mates already has a license but because of this transmission testing must be done with a liscensed amatuer present. So other teammates may need to get a license

3.2.2 I/O Standards

USB Standards

The project will utilize USB ports in a number of applications ranging from connecting a usb hub, touchscreen, and the LimeSDR to the embedded system. The Raspberry Pi Zero has onboard micro-USB input ports capable of transmitting data; one of which is solely used for the input power. Micro-USB is of the second generation of USB technology and has a maximum transfer rate of 480 Mbps. For second generation USB devices, three speed levels are used to categorize the device and are as follows: Low-speed, Full-speed, and High-Speed.

Low-speed and Full-speed devices like our microphone use 0-0.3 V to recognize a low logic level and 2.8-3.6 V to recognize a high logic level. These signals are transmitted via a data wire pair in half-duplex mode. High-speed devices such as the touch screen and radio receiver use the same data lines but with low level logic signals ranging from -10 to 10 mV and high logic signals ranging from 360 0 440 mV. Ground is terminated at either 45Ω or 90Ω. Since we are using a USB bus to host a number of components, we are going to have to try to allocate resources to the most taxing processes (the touch screen and radio receiver) and disable minor components when they are not in use.

Alongside the maximum data transfer rate, second generation USB technology also has a potential bottleneck due to its power standards. Compared to current third generation USB technology which boasts a maximum power transfer of 240 W (48V/5A); USB 2.0 has two primary modes it can vary between. Low power devices have a maximum power of 0.5 W (5V/100mA) and high power devices have a maximum power of 2.5 W (5V/500mA). Since we are using a USB bus, the voltage of the bus supply has the potential to drop to 4.4 V.

Bluetooth Standards

One of our optional requirements as described by the Amateur Radio Club was to have Bluetooth compatibility. Standard bluetooth communication operates at a range of 2.40 and 2.4835 GHz, and has band protection at the 2MHz and 3.5MHz frequencies. While transmitting data, bluetooth uses

frequency-hopping to send packets of data to one of almost 80 designated channels. The standard bit rate of this transmission is 1Mbit/sec but newer technology has increased this limitation to 3 Mbit/sec. Several protocols have been adopted to support the bluetooth interface including Point-to-Point, TCP/IP/UDP, and Wireless Application Protocol.

Unrestricted access to the Raspberry Pi Zero over bluetooth could result in unintended behavior from the device due to outside interference. Due to this, until proper bluetooth support is implemented we will likely simply be disabling bluetooth functionality on the device to prevent any unforeseen circumstances arising from bad actors attempting to access the device and making it run arbitrary code.

Video Standards

The Raspberry Pi Zero w2 has an onboard mini High Definition Media Interface (HDMI) port in compliance to HDMI standards capable of transmitting video/audio data to the touch screen of the device. As such, it is in compliance with the EIA/CEA-861 standard which sets requirements and regulations upon electrical/peripheral devices including cables for the use of uncompressed digital interfaces. [8]

Since the Raspberry Pi Zero is a lightweight embedded system its graphics capabilities are limited to 1080p and we are able to use a standard Category 1 HDMI cable and HDMI to Mini HDMI adapter to display video instead of a more expensive Category 2 HDMI cable. With this we are able to display video at a quality of 1080p60Hz, although the Raspberry Pi is limited to 30 frames per second (fps). This is much more than enough for our intended use case, and we may end up outputting video at a lower resolution to save on processing power. This is because higher screen resolutions have a less pronounced effect on small screens which are being seen up-close, which is exactly what our radio would be.

Audio Standards

The speaker system of the device uses a 3.5mm TRS audio input cable that connects to the touch screen. TRS is an acronym for Tip, Ring, Sleeve; and refers to the three sections on the tip of the cable that provide a connection to the left audio source, right audio source, and the ground. Via the TRS standard, we have the option of playing back audio in either the mono or stereo format.

We are likely going to use a USB microphone for our input data and as such we can expect the component to follow either one or all of the current USB audio standards. UAC 1.0 devices are those which are multi-platform compatible. UAC 2.0 devices encompass High and Full speed USB devices which are used for applications that require high sample rates such as audio functions. UAC 3.0 devices burst data in order to constantly reside in a power saving mode and will shut down when not in use. Three synchronization modes

are also available standard to USB technology and are asynchronous, synchronous, and adaptive.

According to UL 1492, which outlines audio/video devices in accordance with NFPA 70, our device fits the requirements of being a device that reproduces audio including radio products, radio receivers, and amplifiers. Since our product's target audience is amateur radio enthusiasts the lowest probable age of our consumer would be a teenager; we must also be in compliance with UL 696, the Standard for Electric Toys. The Standard for Information TEchnology Equipment Safety (UL 60950-1) and the Standard for Audio/Video, Information and Communication Technology Equipment (UL 62368-1)

The aforementioned UL 696 would include standards related to the packaging of our device if it were to be mass-produced. One exception in those standards is that it does not apply to outdoor devices. Due to one of the goals of our devices being water-proofing, an argument could easily be made that the device should be considered an outdoor-device, and thus it would be exempt from UL 696.

The other 2 standards could have most of their requirements met as long as we follow common protocol when it comes to minimizing any danger that our device might produce. For example, we would be required to make sure that there are no chemical hazards, heat hazards, or electric shocks being produced by our device. Concurrence with these standards should be considered outside of the scope of our portion of this project, as it would require a lot of testing to justifiably say that our device is not a risk for any of these. Additionally, even if our team went through with that repetitive testing, any further development on the device that adds hardware components would invalidate those tests, and thus our efforts would be wasted. Due to this we will not be spending an excessive amount of time verifying that we are meeting these safety standards.

Power Standards

Since we are using lithium-ion batteries, we must ensure that the manufacturer of the battery cell is reputable and follows all standards so that we meet all health and safety requirements for our project. One of the most important standards for lithium based batteries is the UL 1642 certification that specifies requirements that must be met when using rechargeable lithium-ion batteries as a power source. Based on this certification, we know if the manufacturer is producing battery cells that pass the following tests: low pressure, temperature cycling, heating, shock, vibration, crush, impact, abnormal charging and short-circuiting.

The batteries that we chose to buy have this aforementioned certification. By virtue of having this certification, we can more safely claim that our device is able to withstand wide ranges in temperature. This is important because if even one part of our device, such as the battery, is unable to withstand temperatures past a certain point, then the entire device would have to be labeled as not being able to withstand those temperatures. [57]

GPIO Communication Standards

Our design of the device aims to use as few GPIO pins on the Raspberry Pi as possible so that we can make changes as we see fit while developing and prototyping and also to maximize the modular potential of the device for the open source community. The main components of our device that can be switched out for a similar component without compromising the integrity of the core functionality are the microphone and the speaker input. Additional modules could also be added to enhance functionality such as a GPS, thermometer, gyroscope, and much more.

Many sensors and modules use specific standards in order to communicate to the Raspberry Pi for instance: I2C, SPI, and UART. Upon setup of the Raspberry PI we will ensure that all applicable communication standards are enabled and that the consumer will be able to modify the device as they see fit. However, for the Mosfets and RF switch, directly driven GPIO will be used for easy control. The raspberry pi standard for Digital GPIO is 3.3v at a maximum current of 16mA. The Speed of the GPIO is then defined by the processor which for our purpose should be fast enough for what we need.

3.2.3 SPI Communication Standards:

Serial Peripheral interface is a 4 wire duplexed synchronous one to many serial communication standard. Commonly used in embedded system communicating subsystems are broken up in master and slaves. Although both subsystems can trasnmit information uninterrupted master subsystems can initiat communication by addressing a specific slave whereas slaves need to be addressed first. The wires used are MOSI, MISO, SCLK, and CS. MOSI transmits from the master to the slaves MISO transmitts form the Slave to the master. SCLK is a clock used to synchronize communication output by the master. CS is a line held either high or low by the master to address a specific slave interface.

3.2.4 CTCSS standard subaudible tones

Continuous Tone-Coded Squelch System or CTCSS is a single frequency multi user protocol allowing for multiple different subaudible tones to transmit on the same channel at the same time without interference. This can be considered a type of in-band signaling that is used to allow two users to transmit without listening to other users on a shared radio communications channel. This is sometimes also called tone squelch. To accomplish this it adds a low imperceptible frequency audio tone to the voice. This allows more than one group of users on the same radio frequency (sometimes referred to as co-channel users).

The CTCSS circuitry mutes users who are using a different subaudible CTCSS tone or no CTCSS creating virtual sub channels. This happens without creating any additional actual channels. Actual channels would be on separate frequencies but with Tone Squelch all users with different CTCSS tones are still transmitting on the identical frequency This has the drawback over traditional channels of having transmissions interfere with each other; however; the

interference is usually imperceptible. Because the CTCSS is just designed for working on separate tones there is no security allow for this to be used in amatuer radio according to FCC part 97.

3.2.5 Dual-tone multi-frequency signaling

Dual Tone multi-frequency signaling (DTMF) or sometimes referred to as the trademark touch-tone is a dual tone audio standard for transmitting 16 symbols with 8 audio frequencies. Commonly used in telephone systems for transmitting numbers this standard is a type of Multi-frequency. This semi-automated signaling was built for a complimentary automated switching for both speedy and cost effective phone routing. This standard is defined by multiple AT&T standards.

Our system will likely not support data-transmission for symbols such as numbers over radio waves, so we likely will not be specifically attempting to support this standard as the radio is not meant to transmit numbers.

3.2.6 JTAG

The goal of the project is to eventually Have a configurable FPGA for novel Modulation, encoding, and transmission schema like JT65 and/or Codec 2. JTAG is a debug/programmer standard for embedded systems. It will eventually be used to program the FPGA. To go into more detail JTAG is a hardware interface and communication protocol commonly used to debug programs and upload in production/ testing environments. Originally developed by a consortium, the Joint (European) Test Access Group, in the 80s to solve the new and daunting challenge of debugging and mass producing printed circuit boards (PCBs). JTAG has been in almost everything since it was first used in the Intel 80486 processor in 1990. In that same year IEEE introduced standard 1491 that same year.

3.2.7 Frequency Modulation

Frequency modulation is a modulation schema where data is encoded in frequency variation of the RF carrier. This is traditionally done with a nonlinear frequency mixer but in our case it can be synthesized directly with the SDR. The current standard for narrow band fm Channels is 15khz variation with 2 khz padding on either side for frequency drift and intermodulation.

3.2.8 Single SideBand Modulation

Single-sideband modulation (SSB) or single-sideband suppressed-carrier modulation (SSB-SC) is a type of modulation used to encode analogue or digital Data to a wave, And example would be audio signals, Encoded onto by radio waves. This modulation is a refinement of Traditional Amplitude Modulation the Mainly refining the needed transmitter power and bandwidth by increasing the efficiency. Traditional Amplitude modulation produces an output signal with a

bandwidth that is twice the maximum frequency of the original unencoded baseband signal. Single-sideband modulation uses half of that or the bandwidth needed is the maximum frequency of the unencoded transmission. This of course increases device complexity and adds complexity tuning at the receiver.

In AM transmitters Mix and RF carrier with the baseband signal in the final RF amplifier this is known as high level modulation. This results in the original signal to be encoded on both Side bands of the RF carrier. Obviously this is not necessary; a suitable receiver should be able to extract the entire original signal from either the upper or lower sideband. The main disadvantage is SSB Modulation must be done at a low level and amplified in a linear amplifier Which can be expensive. But it allows for better amplifier efficiency. Another disadvantage is SSB reception requires a significant amount more frequency stability and selectivity beyond that of inexpensive AM receivers. This is the main reason broadcast radio still uses traditional AM. So the Main use of SSD is In point to point communications Because Receivers can get more expensive

3.2.9 Soldering

While prototyping and developing our software defined radio we will encounter soldering by hand in a few instances. During these times it is crucial that we follow the best practices outlined by the National Aeronautics and Space Administration (NASA) and the Institute for Printed Circuits (IPC) in order to keep ourselves safe from inhaling toxic chemicals, unwanted burns, and other accidents. Following these guidelines will also ensure that the contact between components are durable and acceptable to prevent short circuits, burning the components or motherboard, and disconnection.

NASA-STD-8739.3 provides the documentation on how to safely and efficiently. They recommend that the work area be cleaned and organized before starting work and ensuring that the environment is controlled to limit accidents and contaminations. Both NASA and the Occupational Safety and Health Administration (OSHA) urge that a ventilation system be used whenever a soldering station is on and heated due to the toxic vapors that come from lead based soldering tin. Once ready to begin soldering, electrostatic discharge should be performed on a ground material to avoid component damage and all tools should be inspected to ensure that they are all functioning properly. [38]

3.2.10 Software/Hardware Standards

For software standards, we will be combining common standards that have been developed by major corporations as well as self-created standards based on the knowledge and skills we have acquired throughout our education and employment. By creating a set of best practice and development guidelines before we start any programming, we provide the end user with clear and concise code that they can extrapolate on as well as safeguarding ourselves from having differing programming syntaxes and errors in the program.

Furthermore, implementing software standards will aid us in avoiding non-productive work, reducing risks, reaching quality goals set by the Amateur Radio Club, and relieving some of the burden of our time constraints.

The main organizations we will draw our software standards from are the International Organization for Standards (ISO) and the institute of Electrical and Electronics Engineers (IEEE), both of whom are global leaders in the production of international standards. Leading experts from around the world come together at committee meetings held by these organizations in order to regulate the field of technology for the advancement of humanity. From the International Organization for Standards, we will be utilizing the ISO 9001, ISO 12207, and ISO 15504 standards. These standards govern the overall quality of the product we are creating via means of constructing a robust software life cycle process as well as setting a model as to how we should work productively.

The ISO 90001 Quality Management standard can be implemented without obtaining the certificate of authentication and provides the guidelines on how to verify whether or not our decision making process is concrete. This standard places an utmost importance on the communication, planning, and teamwork aspects of development. By aiming to exceed customer goals with consistent meetings, being transparent with obstacles and accomplishments, and providing engineering feedback we are able to better direct our workflow to key tasks and streamline areas such as part selection.

In order to be prepared and minimize the risk of redesigning our software framework, a meticulous procedure of engineering based problem compartmentalization will result in a software process that manages multiple interrelated components and programs. Throughout the development and prototyping phases of the project, we will take from the ISO 9001's standards on improvement and evidence-based decision making. While testing our components we are going to take note of performance and all drawbacks not listed in the specification parameters. Via the data we gather while testing we are going to be able to make informed decisions on whether or not a component needs to be swapped out for a better one, if we can use a lower cost component to aid overall cost, and see if there are any areas we can make more efficient via the use of higher level functions.

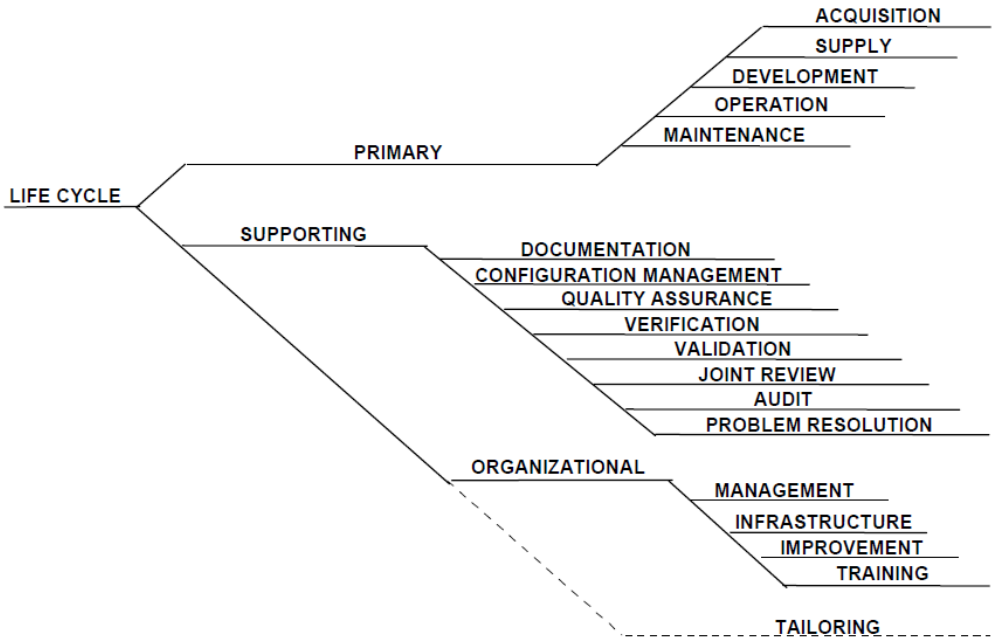


Figure 48. EIA/IEEE 12207 Process Tree

The ISO/IEEE 12207 systems and software engineering - software life cycle processes standard was developed to outline a software life cycle process that can apply to enterprise level solutions. Even though we as a group do not plan to commercialize the device, what the Amateur Radio Club decides to do is at their discretion. In order to make as smooth of a transition of technology as possible we plan to research, prototype, and develop with scalability in mind in terms of both our hardware components and software.

The figure above depicts the EIA/IEEE 12207 process tree where the software process life cycle is divided into three sub-trees consisting of factors that comprise the primary workload, support workload, and organizational workload, respectively. By completing each factor within the subcategories, we ensure that the project will be completed in an efficient and effective manner. Towards the end of the process life cycle we can start focusing on the optional tailoring branch which encompasses making any final adjustments to the device before it is time to present. During the tailoring portion we will also place an importance in completing any unfinished stretch goals as described by the end user. Specifically this will encompass tasks such as waterproofing our device, adding the waterfall/panadapter feature to the software defined radio program, a data storage ability, and global positioning system support.

A procedural representation of the process tree is depicted below in the ISO/IEEE Primary Process Flow Diagram. The separate stages of the primary process flow diagram show the main components of the process tree and can be extrapolated further to show correlations between each of the factors needed to be successful in our project. We can refer to the project milestones provided to us by the administration in terms of this process flow diagram by linking the separate phases of Senior Design 1 and 2 to these components.

The acquisition phase of the project is the beginning of Senior Design 1 when we select our project and covers the verification/validation by professors as well as a joint review with the Amateur Radio club. Until the end of Senior Design 1 we are in the supply and development phases of the project and can complete our initial documentation, component selection,and configuration management. During this time we will also undergo the steps of the acquisition phase again with the addition of quality assurance of hardware and core functionality of the software aspect of the project. At the start of Senior Design 2 we will have a functioning prototype to further our development. We then can enter the operation phase and resolve any problems that arise during testing. The end few weeks of Senior Design 2 we are going to have a fully functional device and enter the maintenance phase to ensure we have repeatable and demonstrable functions for the final presentation showcase.

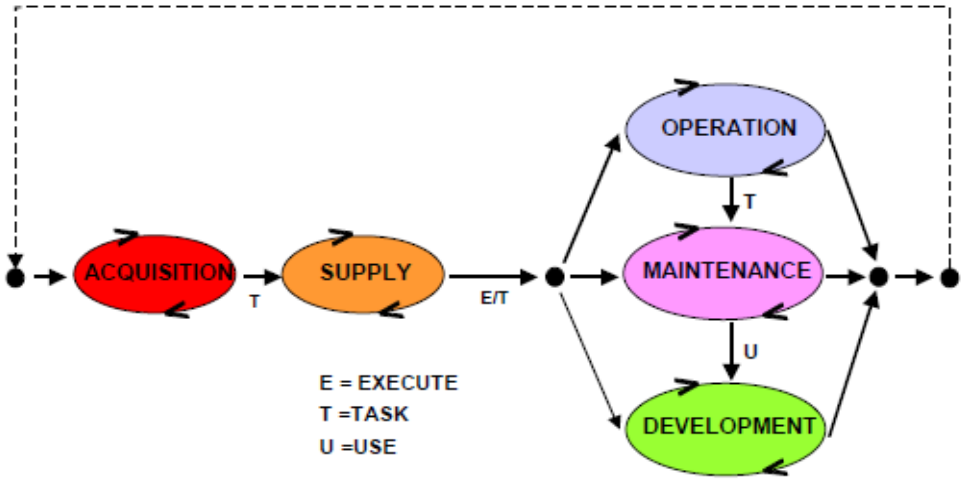


Figure 49. ISO/IEEE Primary Process Flow

The ISO/ IEC 15504-5 Information Technology Process Assessment enables us to self-critique the processes we have outlined above. There are a few components of the process tree that rarely apply to us such as the Audit and Training factors. Since we have a bill of sale the audit can only be updated once new parts are introduced and in order to complete this project all members are supposed to have the necessary level of knowledge in electrical and computer engineering. Instead of removing these sections altogether we have revised the audit to consist of a mandatory review and editing meeting where potential problems in the documentation and development can be addressed. This specification also guides us on which characteristics of our software defined radio we can demonstrate during the final presentation showcase and how they are relevant to the engineering curriculum.

4. Design (25-30 pages)

4.1 Hardware Design (15-20 pages)

RF Subsystem

The RF frontend will work as follows: The SDR will be connected to the power and RF switch board which will be controlled by the pi. Testing may determine that the RF switch should be controlled by the SDR GPIO but for now the Raspberry pi will control them because it is the simplest to implement. From here there will be two modes: transmit and receive. For the receive mode the power to the amplifiers and the input switch shall be turned off. Then the RX switch will be turned on allowing the sdr to receive.

For the transmit mode the receive switch will be off, the band will be selected and the correct amplifier will be powered. This ensures the SDR will never be exposed to the full power of the power amplifiers and the amplifiers will not waste power while in standby. One issue is push to talk latency. We want the push to talk button to almost instantly turn on transmt because of this it may be necessary to either make that control the system from an analog circuit or make that a priority when writing the software. Another thing that could change the final design is receiver loss and noise

To achieve the discussed achievable gain there will be multiple options that will be possible. Firstly the power to each amplifier will be controlled by mosfets allowing for thiem to be shut off when not in use. The power going to the Power amps will also have a controllable voltage as discussed in the power system this will control the power gain for those amps between 2.5 and 5 Watts. This will need to be further tested

The second method for controllable gain will be the RF switch. The RF switch is a SP4T switch that allows the pi to select between direct SDR output. The LNA preamp output or RX. The 4th switch connection will be connected to a SMA adapter for possible 900MHz options. This wont allow for perfect continuous gain control but it will assist in the other options when amplifier backoff becomes an issue.The third Gain control method will

be controllable power output of the SDR. This will allow controllable input into the antenna directly from the LNA of the LNA to the power amps. The internal Amplifier of the SDr is not well characterized and the dynamic range it gives will have to be tested. But since perfect gain control is a stretch goal, if some of these options fail the output power will still have enough variation to be useful for whatever the use case.

Another possible addition is a filter on the output of the PreAmp or the output of the SDR directly. Because of the strict EMi regulation put forth by the FCC the EMI out of the band of transmission needs to be below the thresholds described in design constraints. One possible noise source is the SDR itself because of the frequency of the DAC it is possible that there are high frequency

Noise Bands caused by the RF synthesis itself. This is most likely filtered by the SDR modules itself but this concern was mainly the sponsor's.

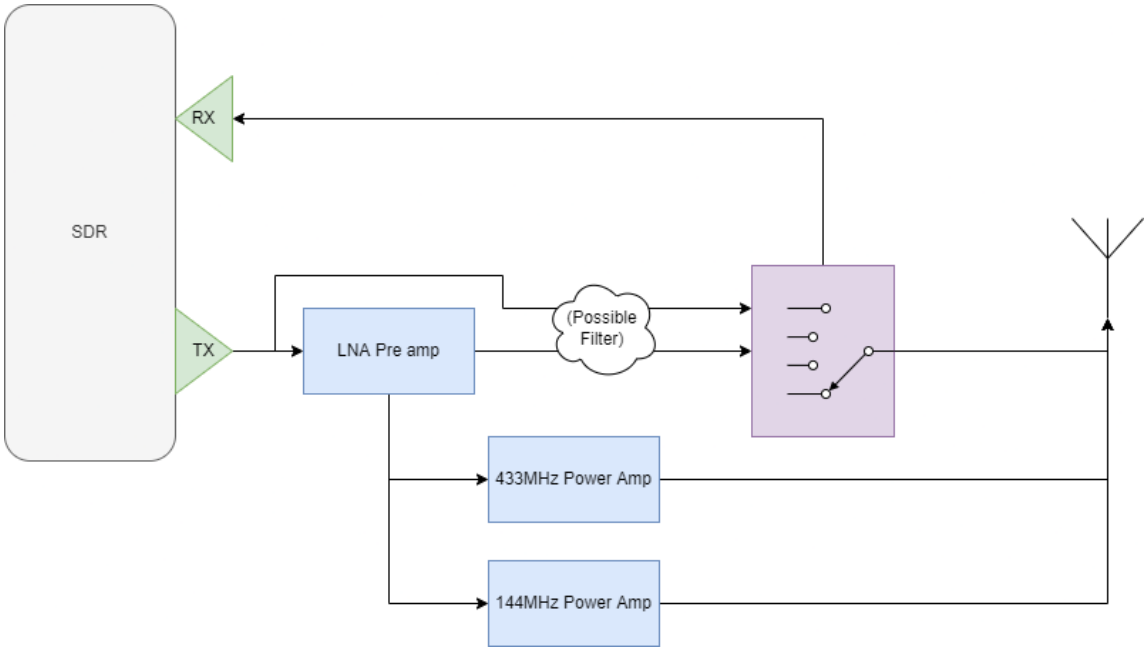


Figure 50. RF Front End Block Diagram

Digital Subsystem

The digital system works with 4 communication types excluding the internal communication of the SDR which may have to be interfaced with later based on testing. The first and main communication protocol is USB. With USB the Microphone, the SDR, and the Touchscreen will communicate with the pi. Because the pi only has one usb port everything will pass through a USB Hub The USB hub will have the Pi as the only host hopefully decreasing overhead.

The second way the digital system will communicate will be over GPIO. This is by far the simplest and will generally operate in one direction. EG the buttons will send the Pi there state the Pi will never respond. Likewise the RF switches will be controlled by the Pi but from a completely analog perspective the Switches will just receive something like a High for on and a low for off.

The third Protocol is HDMI: The HDMI will carry the Video signal to the screen and the audio signal to the speakers. This option is ideal because the Raspberry pi has a built-in HDMI transceiver which will allow for minimal required onboard processing unlike certain I2S or GPIO audio options.

The Fourth and final Protocol is SPI: This protocol will be used to connect to the monitoring ADC. The Monitoring ADC will be used for measuring the Volume, gain adjustment, Output Power, and battery level indicators. The volume and Gain knobs will just be potentiometers. The battery output will require a voltage divider to shift the levels to within the chosen ADC's range. The Output power monitor will be connected directly after the mosfets to monitor the voltage

going into the power amps. This presents the issue of potentially non linear power output vs input voltage curve of the power amps. So to rectify this concern the power output will be tested with an RF meter. This protocol also has onboard controllers but it will take a bit of processing power and latency to run tests so for the software design this will be a separate SO file that will just be called every 30 seconds or so. If this does not allow for real time monitoring of volume and Gain/squelch the ADC may end up getting monitored directly by the SDR gpio pins.

V1	V2	V3	State
0	0	0	External SMA
0	0	1	Direct TX connect
0	1	0	LNA
0	1	1	RX Mode
1	0	0	All Isolation/Test Mode
1	0	1	50 Ohm Sync
1	1	1	Shutdown mode

Table 19. Digital System Block Diagram

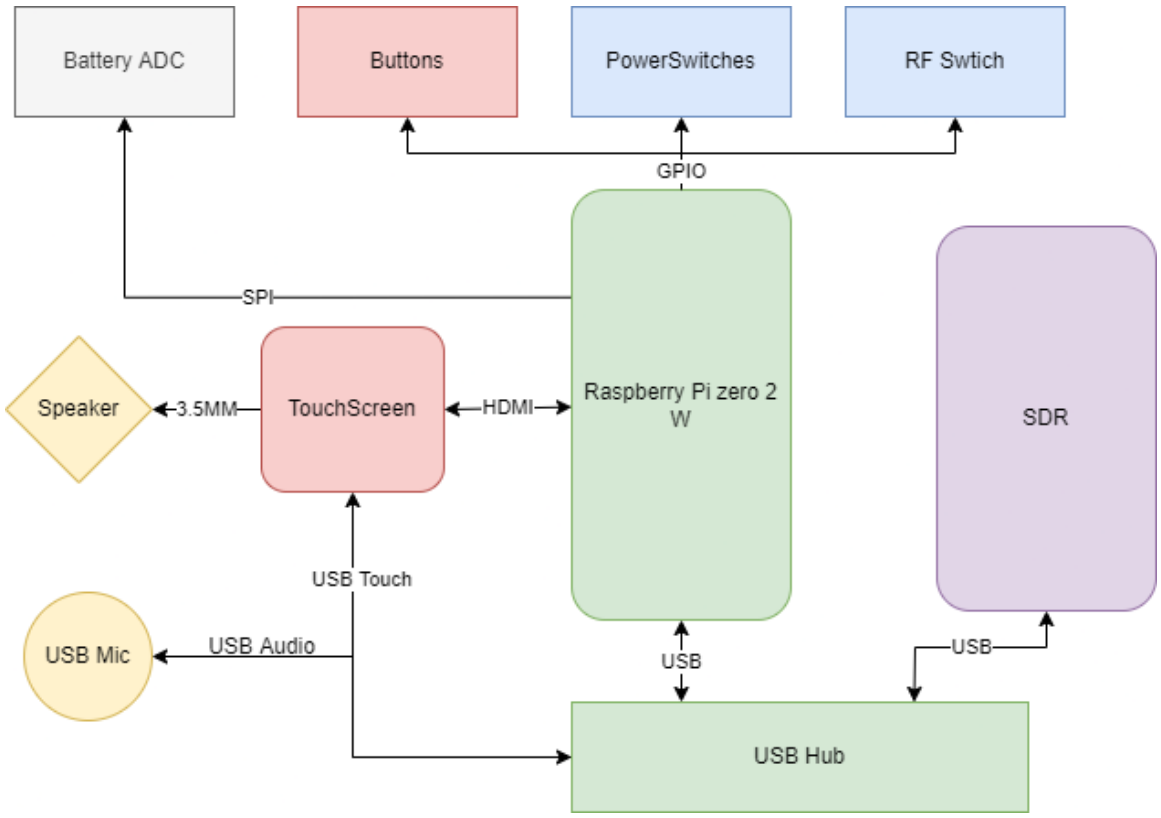


Figure 51. Digital System Block Diagram

Power Subsystem

The Power Subsystem Provides power to every system on the Radio. The voltages and current were determined by the part requirements. The main batteries will be the Panasonic 18650B NCR as discussed in the part selection section. The battery voltages will be regulated to a single stable 5V output and another variable output with a voltage range of 5V-7.2V.

The power going into the amplifiers is controlled by two dual mosfet modules connected to the raspberry pi. This allows for output power control and selection as explained in the RF subsystem section. The mosfets will be used as a switch allowing the gpio on the Pi to turn on or off voltages on different parts of the rf subsystem. The on resistance of the mosfet at our current will provide a low power dissipation and therefore a low power drop of less than .01 volts.

The battery voltage will be monitored by an SPI adc connected to the pi. This will allow for a low complexity solution to battery monitoring and charge control. Another step to avoid unnecessary complexity is the charging setup. The batteries will be made hot-swappable in design so they can be charged outside of the actual system, and the case in which they are contained will be directly able to be charged with a USB-C cable for simplicity and accessibility.

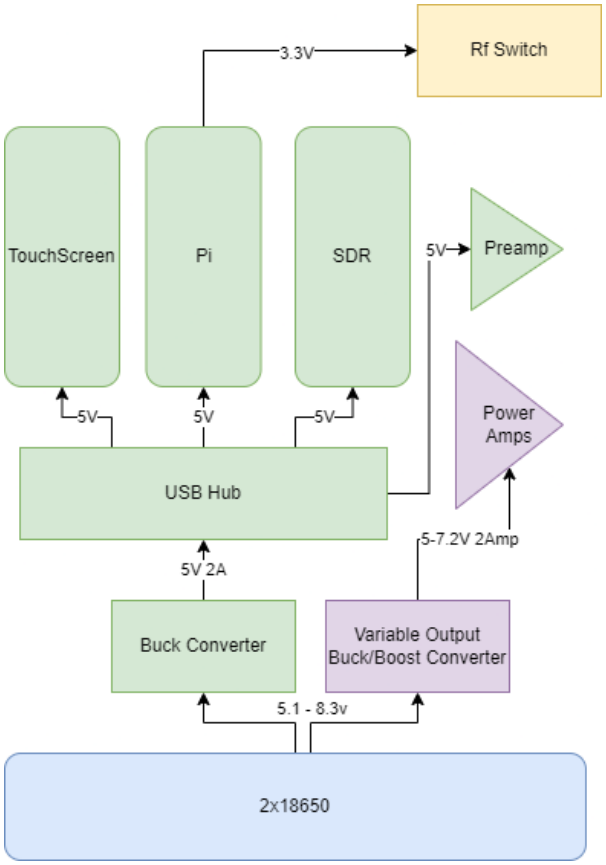


Figure 52. Power System Block Diagram

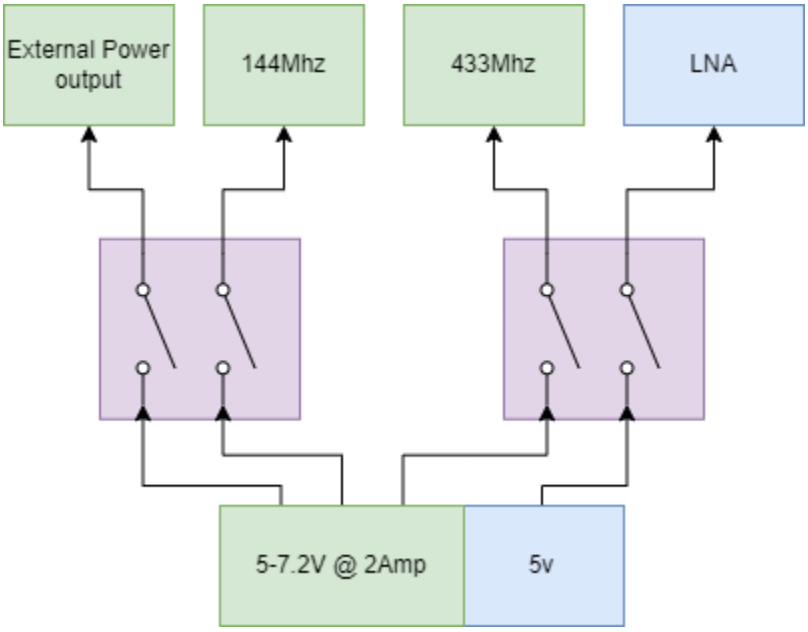


Figure 53. Power Switching Block Diagram

Full System Overview

The Entire system combine will work like described in the description to meet the requirement specifications. The Pi will be the main controller for the entire system controlling all other subsystems. The Pi will connect to the screen and run software for a GUI which will allow for the SDR power control mosfets and RF switches to be controlled over the touch screen.

This design obviously has one major risk or potential flaw. The pi may not be able to provide enough processing power for graphics, power, and RF control. Since this is a possibly the main design choice to combat that will be switching controls / processor heavy operations to the FPGA on the SDR. The first components to go would be those which are determined to affect the overall user experience the most. So first and foremost the volume squelch and PTT controls would be switched over to the SDR with no necessary system architecture change.

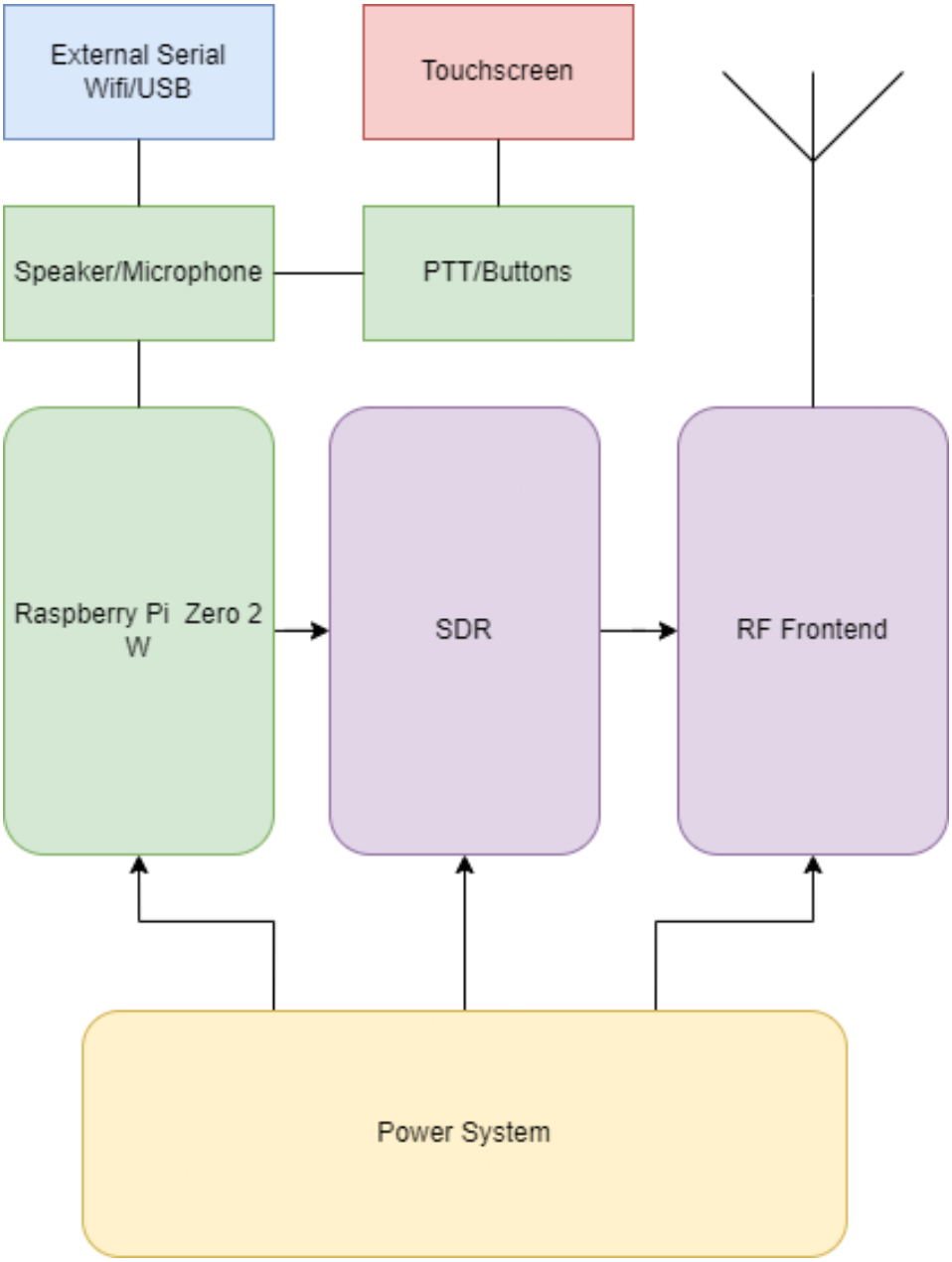


Figure 54. General System Block Diagram

Physical Design

The physical design must be handheld and durable. We also need to dissipate a large amount of heat from the Pi, SDR, and Amplifiers. The amplifiers come with preinstalled heat syncs. The plan for the pi and the SDR is to thermally couple them to a metal plate. But because transmission are generally shorter than 30 seconds and the unused components will be in a low power mode active fan or thermoelectric cooling was decided against

A good comparison of the size of everything compared to common of the shelf hand held radios:

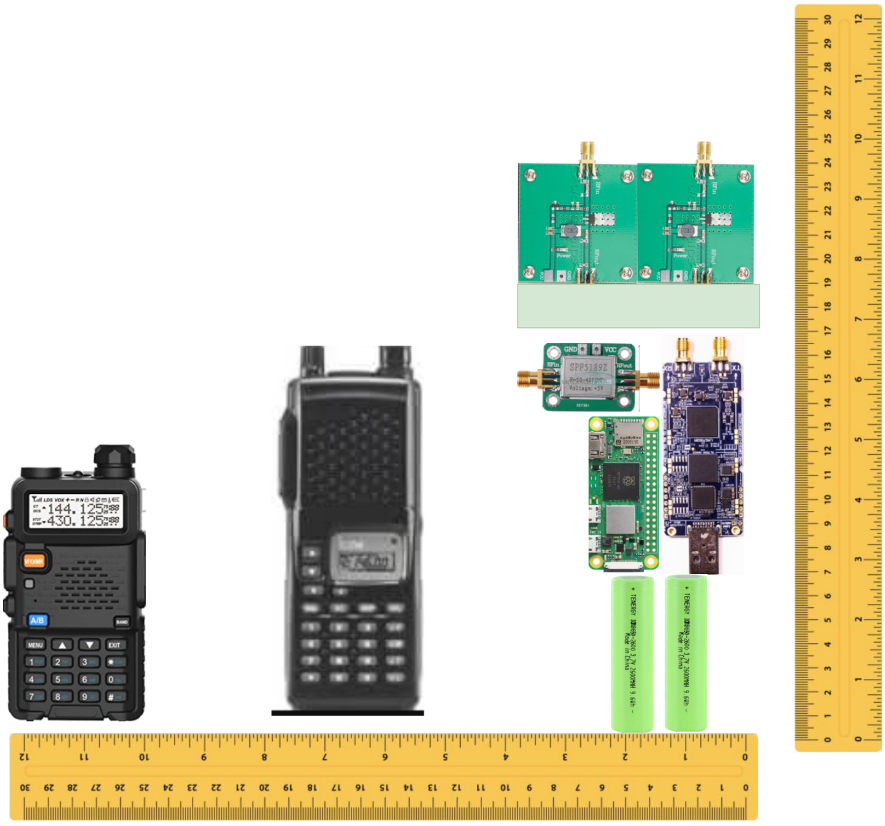


Figure 55. Hand held radios

Since the largest components are going to be the screen at 4in in width and the width must be able to accommodate that. The size of the amplifiers will also be a challenge possibly requiring us to physically cut the amplifiers thinner. The outer case will be plastic and 3d printed except for the metal parts for heat sync.

The case will also function as a battery holder designed to allow easy battery hot swapping or charging. This decision is discussed in the part selection and power subsystem sections further. The batteries and casing will be removable from the radio itself and easily charged via a standardized USB-C cable that can be plugged into any power source (via wall adapter if necessary) and left alone. The casing itself features overcharging protection so the user does not have to actively check the batteries once they begin the charging process. The batteries themselves also feature a protective IC so that they can guarantee their safety while being recharged.

Each amplifier will be shielded so as to not pick up noise from clocks or switches on other modules. This is important to meet FCC regulations as unwanted noise can be easily picked up and amplified especially by the wide band LNA. This is luckily easily tested with an oscilloscope and is a priority in the testing section.

3D Printing

For our device enclosure, we are planning to use 3D printing to create a case that will hold all of our components as well as the lithium-ion battery cells. As mentioned in the case materials section, we will be experimenting with different 3D printing filaments that are only compatible with the standard extruder style printers. Of this style of 3D printer, the two forms are either a standard XYZ CNC style or a Delta style. The latter of the options are best suited for projects that have high vertical height requirements and have to print with a very high level of accuracy. The former is a more common style of printer and is able to handle the case we model. Alternatively, we have the option of using Resin 3D printing which utilizes various compounds that react to a UV light which hardens the material and forms the final product. This style of printing is very cost effective and is not applicable in the current design of the project, but when the Amateur Radio Club is satisfied with the end product they have the ability to use the model we create to make a more robust case from this procedure.

We will likely need to follow certain steps to ensure that our final print comes out in acceptable quality. First and foremost, the case has to be designed in a minimum of 2 separate parts. One part is the actual case itself which will hold all the components of the software defined radio, and the other is the back cover for the battery cell bank. If in the event we choose to use only 2 separate parts in our case design, the printing process will have to be paused at the halfway mark in order to mount all of the main components in the casing and let the casing resume printing around everything. This proves difficult as the print has to remain in the exact same position as when the pause first occurred or else the filament will be printed off centered and the casing will be ruined.

The best course of action, then, is to print in 3 separate pieces. One for the battery cell cover, another for the lower half of the casing where we are able to mount all of our components, and the last to enclose the device. This is the easiest route we can take when it comes to finalizing the design for our radio case. Since this is likely going to be our final design after case testing, we will need to find a good solution for waterproofing as this design will need to be joined together with some sort of fastening mechanism and will not provide a watertight seal.

Before the print starts there are several factors we can take into account to ensure we have the best quality case we can produce. First is the speed of the print itself. 3D printing is generally a very time consuming task and for the size of print we are aiming for, the duration could easily reach over a day's worth of time. To alleviate this we can speed up the print at the cost of the end quality and durability. Aside from speed we are able to alter the printing density. Instead of extruding more filament out of the print head extruder, .gcode rendering software lets us choose what internal support structure is going to be implemented; for example, we can choose from a solid packed print, a honeycomb structure that has spaces in between, or a crossed pattern also with empty space. Finally, we are able to change the heat at which the filament is extruded at and also the temperature that the print bed itself cycles at. Filament heat changes how the filament will bead and join to another strip of filament

whereas bed temperature will have an affect on factors such as stability during printing, warping, and finished print removal.

Power Supply Overall Schematic

Our power supply schematic has multiple distributions of power that are necessary to power the entire system. Since the Panasonic batteries are connected in parallel, the voltage coming into the system is 7.4V which is the voltage of the two batteries added together. The power necessities of our board include multiple 5V outputs and an output range of 5V-7.2V with all values in between this range included, as the purpose of this is to be able to control the watt output of our radio and vary the frequency bands we are operating on by extension. For this reason, we need to design a PCB capable of receiving 7.4V battery power and utilizing buck-boost converters to split this power amongst two separate outputs that our other components can attach to as needed.

Initially, we require a consistent 5V output from our power system that will never fluctuate in order to power multiple components such as the Raspberry Pi, touchscreen and LimeSDR. As the power requirement for this section never changes, this becomes a much simpler overall goal to reach than the power amplifiers will be, and we chose to use a standardized buck converter to shrink our input voltage from the 7.4V nominal provided to our 5V output. The power from the batteries will slowly drain as the batteries are used, or by extension be higher than the nominal when batteries are fully charged, so our buck converter must be able to sustain both the upper and lower boundaries of our voltage range from the two Panasonic 18650B batteries. Using the data sheet, we can see that this range is from 8.4V to 6V, and means any buck converter that can output at 5V while having boundaries that contain this range will work. For this project, we selected the “TLS4125DOEPV50XUMA1” buck converter, an optireg switcher produced by Infineon technologies. It functions primarily as a 2.5A step down regulator, and can accept any input voltage from the range of 3.7V to 35V while outputting at a fixed 5V output voltage. Other benefits of this buck converter include its extremely low current consumption (rated at 31 nanoamps) with high switching frequency ranges/oscillation frequency. If chosen to be used, it also has spread spectrum frequency modulation for improved EMI performance and a wide temperature range, proving its full sustainability within our project and featuring our project well. The schematic for the 5V fixed output regulator is shown below.[26]

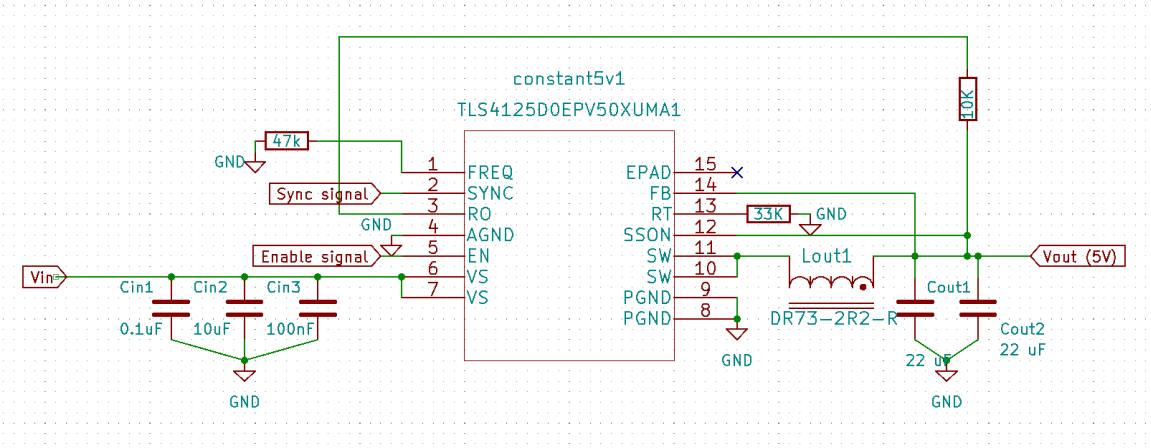


Figure 56. 5V Buck Schematic

Within this schematic, there are many design choices that are made for the sustainability of the overall project and performance of this regulator. As if this regulator fails our entire system will turn off, we felt it wise to add an extra capacitor to the input voltage in order to improve EMI performance and to also add an extra capacitor in series to our output voltage to smooth the ripple as much as possible. The signal “RT” with its attached pull up resistance is connected to a reset functionality in line with the “RO” which is kept as an under or overvoltage protection as well as overcurrent protection within the system so the selected IC will automatically protect itself in case of an unprecedented situation rather than frying the component itself and needing a replacement. The value of the resistors refer to the setting of the reset values themselves, such as the 33K ohm resistor attached to RT being set so that the expected input voltage of the system being 7.7V and resetting if the voltage is to ever drop below 5.1V. As the batteries cannot sustain themselves if they are totally discharged to this point anyway, it is a safety measure that will protect us from anomalies. While there are many buck converters that would have worked for this purpose and possibly some with higher efficiency (the above selection is rated at ~95% when in rated conditions), we struggled to find a buck converter with this specific voltage range that was available and in stock. Using the digikey search marketplace only returned less than 100 immediately available ICs, and thus eliminated much of its competition (such as the TPS TI IC series were all primarily sold out for those that exist within our voltage boundaries). This IC is also very cheap in comparison to its competitors which can often be more than five dollars for the single component alone, while this IC is currently priced at $3.35 per IC.

Now that we have a stable and optimized 5V output regulator to be supplied to the USB hub and provide power to our major components, we must now stabilize the power for the power amplifiers. This poses a unique challenge as these power amplifiers must be capable of being given anywhere from the 5V minimum to 7.2V maximum at any given time in a controllable manner so the user can both select the power output of the system as well as the frequency the

radio is outputting on. To accomplish this, we require a variable output boost-buck converter capable both step-down and step-up dc-dc switching regulation for battery power, as after the batteries drain from their nominal voltage it is possible that we will require more power than we are actually getting from the batteries (for example, the system can be changed to the 7.2V output while the batteries are outputting 6.5V). To accomplish this task we chose the IC TPS630702, which possesses an input voltage range from 2V to 16V and an adjustable output voltage of 2.5V to 9V, which fits within our ranges perfectly. The IC was chosen due to its high efficiency rating (up to 90% when producing an output current higher than an amp) and high switching frequency, peaking at 2 MHz oscillation which is very important to the general design of our board due to the response to changing the voltage needing to be completed as fast as possible. This TPS series IC has a counterpart known as the TPS630701 which features a static fixed output, so it is important to distinguish specifically that this is the TPS630702 and is being used in its adjustable buck-boost version.

However, the TPS630702 itself is not enough to change throughout the entire range of the voltage, so we must add a potentiometer in series with a second resistor that we can use to adjust the voltage at will. The added resistance will decrease our output voltage as the resistance is increased, as without the potentiometer the current schematic outputs our maximum voltage of 7.2V. Due to the fact that the rest of the circuit is held by a schottky diode, we are not concerned with the possibility of an overcurrent and added a capacitor after the addition of the potentiometer to smooth voltage ripple for a stable output. As this IC features a gated oscillator control scheme, it provides a high efficiency over a wide load range and only seems to fall in efficiency when Vout drops below 5V. The IC uses a hysteresis window to regulate the output voltage with this system and switches continuously with a fixed duty cycle when below the upper threshold of the load range. In each switching cycle, the internal N-channel MOSFET switch is turned on and causes both inductors to begin storing energy, and this highly efficient process will allow us to eliminate output voltage droop as well as provide overcurrent protection within the circuit to protect the IC in a similar manner to the 5V regulator. The schematic for this IC is shown below. (Infineon) [51][52]

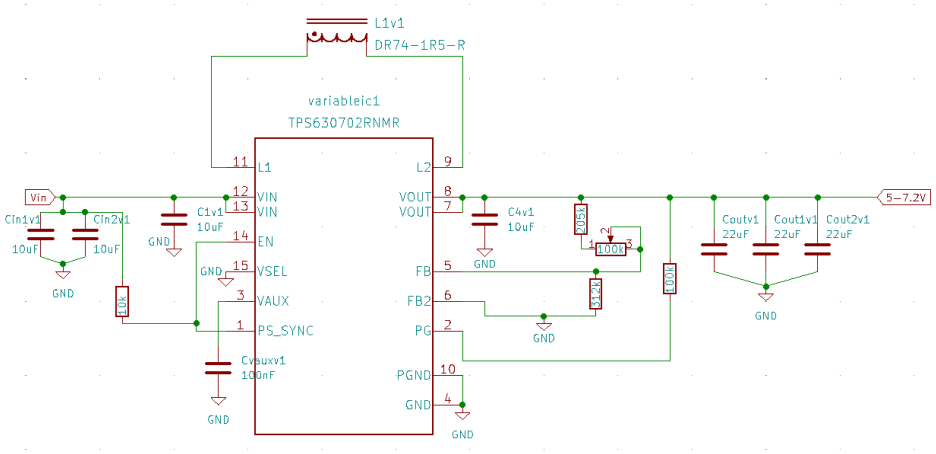


Figure 57. Variable Output Boost-Buck Converter

The remaining major component that will require power is the RF Switch. This will feature three RF channel utilization and provide a connection for the antenna, but only requires 3.3V to be powered. Instead of making a new regulator and taking battery directly from the batteries, the Raspberry Pi Zero 2W has multiple pins with 3.3V capability which can directly connect to the pins labeled as “V1”,”V2”, and “V3” in the following schematic. The RF Switch’s Vdd can be set to 5V conveniently, which means we can tie that pin directly to the output of our original regulator to stabilize our system and show that the RF switch will receive all proper power as shown below.

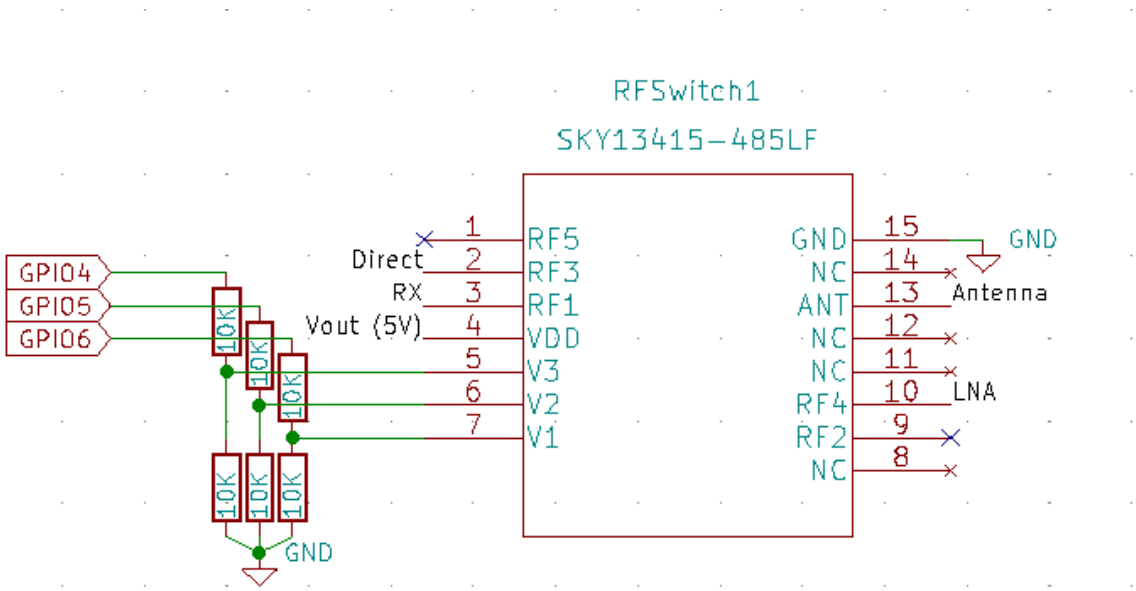


Figure 58. RF Switch Schematic

Knowing that one of our design constraints consist of the overall power of the system being able to run for a long period of time without needing to be charged, we felt a good design constraint would be the ability to monitor the power level remaining in the system similar to a phone battery-level indicator. To accomplish this, we decided to utilize a 10 pin ADC module that has the capabilities to communicate with the serial communication protocol known as SPI. We can connect this directly to the Pi as the Pi’s SPI capabilities are not currently being used and would be able to directly report the amount of voltage that is coming from the batteries. To power this ADC, it requires a constant 5V output which we can use the earlier created 5V buck regulator to supply. We then add the battery connection directly to channel 1 with a voltage divider to protect the component which can then be communicated as a value over SPI to the Pi, which we can then display on the screen. The current draw and power draw from this is minimal and the benefit of being able to monitor our power at any given moment far outweighs the negative of the slightly higher current requirements, as the watt hours of our batteries far exceeds our necessities.

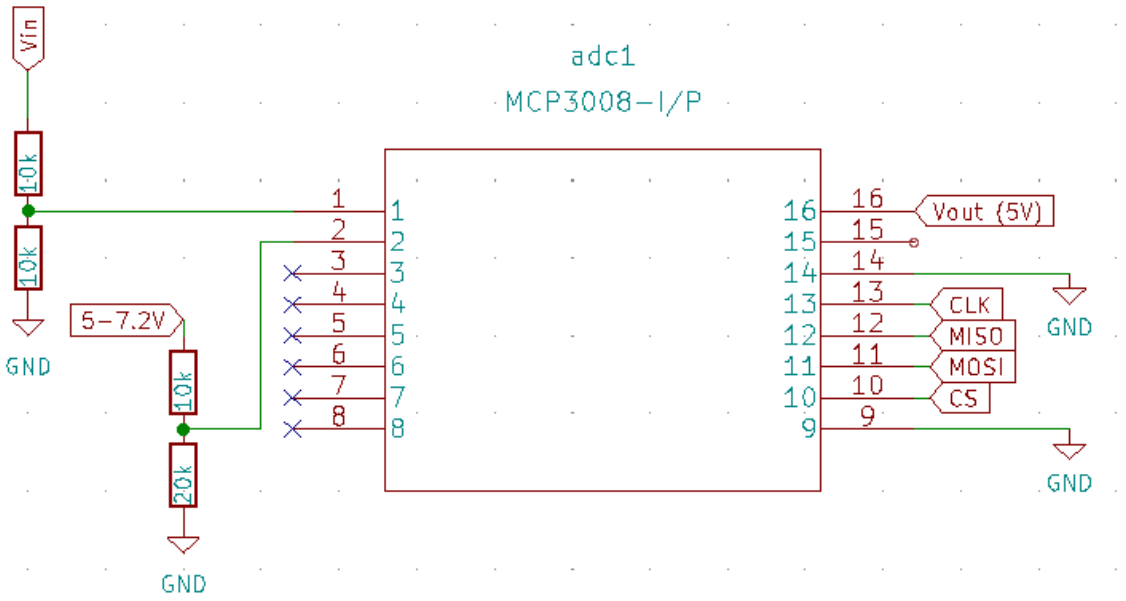


Figure 59. 10-Pin ADC schematic

The Power Mosfet circuit has multiple GPIO connections allowing the Power going in each of the amps to be completely switched off. We opted for this option rather than multiple high power RF switches because generally the higher the power the RF switch the more noise and insertion loss. The mosfets will have a low on resistance causing a slight voltage drop of less than .01 volts. The gates of the mosfets have a “on voltage lower than the voltage of the pi's gpio so direct driven GPIO will provide adequate IV characteristics to allo the mosfets to act as a relatively ideal switch. Because the turn on voltage of the mosfets is so low the gates need to be adequately pulled down and although the pi can do this but since the plan is to have low power mode this may not be enough. Because of this at the very least resistor footprints will be added to the PSB and unpopulated in case the low power modes won't be able to pull down GPIO. from here the output of the fets are connected directly to pin headers to allow for the

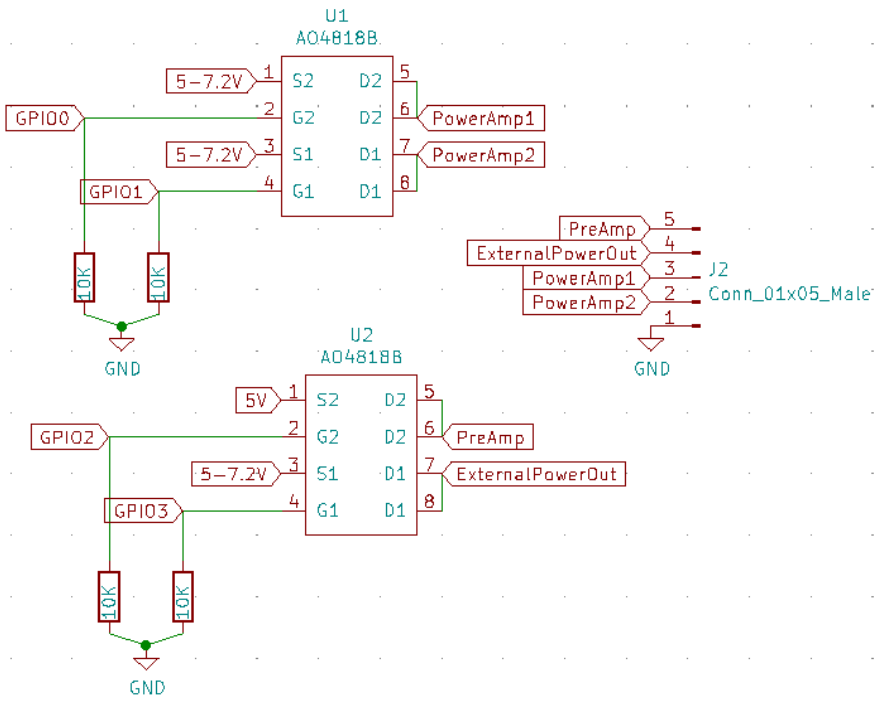


Figure 60. Power Mosfet Schematic

All in all, the full system schematic showing the connections across the entire radio is shown below. The overall schematic is designed to work together with each component designed with its effect on the overall system. The regulators are designed for low EMI in the relevant bands; the mosfets are over specification to allow for lower heat generation that would affect the nearby RF switches.

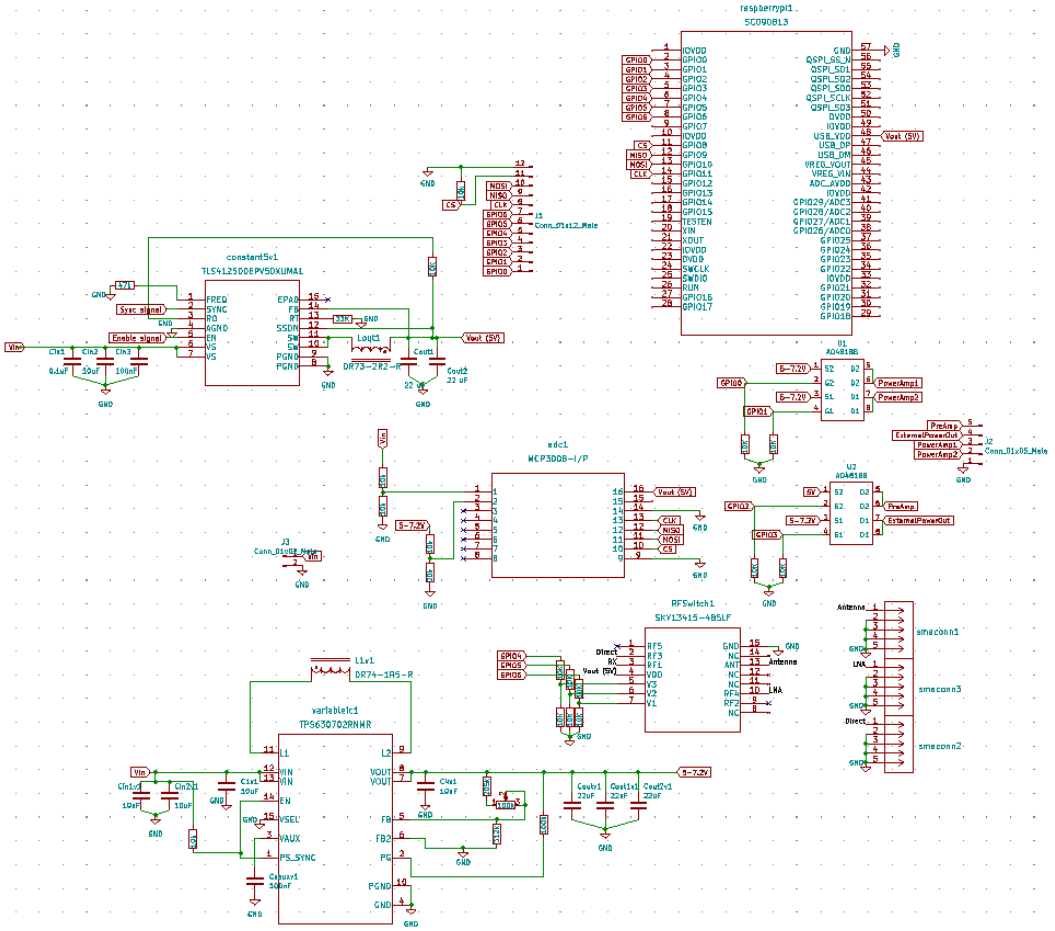


Figure 61. Overall Full System Schematic

4.2 Software Design (10-20 pages)

4.2.1 Avoiding Anti-Patterns

One of the requirements for our project is that all of our code will be open-source. The intent behind this is that once completed, our project would be iterated on by others. While for some other projects writing code that just barely works could be considered acceptable, in our case part of the development process is leaving behind code which we could be proud of and that other students would be able to iterate on without a lot of trouble. Because of this, we will attempt to consider various anti-patterns during development. [49]

An anti-pattern is a common design problem that usually comes up in software development due to developers who are any combination of lazy, incompetent, mismanaged, misinformed, or rushed. Because this code will be passed on to others, we will try our best to avoid these patterns as much as possible. Here are some of the common patterns that we might run into, how likely we are to run into them, and how we specifically will attempt to avoid them.

Spaghetti Code

This generally refers to unorganized code, and comes up often whenever development focuses on adding one feature at a time instead of taking a big picture view of how the overall device's features will be organized. One common way to avoid this problem is to have frequent code reviews where somebody else takes a look at how you are implementing a solution. While this normally helps, this might not be enough of a fix for our small group as at most only 3 other people would be reviewing code, and not all of us are programmers.

To help avoid this issue further, once the software development process starts we will be making diagrams that describe the flow of code so that we can keep track of how things are organized. The idea behind diagrams is that if your code is too complicated to make a drawing of, then the code itself is too complicated and you have failed to keep it organized.

Golden Hammer

This anti-pattern refers to how once developers find one tool that works, they will try to apply it to everything even if it might not necessarily be the best tool for the job. Unfortunately this anti-pattern is one we may have to deal with allowing, as there are a lot of different moving parts in our project. If we looked for the best possible solution to each different part then we would not finish on time when Senior Design 2 began. A good way of dealing with the impact that this anti-pattern might have on the code is proper modularization and separation of responsibilities between different sections of the code. What this would accomplish is that even if a tool is being used improperly, it would be easy to swap out for something that works better for that specific task.

Specifically in our project, the use of an OS, Python, C++, and LimeSuite may end up being excessive. Due to this, we will plan for allowing any of these to be swapped out into something else or absorbed into something else.

Boat Anchor

This anti-pattern simply refers to putting in code that solves a problem that we will have in the future. This should be easy to avoid as we ourselves do not know in which direction the project will be taken after we have finished it, so we would not be able to account for future features during development.

Dead Code

This refers to code which has been left in the code base which does nothing. This anti-pattern commonly shows up in matured codebases which have been under development for years. As our code will only be worked on for a few months, this is extremely unlikely to happen. To minimize it we will reconsider the inclusion of code whenever a system that it touches is modified.

God Object

This refers to having a single object with too many responsibilities. In particular we are at risk of falling into this anti-pattern with our Python code, which will be in charge of speaking to the OS, GUI, and LimeSDR. During our development cycle we will pay particular attention to making sure that if any one Python file is getting too large that it be split up into separate sections.

Cargo Cult

This pattern refers to implementation of a solution which is not the right fit for our problem. It commonly happens when a developer uses an outside solution to solve their problem, without fully understanding the outside solution.

The problem that we are working on is new and unique, so instances of using others open-source code solutions will be sparse. To minimize these instances even further, our code reviews will involve sharing whenever a section of the code requires outside assistance to implement. By knowing this, our team can work together to check that section of the code to see if it works properly for our system.

Magic Numbers and Strings

This refers to the lack of symbolic constants when defining specific numbers or strings with significance to the business logic and flow of the software. For example, if the codebase has an if condition such as “if (numberOfWindows > 3)”, this would be poor. A way to improve this would be to have a constant such as “#define maxWindows 3” and then have our if statement read as “if (numberOfWindows > maxWindows)”. The reason that this is better is that without proper naming, somebody would not be able to understand why the number 3 is significant for this if statement. While this could be explained in a comment above the code, self-documenting code is often better, so if an if statement requires a comment to explain a magic number then that just should not be a magic number in the first place.

To avoid this problem, we will be checking for magic numbers and magic strings in our code review sessions whenever somebody is adding new code to the existing codebase.

Interface Bloat

This problem often comes when designing the UI of a system. This refers to making a poor interface due to lack of prioritization when it comes to deciding which features are useful and which are not. We will easily avoid this due in part to our time constraints, as with how little time we have to develop this project there will not be a chance to add so many features that our interface gets bloated.

Poltergeist

This problem refers to the use of a layer of abstraction that does nothing but call some other part of the code to do something. This is similar to the Boat Anchor problem mentioned previously, as it may occur whenever somebody is attempting to future-proof their code by adding layers of abstraction which are not currently being used, but that they think may end up being used later. To avoid this problem, we will be trimming our code down at the end of software development to get rid of any code that is not doing anything meaningful.

Shotgun Surgery

This problem is more likely to occur on a matured database. It refers to when implementation of a new feature requires precise changes across multiple areas of the code. This is extremely likely to happen to our codebase when it gets further developed by others, but it is unlikely to occur during initial development of the system. No particular efforts will be made to avoid this anti-pattern.

Copy and Paste Programming

This specifically refers to the issue of copying and pasting code from our own codebase from one section to the other. For the issue of copying and pasting code from the internet, see Cargo Cult Programming.

The reason that this antipattern is bad is that what will usually end up happening is that a developer would write code in one section, and then need to use that code somewhere else. The problem with copying and pasting the code from one section to another is that if a bug is found in that code, then that code needs to be changed in every single section that it was copied and pasted. If the code was modified slightly for each of these sections, it may be very difficult to track down each instance of the code.

To avoid this issue, we will be making use of proper programming practices such as turning code into methods whenever possible. This would allow us to reuse code as much as possible without having to change code multiple times.

Error Hiding

This simply refers to when the code is catching an error but we do not report it properly to the user. This is likely to happen to our code as we will likely not have proper error catching methods in place for each possible error. Unfortunately this error is one that we likely will not be addressing during development, as error catching can be a sinkhole in terms of development time. While this normally would be prioritized during normal development to avoid issues related to users being frustrated with the end product, we do not have the time available to dedicate to this part of software development.

Manual Memory Management

This refers to attempting to manually handle each byte of memory as opposed to allowing the language's built-in garbage collector to from doing its job. This antipattern is very easy to avoid in Python as code written in Python does not usually involve memory management.

4.2.2 Version Control

Our group will be implementing proper version control practices in order to streamline the software development process as much as possible. While version control practices might be skipped in a very small group of developers, with all 4 of us working together on software simultaneously it would be best if we stayed organized. Additionally, one part of the requirements for this project was that we were required to maintain open-source code so that the project could be continued by future students. Thus because of both the need for this group to remain organized and the requirements that our code be open-source, we have decided to host our code on a public GitHub repository. Finally, we will be handing over admin privileges over the GitHub repository to the standing officers of the Amateur Radio Club.

The process for implementing code will be as such:

1. Create an item with acceptance requirements
2. Create an appropriately named software branch based off of the dev branch
3. Complete the item while maintaining properly scoped commits
4. Begin a Pull Request to bring the code into dev
5. Address all comments brought up until a majority of peers approve of the code changes
6. Bring the code into dev

To create an item with acceptance requirements, we must hold planning meetings where we will decide what the members of the team who are working on software will be developing next. These items should usually be specific, such as "Add navigation buttons to the home page". Non-specific items such as "Add a homepage" should be broken up into smaller chunks until it is easy to define what the minimum work that has to be done to have that item be considered 'complete' is.

When beginning work on an item, it is bad to work directly on the dev branch because this might affect what somebody else is working on, and you may end up leaving the software in a broken state if you stop working on your item halfway through. Because of this, it is best to make a new branch, based off of dev, for you to make all of your changes in. Usually these branches should be named either feature/~~~, refactor/~~~, or bugfix/~~~. These correspond to branches that introduce, remove, or change a feature of the code, a branch that changes the code without actually changing what the user sees, or something that fixes a bug.

While working on the item, it is usually helpful to separate the changes into commits based on what was done. This can make it easier to parse all of the

changes commit by commit during the Pull Request, and can make it easier for the developer to undo a batch of changes at once if they know exactly which group of changes was incorrect based on the commit messages.

A Pull Request is a feature on GitHub that allows developers to show their code to other developers on their team so that they can approve it before the code is combined with the main codebase. Making sure that code only enters the main code base through pull requests is a good practice because it ensures that multiple eyes have seen and verified every single line of code which has been integrated. This minimizes bugs, inefficiencies, and other potential problems. The process of how to begin a Pull Request changes whenever GitHub updates, so it is best to look up how to do this.

While reviewing somebody else's Pull Request, the standard procedure which we will be following is that if you see a problem in their code, you should put a comment about it and refuse to approve the Pull Request until your comment has been addressed either with a reply or with a change in the code. The person who created the Pull Request should be taking everyone's feedback into account. Once nobody has any more feedback to give and the code changes have been deemed complete, it is safe to bring the code into the development branch of the GitHub Repository.

Desktop Environment

PiXeL

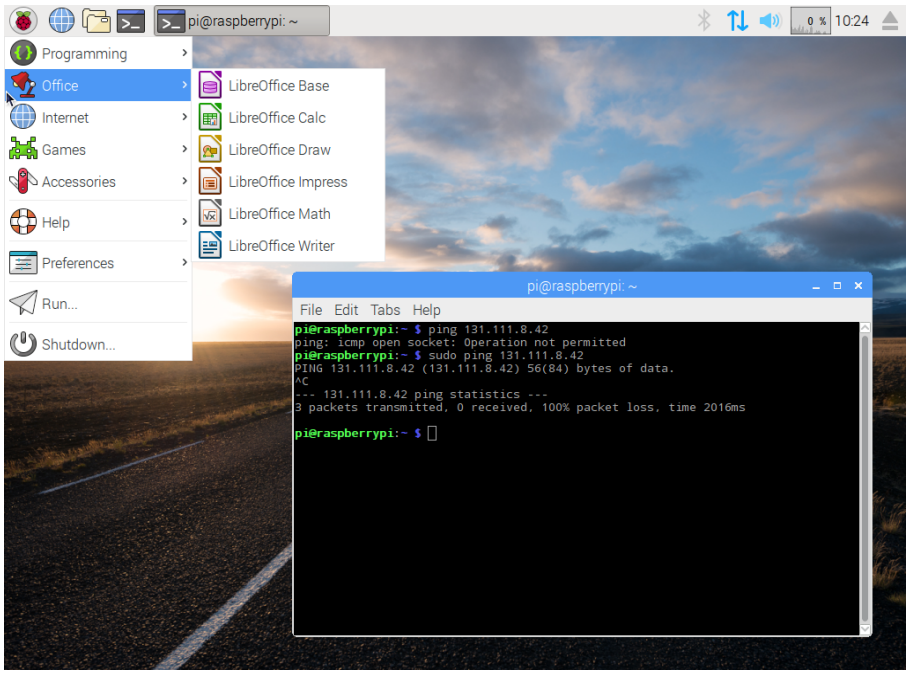


Figure 62. PiXel Desktop Enviroment

The Raspberry Pi OS comes pre-installed with a modified version of the LXDE (Lightweight X11 Desktop Environment), called the PiXeL (Pi Improved Xwindow Environment, Lightweight). This desktop environment has been

designed specifically with the Raspberry Pi Suite in mind. Its benefits include the fact that the install process is simple, it has an easy to use menu, and it has very good performance due to both being a lightweight desktop environment and having been designed specifically with the Raspberry Pi in mind. Its negatives include an overall unappealing interface, and difficult to access settings due to different settings being modifiable through different menus. These are relatively minor downsides as they would only affect the development process, as an end user would not be directly accessing the desktop once development is done.

KDE Plasma

Figure 63. KDE Plasma Desktop Environment

KDE (K Desktop Environment) Plasma is a desktop environment for Linux systems which borrows heavily in its design from OS' such as Windows XP. Its benefits include simple to access settings, and an overall well-designed interface. Its main negative is that the performance of the desktop environment is poor. This is a huge negative because on the Raspberry Pi Zero 2 W there is only ½ GB (GigaBytes) of RAM (Random Access Memory).

GNOME

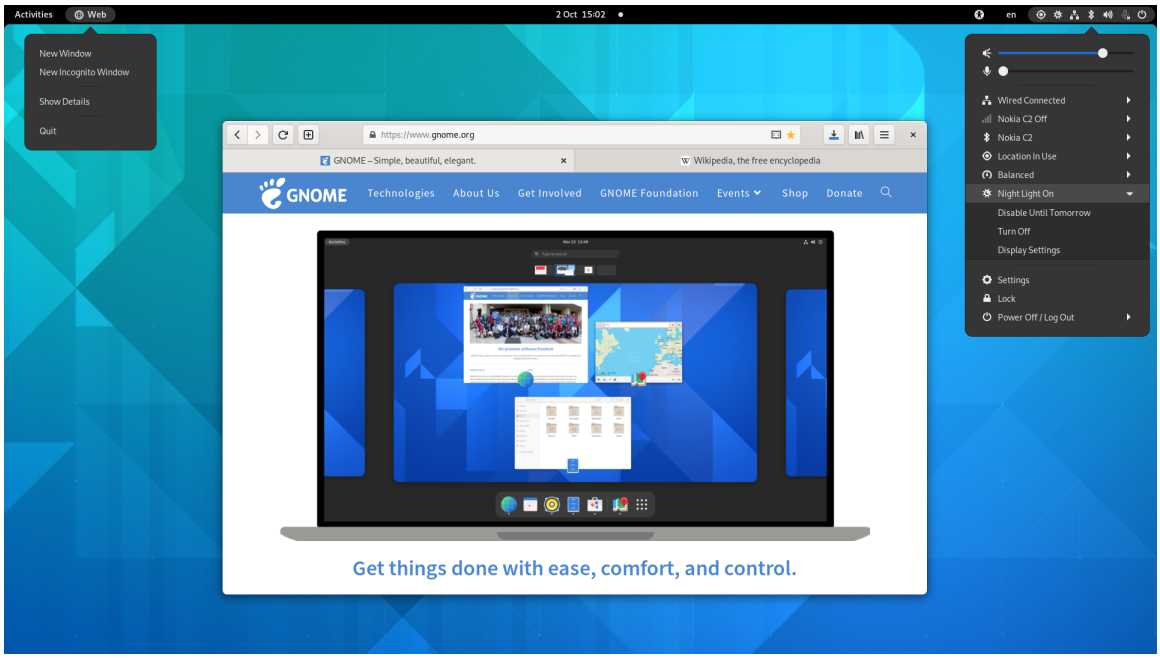


Figure 64. Gnome Desktop Environment

GNOME (GNU Network Object Model Environment) is the most popular desktop environment for Linux devices. It has easy to access settings, similar to KDE Plasma, and is well-designed overall. Unfortunately, it is hard to learn for beginners, and similar to KDE Plasma it requires more processing power than a lightweight desktop would require. Because of that, this desktop environment is not suitable for our project.

LimeSuite

LimeSuite is a collection of software drivers meant to support many different hardware platforms, including the LimeSDR, LimeSDRMini, and LimeSDRMini 2.0. Installing LimeSuite is necessary to allow SDR applications to work, and installing this is the first step when attempting to use the LimeSDR, even before plugging it in.

The installation process for LimeSuite is different depending on which OS it is being installed on. As we will be using a Linux-based OS, we will have to follow the Linux installation instructions. The easiest way to install LimeSuite would be to install it via PPA (Personal Package Archive), which is just inputting a few commands into the terminal and letting our package manager handle the rest. The other way to install LimeSuite would be to download the source code and build LimeSuite ourselves, which is a more involved process. Either way we would be getting the latest version of LimeSuite from the github repository, so it may be worth noting the exact version of it that we are installing in case future versions break dependencies.

Firmware Management

While LimeSuite described the suite of software we install on whatever system we are using to communicate with the LimeSDR, it is still necessary to update the software on the LimeSDR itself. This involved updating both the firmware, and the gateware.

Updating the firmware refers to updating the software on the microcontroller on the LimeSDR. Updating the gateware refers to updating the CLBs (configurable logic blocks) on the FPGA (field-programmable gate array) on the LimeSDR.

The act of updating both the firmware and gateware is handled by LimeSuite, as each installation of LimeSuite is packaged with the latest firmware and gateware which will work with it, and only a single command is needed to begin the update process.

LimeSDR Test

Before starting on development, it is important to check that the hardware and installed packages are working. To do this, we follow the instructions at https://wiki.myriadrf.org/LimeSDR-USB_Quick_Test . By following these instructions, we can verify that the LimeSDR is able to communicate with its host device, transmit radio signals, and receive radio signals. The loopback test is enough to verify that the hardware works, while the other 2 tests are meant to show a beginner how to use LimeSuite.

Flow Chart

The figure below describes the flow of information on the software end of things. At the heart of our software ecosystem will be our operating system, and python.

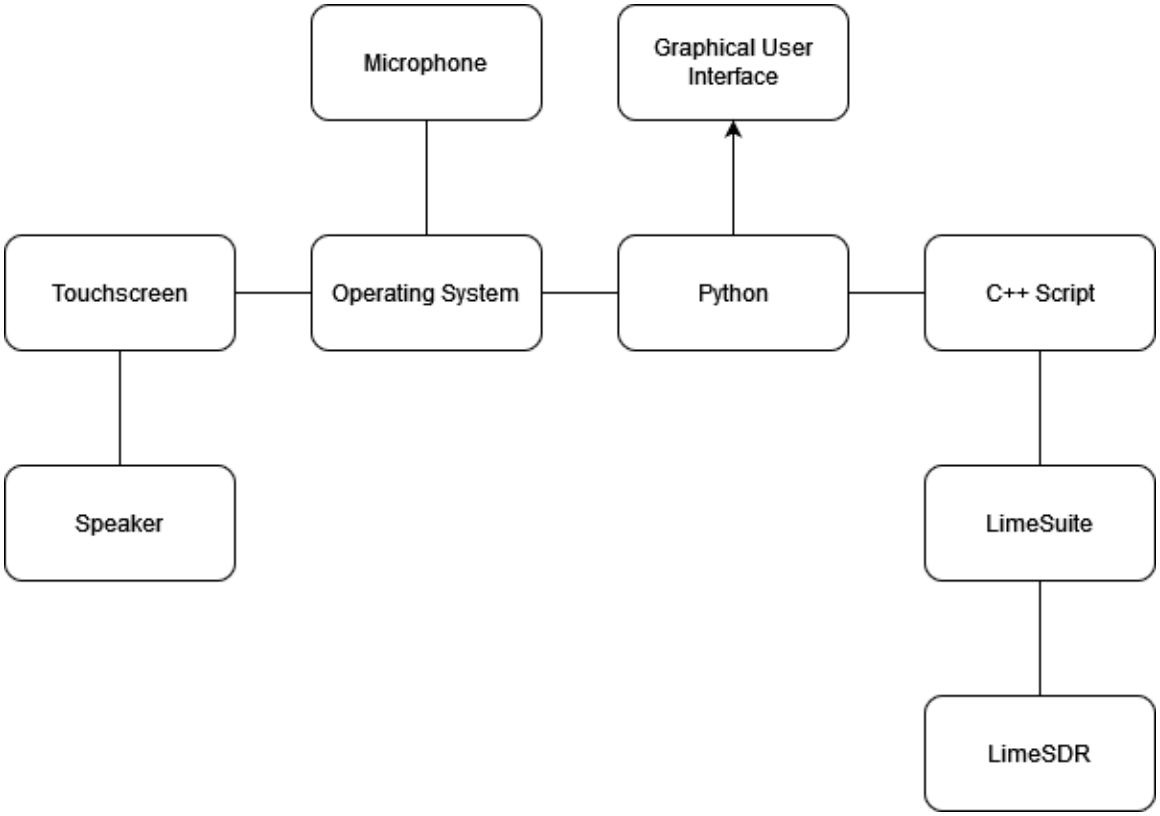


Figure 65. Software Flow Chart

The operating system will handle a lot of the background tasks which we are not prepared to code from the ground up. This can include responsibilities such as resource allocation, communicating with a microphone for audio input, communicating with a speaker for audio output, task scheduling, etc. This might include something as complex as knowing the correct routing algorithms to be able to send audio out through an hdmi port which is connected to a touchscreen with a 3.5mm audio port.

Our python code will handle creation of the Graphical User Interface (GUI), as well as communication between the different software modules that we will be implementing. For example, our python code would be in charge of things such as making sure that when somebody presses a button on the touch screen, such as the home button, that the GUI responds appropriately, such as by changing screens to the touch screen.

Communication of the LimeSDR is difficult and would take multiple layers of abstraction to do. There is a public tool called LimeSuite which helps speak with the LimeSDR, but it is based on C. Thus to be able to speak with the LimeSDR, we need to be able to make calls to LimeSuite inside of code. This is done easily in C, so we will have to have our Python code speak to a C++ script, which will then speak with LimeSuite, which will then handle communication with the LimeSDR.

4.2.3 User Interface

Related Works

There are many existing software defined radio programs available for amateur radio enthusiasts to utilize to meet their needs. Most of the programs themselves are free to install and use, come with documentation and tutorials, and are user friendly for those who have the basic knowledge and skill set required. Since our software defined radio will be based on a limited resource embedded system with a very small screen, we will not be able to include a full set of features as seen in many software defined radio programs. Nevertheless, we will be able to replicate some of the most common functionality of popular programs in our software defined radio.

HDSDR

One of the most popular software defined radios available today is the HDSDR. Its primary uses include listening to regular radio stations, ham radio applications, astronomy, and spectrum analysis. It offers many features that match the requirements of our own software design radio program such as a waterfall display, I/Q modulation for signals, recording and playback, as well as options to select standard bandwidths.

Currently the HDSDR is only available on the Windows platform, but there is development for Linux and Mac based systems, an area where our system should excel since most of its software components will be cross platform. Another downside to the HDSDR is that it is for use on recommended receivers, all of which offer a much lower bandwidth than the LIMESDR at far higher prices. The HDSR also has very in depth options for RF front-end frequency calibration.

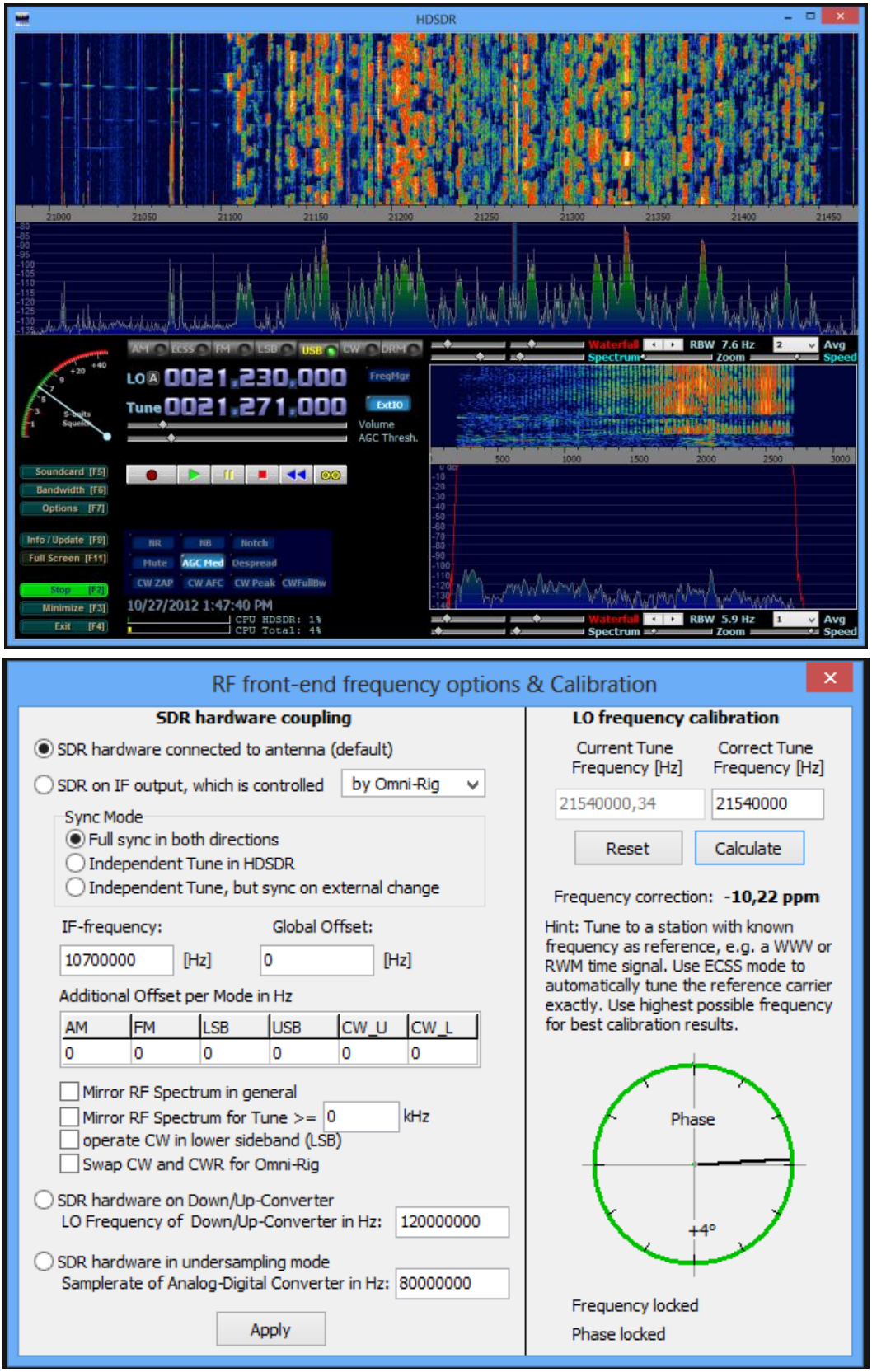


Figure 67. HDSDR Interface [23]

SDR#

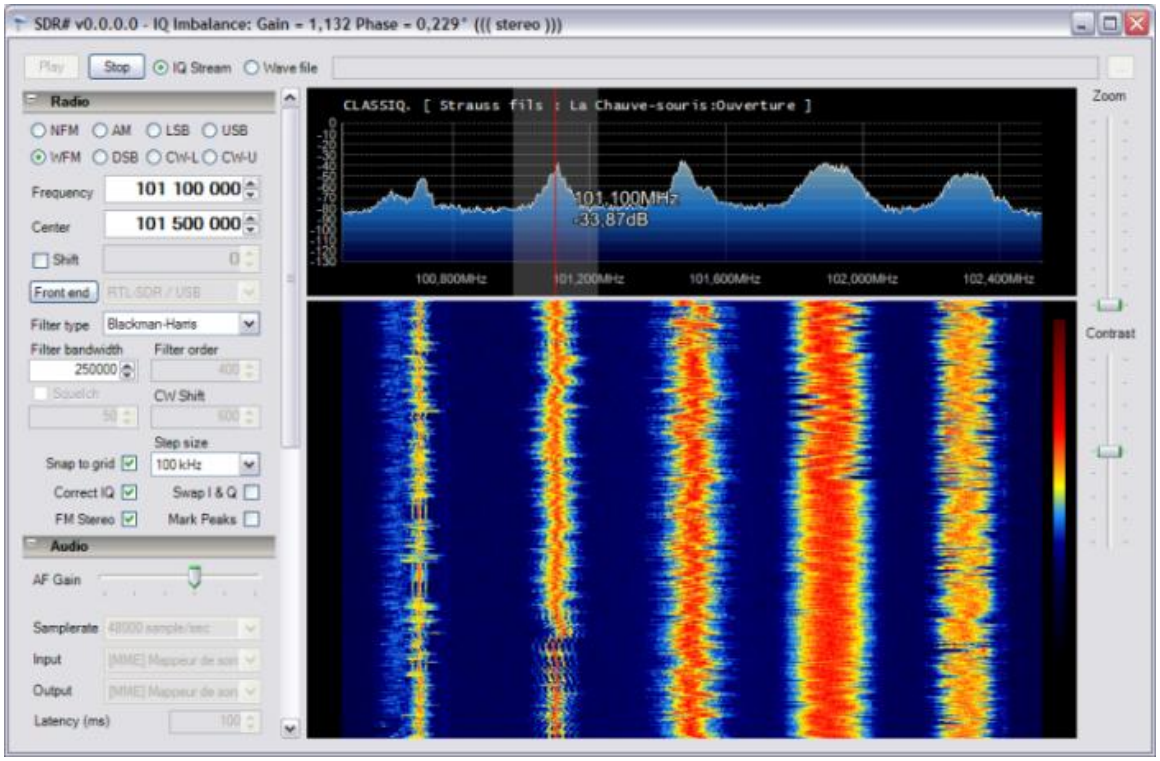


Figure 68. SDR# Interface [48]

Regarded as one of the best software defined radio programs available for entry level radio enthusiasts, SDR# provides a free and intuitive design for use on RTL-SDR. It is recommended that it is to be used with the $199 Airspy, but can be used free with any RTL-SDR.

SDR# was designed to have all the basic/advanced features that you may need as part of their standard installation package, but just as our product is designed to be modular and modified by the amateur radio club and community, SDR# offers modular compatibility with third-party plugins. As such, there are functions for almost any signal analysis and display task you could need for example, the fast fourier transform, waterfall display, and functions to reduce signal distortion. It is one of the most accessible software defined radio programs on the market; available on windows, linux, as well as on the Raspberry Pi Operating system (Raspberry Pi model 3b+ minimum hardware recommended).

Waterfall Page

The waterfall page is part of our stretch goals and would feature a waterfall/panadapter. Once the user has selected this page, the system will stop all unnecessary functions and allocate most resources to computing and displaying the waterfall. Functions will return to normal once a different page is selected.

Home Page

The home page of the SDR program will contain some of the most commonly used features for amateur radio enthusiasts including a display to see what frequency you are currently on, a seek feature to go to the next radio station with stable connection, an option to change whether you are on the AM or FM channel, an option to record media, and 5 customizable preset channels you can select from. Via the navigation bar on the top of the screen, the user is able to select between all other pages and functions available on the SDR software.

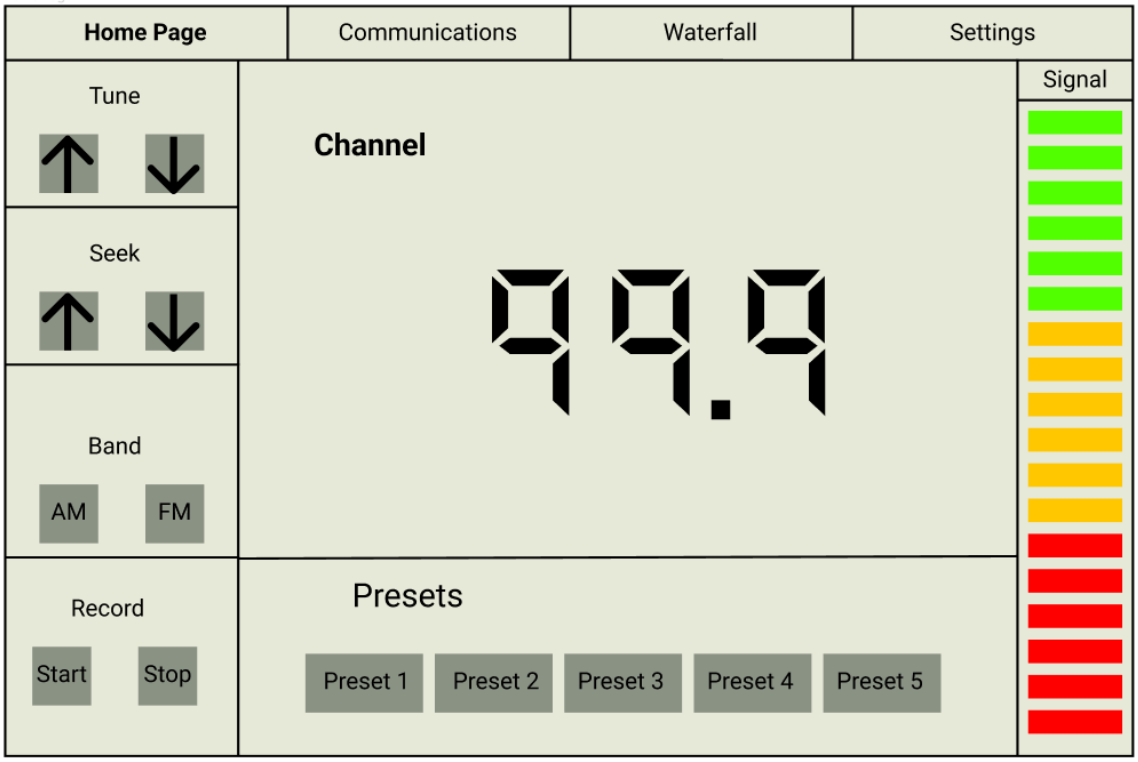


Figure 69. Home Page

Settings/Connections Page

The settings page of the SDR program will enable users to change certain features of the system as well as perform tasks such as updating the firmware of the system and its components whenever available. It features an adjustable volume and brightness control function, a power saver mode, an option to connect to a bluetooth speaker, an option for dark mode for accessibility, and a check for updates button.

If in the event the user was in a situation where he or she needed to have extended battery life, for example spending the day in a remote area, the power saver feature could be utilized. In order to maximize battery life the most power consuming tasks will be limited until power saving mode has been turned off. One of the most power draining pieces of hardware is the backlit touchscreen. During power saving mode, the touch screen's brightness will be limited to 50%

of its maximum value and will enter sleep mode after a predetermined time to be woken upon touch. The second piece of hardware that could potentially drain power quickly would be the speaker system since the higher a speaker's volume is set to, the more energy it will draw to amplify the sound input. To counteract this, the speaker's volume will be limited to 70% of its maximum volume (with the exception of a connection to a bluetooth speaker). The waterfall feature will also be inaccessible to the user due to its high visual and computational requirements of the embedded system.

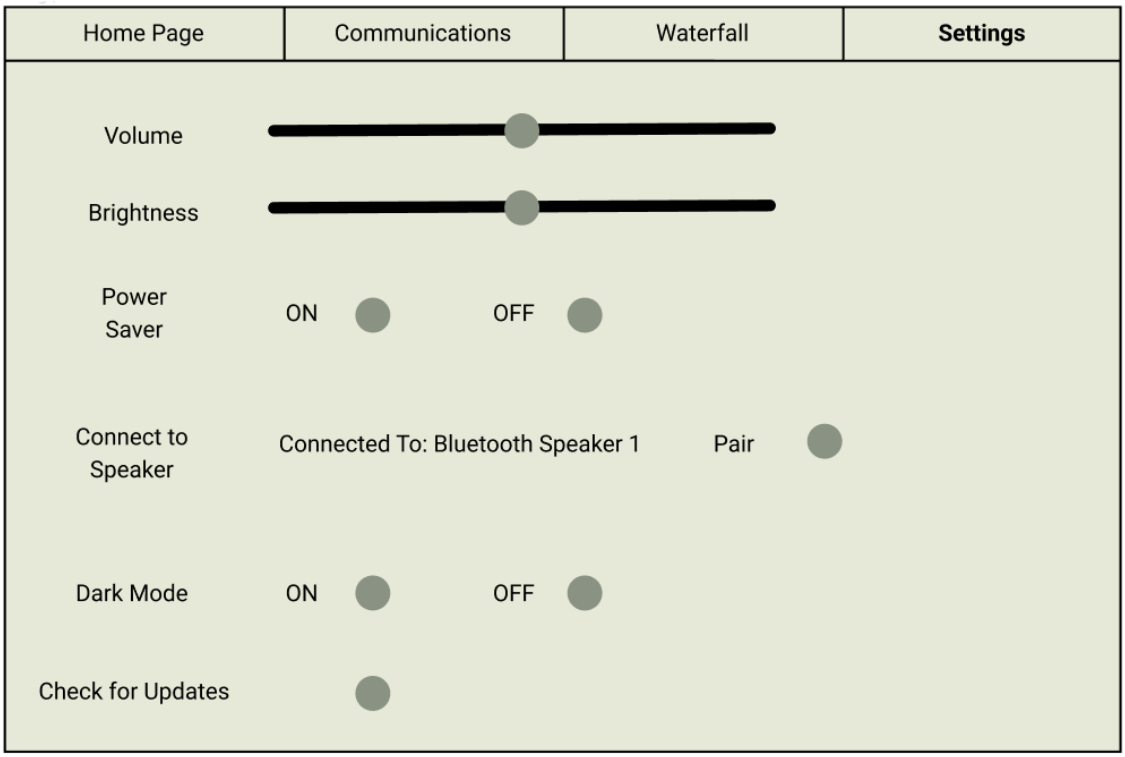


Figure 70. Settings/Connections Page

Communication Page

On the communications page, the user will have most of the same functionality as the home page with the exception of its limited amount of presets and a visual push to talk button. Once a user is connected to a channel and a band accessible for communications, they will be able to communicate with others on the same channel.

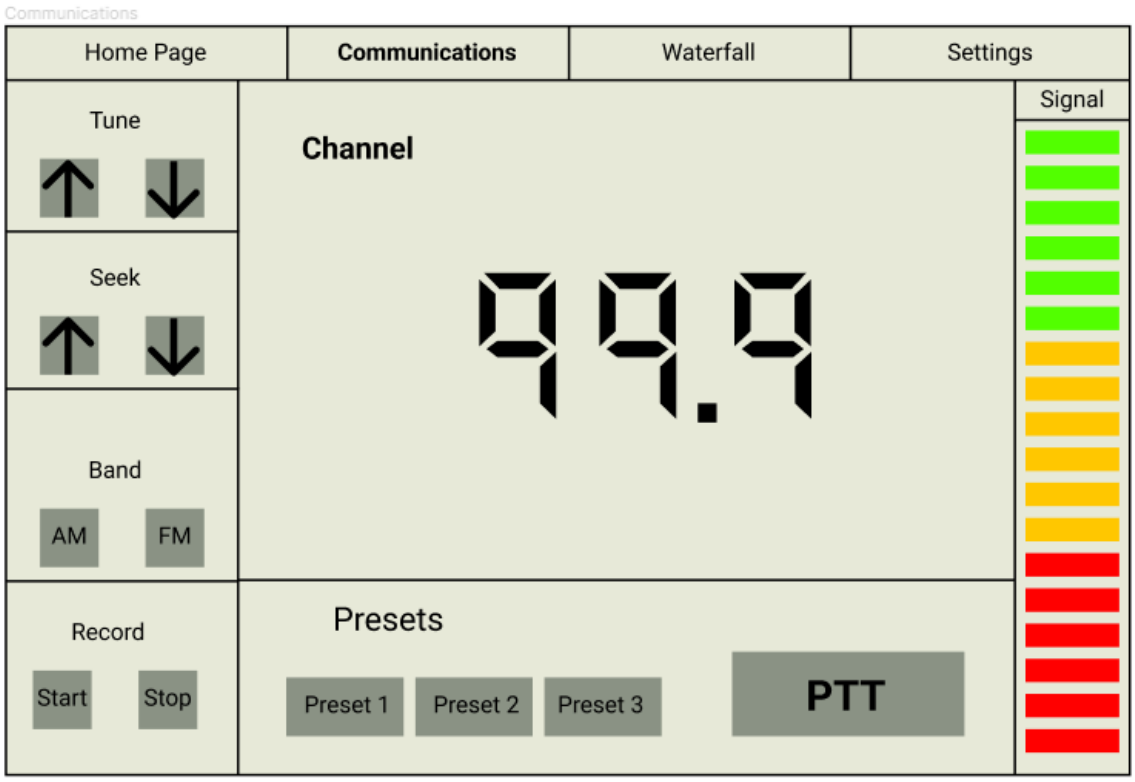


Figure 71. Communications Page

4.2.4 Microcontroller

Our system will be making use of the Raspberry Pi Zero 2 W as if it were a microcontroller to handle things such as communication, on/off signals, outputting sound, and more. While a lot of these functionalities will be enabled over USB communication as this is the easiest to deal with method of communication available, and supports a modular design, there will come a point where we will be required to interact directly with devices through the pins on the Raspberry Pi Zero 2W.

There is a built-in library on the Raspberry Pi systems called wiringPi.h. This library is useful to be able to do things such as send power out through a certain pin, or change the function of a pin if it is being used for multiple purposes. This library is based on C, so using it would be tricky for us. We plan on using Python instead of C so we will have to set up some kind of abstraction layer where we could call the functions from this C script from Python. The reason we would do this over just programming in C++ is that Python is a lot easier to use for developing the front-end of our software, and switching between languages frequently could introduce translation issues when going from one layer to another.

4.2.5 3D Printing

Our system will be exposed to the elements so a 3D printed case will be necessary. A bare metal design would make it immediately fall apart and stop working if it was out while in the rain, and a generic case would not properly fit all

of the components that we have since this is a unique design. When it comes to 3D printing, going from an idea in your head to a physical product is a multi-step process that involves both software and hardware processes.

Blender

Figure 72. Blender Logo

Blender is a 3D modeling program used by artists, video game developers, 3D printing enthusiasts, professional engineers, and more. Blender's main selling point is that it has a vertical slice of everything that somebody would need when 3D modeling, including modeling, rigging, and rendering. It is under the GNU General Public License (GPL), making it available for us to use during this project.

Although we will not be using the animation or rigging aspects of blender, we will be taking advantage of it to create a 3D model of what the case for our system will look like. Once this 3D model is finished, we will export it to Cura.
[1]
Cura

Figure 73. Cura Logo

Cura is software specifically designed for 3D printing. How Cura works is that it prepares models for 3D printing by converting them into a language that a

3D printer can understand. For example, by importing a model from Blender, Cura is able to convert that into .gcode which can be understood by a commercial 3D printer as instructions for how to print the aforementioned model. This is both extremely useful and extremely necessary for us to be able to print our 3D model of our system's case, as Blender itself does not directly support 3D printing and does necessitate the use of external software. Cura does allow use of a stripped-down version of itself for free which will be more than enough for our purposes since our case will essentially just be a rectangular prism with some mounting holes. [58] [59]

5. Integration (10-15 Pages)

5.1 Overall Integration

EMI is a significant integration challenge to minimize EMI on all pcb shielding will be liberally used since shields are cheap. The plan is to 3d print the casing and use off the shelf hardware to fasten everything together. The plan is to get the actual functionality running outside of the enclosure. So firstly we will test the SDR without any additional components to make sure we can modulate and demodulate while directly streaming to the pi. From there we will test the output when directly connected to the amplifiers and RF frontend without the pcb. From there the same test will be done with an attenuator in the loop to simulate the rf switch in the loop. Then the PCB can be ordered and full system testing can be done. Then the enclosure will be built and the final testing will be done.

5.2 PCB Design

One thing that was important to consider when designing the PCB was EMI the generally accepted standard is 1/17 the minimum wavelength is a good maximum trace length. We want the maximum frequency to be 900MHz which gives a minimum wavelength of .33m which equates to a desired maximum trace width for rf routes of 0.0195m. This is to prevent self interference and distortion. So for the design of the PCB the trace width was carefully considered for tracks that matter. This restriction was less important for the RX path so that is the longer trace.

Another thing to consider was connectors to and from the board for this project since the board itself need not be hot swappable the connectors will be soldered on. However the connectors that go to hot swappable components will also be soldered to latching jumpers. This will be matched by easily swappable connectors soldered to the amplifier. Lastly the actual RF connector was chosen to be high quality SMA with adequate frequency range.

The size was another limitation on the board; the board should not be wider than the desired screen size and as stated in the next section the components at least for this iteration were not chosen for small footprints. Because of this the pcb was designed to not exceed the width of the screen. Another physical characteristic we accounted for in PCB integration was mounting. M3 mounting holes were placed in each corner to account for this.

Component size was chosen based on necessity and ease of installation. Although the plan is to use a solder mask and reflow the entire board the minimum resistor size was chosen to be 0603 to be easily manageable. The capacitor size was chosen to generally be 0805 for larger capacitances in power filters and 0603 for smaller decoupling caps. The inductors were chosen to have appropriate saturation currents, dc resistances and shielding first so they came or to be rather large.

Another thing we needed to consider when designing the pcb was heat dissipation. Although the mosfets will have a low on resistance the PSU will still get hot at maximum heat draw for this reason extra thermal relief vias were added under the regulator ics. The heat vias were connected to large ground planes providing adequate heat dissipation and thermal mass.

For this first design the SMA connectors chosen were designed to be vertical facing. This will allow for easier testing. However SMA connectors generally use a similar if not the same footprint so it should be easy to find right angle connectors to save size in the final iteration with no redesign required. When the PCB is fully tested a second iteration may be built to account for size constraints and final implementation.

The last challenge in the pcb design was the RF switches which are designed to have an output impedance of 50 ohm. Because the frequency is relatively low even at 900 Mhz the impedance wont be high enough to be 50 ohm. Because of this the track was just designed to be short so as to not add extra impedance on top of the 50 ohm SMA interconnects. The track lengths of the RF section were designed to be short and rules of thumb were used for layout. These PCB design techniques should be adequate for our RF frequencies and purposes but after testing a second iteration may be necessary.

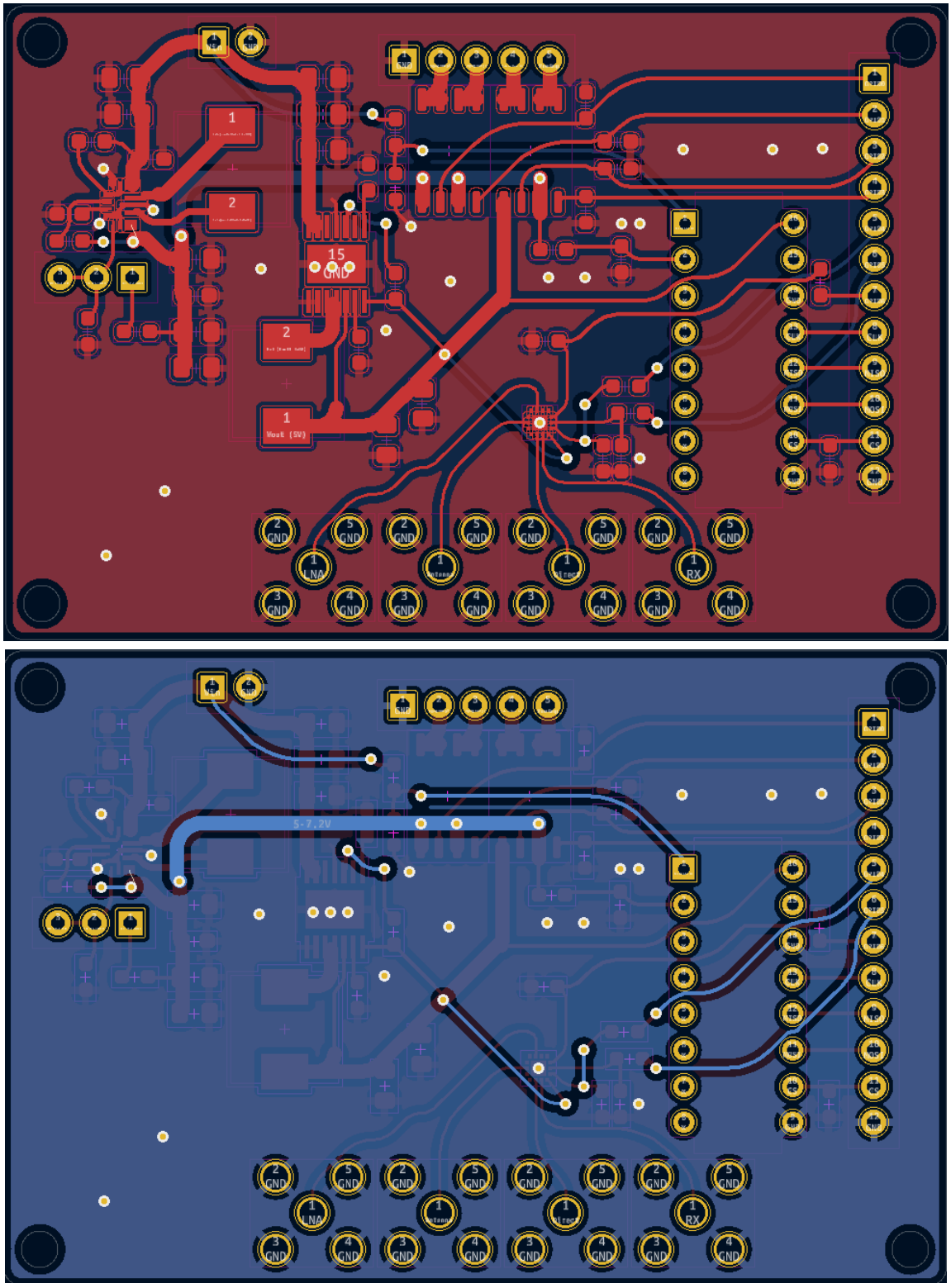


Figure 75. PCB Front and Back

5.3 Overall System Testing

5.3.1 Test Configuration/Environment

The project was primarily tested in a live laboratory environment within the University of Central Florida, in which we utilized many resources provided to us by the University such as Tektronix Oscilloscopes, Tektronix Dual Arbitrary Function Generators, Keithley 2230-30-1 Triple Channel Power Supplies, and Tektronix DMM 4050 Digital Multimeters. This lab was accessible at any time via an electronic door lock that all senior design students were given unlock access to. This lab also featured SMD Rework stations, Soldering and desoldering stations and digital microscope inspection sections that were provided with supplies such as jumper cables, electrical alligator clips and other basic necessities commonly associated with the equipment being provided. The equipment provided such as the Tektronix oscilloscopes were verified for their accuracy and functionality. Within this lab we set-up and tested all of our RF Amplifiers, individual parts as well as we will assemble our PCB as this lab also provides other instruments such as a reflow/infrared oven for the soldering of surface mount components.

Some such testing we were able to do was to utilize the soldering station provided by the university to set up our initial testing environment such as putting live wires to V_{in} and GND to be able to utilize the RF filters and amplifiers that were previously discussed. One such example is shown below such as the 900 MHz RF Filter utilized in such manner to be able to be given power from the Keithley triple channel power supply in an easy manner for ease of use and testing:

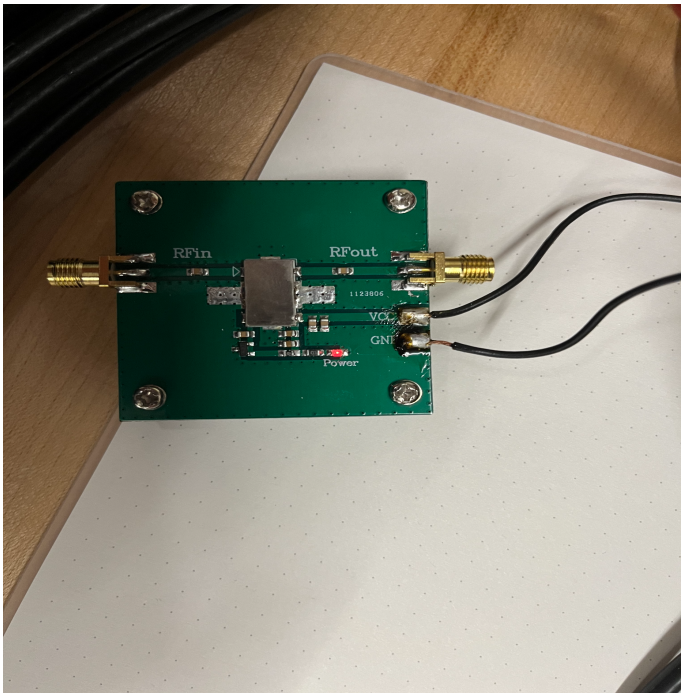


Figure 76. 900MHz RF Filter Environment Preparation

We then had to prepare our testing environment with another radio in order to ensure we can produce a similar environment for testing these components to our actual radio environment. For this, we chose to use the Icom T2H handheld radio for signal generation connected to an antenna then received by another antenna utilizing an SMA to BNC connection connected to an oscilloscope. This was able to let us test the effect of the LNA without needing a low power UHF frequency generator. This method was also used to test the other amplifiers in and outside of their relevant bands. A picture of this general testing setup is shown below.

Figure 77. SMA to BNC connection

As we can utilize a very similar setup for all of our filters and amplifiers, we can utilize this testing environment to simulate a similar response out of our ICOM radio comparable to our actual finished product. Pictured within the above picture is the low noise amplifier, showing that we are able to utilize this connection regardless of the type of filter as the RF filter pictured above has the same connector. The input power can also be controlled by measuring with an rf meter at different distances from the filter attachment when the PTT is on. Showing the controllable power capabilities.

5.3.1 Part Testing

Speaker Testing

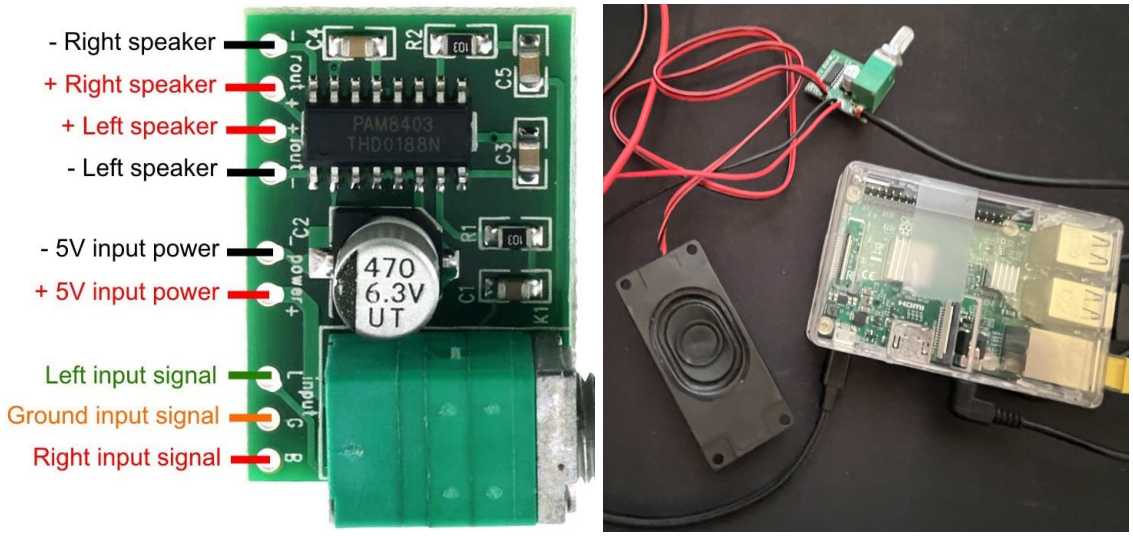


Figure 79. Speaker and Raspberry Pi [11]

Objective: The speaker will be tested for audio clipping, support for Linux devices, support for Windows devices, output volume, and overall audio quality.

Environment: The Speaker was tested in the Radio Room at UCF in the back of the SD lab. It was connected to a Raspberry Pi running Raspberry Pi OS Lite, as well as a desktop computer from the SD lab running Windows 10.

Procedure:

1. Solder speaker connections
2. Connect to Raspberry Pi
3. Attempt to output audio
4. Connect to Windows PC
5. Attempt to output audio

Conclusion:

To test the speaker we had to solder the connections from the speaker's board to the speaker ourselves. This involved using the soldering iron in the Senior Design Lab. The speaker system we bought comes with both a left and right speaker, but for the purposes of testing we only soldered one speaker to our system.

We were unable to output audio through the Raspberry Pi as we had not figured out how to output audio through Raspberry Pi OS Lite at the time of testing. By connecting the speaker to one of the desktop computers at the Senior Design Lab, we were able to make it output audio from a youtube video. One of the problems we ran into was that when the speaker's volume dial was dialed past the halfway point, the audio would begin to stutter and cut out. When dialed even further past the halfway point, it began making a chirping sound similar to a fire alarm. This issue was difficult to troubleshoot.

After connecting the speaker to multiple systems, we noticed that the point at which it began to chirp changed based on which laptop it was connected to. On all platforms the speaker was still loud enough to hear at the points before the chirping began. Due to this, we plan to simply ignore this problem as it does not merit buying a different speaker as that would cost more money and take up more space than this one does.

Microphone Testing

Objective: The microphone will be tested for audio clipping, support for Linux devices, support for Windows devices, input volume, and overall audio quality.

Environment: The microphone was tested in the Radio Room at UCF in the back of the SD lab. It was connected to a Raspberry Pi running Raspberry Pi OS Lite, as well as a desktop computer from the SD lab running Windows 10. At the time the room was quiet and there was little echo at the volumes we were speaking during testing.

Procedure:

1. Connect to Raspberry Pi
2. Attempt to record audio
3. Connect to Windows PC
4. Attempt to record audio
5. Connect a different microphone to a Windows PC
6. Attempt to record audio
7. Compare audio recordings

Conclusion:

The microphone that we were testing connects to a USB 2.0 port and is supported by both Linux and Window devices. Testing this device involved no breadboarding or soldering as it only had to be plugged into a working system.

Once again we were unable to measure audio being inputted to the Raspberry Pi as Raspberry Pi OS Lite as there was no transmission medium available to indicate success. Additionally, installing the drivers to have the device work on Linux devices was difficult to do over an SSH connection. Because of this, we decided to test the microphone by connecting it to a Windows computer.

Once we devices to test on a Windows computer, drivers became an issue again. Windows had attempted to recognize the device and automatically install drivers to support the device, but the drivers that Windows installed were not compatible with the device. Troubleshooting this took a while, and we ended up having to manually uninstall the drivers that Windows installed and going to the manufacturers website to install the drivers on our own. Once that was done, we were able to reset the computer and test out the microphone using Audacity.

We compared the performance of the microphone to that of the built-in microphone in one of our laptops. The top waveform corresponds to the laptop microphone, while the bottom corresponds with the USB microphone. When speaking at the same volume in the same environment we noticed that the USB microphone is a lot more sensitive, and is more susceptible to noise overall. Additionally, when speaking loudly we often ran into issues related to audio clipping. Due to the nature of the device and the limited resolution of audio transmitted over amateur radio bands, this clipping has been deemed acceptable and we have decided that the microphone we chose is still good enough.

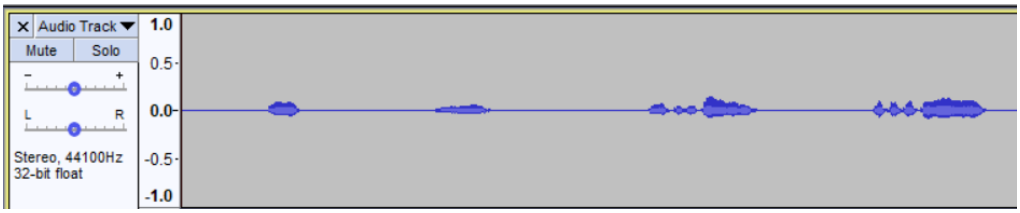


Figure 80. Laptop Microphone

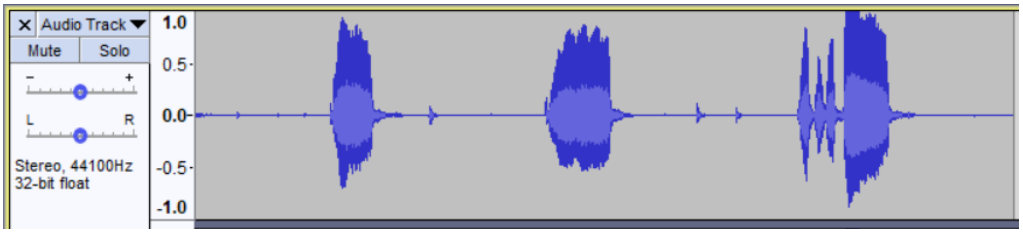


Figure 81. USB Microphone

RF Component Testing

Each component in the RF system must be tested to ensure it meets the requirement specifications and the FCC standard and constraints. To ensure this, each component must have a testing protocol tailored to its purpose.

Amplifier Testing

Each external amplifier will be tested with an oscilloscope to check for noise sidebands and EMI.

The Power Amps will be tested at different voltage inputs with an RF meter to find the power output vs input voltage curve that can be applied to correct the readings later.

Antenna Testing

Objective: The Antenna will be Tested for VSWR, output power vs absorption, and heat dissipation.

Environment: The Antenna will be Tested in the Radio Room at UCF in the back of the SD lab a nano VNA will be used for VSWR and an RF power meter to test output power. And an ICOM radio with controllable output power was used for power and transmission testing.

Procedure:

1. The antenna will be tested with a portable RF meter to see transmitted vs absorbed power.
2. The Antenna will be tested with a VNA to see that it meets impedance and VSWR characteristics.
3. Lastly the Antenna will have 5 Watts of RF power pumped into it at each band separately to ensure the heat is properly managed.

SDR Testing

Objective: The output of the SDR will be tested with an Oscilloscope to ensure high frequency harmonics are present on the output.

Environment: The SD Lab was used with a Raspberry pi 3 running Raspberry pi os and 5Gsps Oscilloscope and a Laptop running windows 11. For the SDR the Lime Mini has not shipped so the equivalent full size LimeSDR was used for this test.

Procedure:

1. The LimeSDR was attached to the Windows 11 laptop and Lime Suite Was installed. The connection was then tested to ensure the SDR was working properly.
2. The LimeSDR was then connected to the Raspberry Pi Lime Suite was installed and connected to the LimeSDR
3. Test output waveforms were then uploaded to the SDR in a loopback configuration.
4. The same Test Files were played again with the SDR connected to the oscilloscope to test for high frequency harmonics.

Conclusion:

The Lime SDR was unable to connect to the Windows 11 laptop due to driver issues. So the LimeSDR was then connected to the Raspberry which successfully verified the LimeSDR was working. From there the loopback test was attempted but the LimeSDR crashed while updating; will be attempted again.

SDR connection Testing

Objective: the LimeSuite Quick test will be run on the raspberry pi to ensure the LimeSuite SDK works on the Pi and the USB port can properly communicate.

Environment: A raspberry Pi 3b will be connected to the LimeSDR and the loopback test will be done from the LimeSuite documentation.

Procedure:

1. The LimeSDR will be connected and the firmware will be upgraded using Limesuite.
2. The given configuration file will be loaded into setting the RF switched to loop back the SDR.
3. The Given play file is transmitted through loopback an an FFT is measured

Conclusion: the following output was shown on the Raspberry Pi showing an FFT that closely matches the result expected by the documentation proving the SDRs functionality in multiple modes.

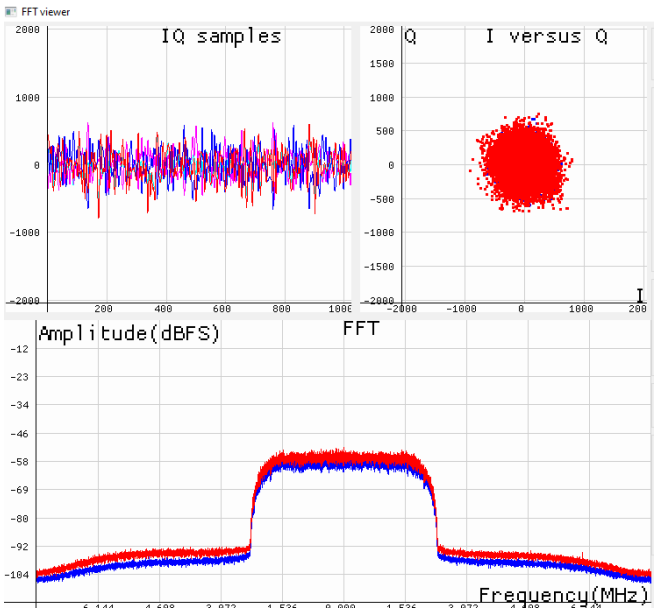


Figure 82. LimeSDR connection and loopback test.

6. Administration (5-10 pages)

6.1 Milestone Discussion

For this project we are restricted due to both due dates imposed on us by UCF/ABET, and due dates imposed on us by our sponsor. Because of this, we have to make sure we weigh the priority of different tasks to ensure that we are able to meet the due dates for both of our stakeholders.

These timelines account for mandatory due dates of when things have to be turned in, and progress based due dates which would serve as a good sign of whether or not we are making enough progress on the project to finish on time. The overall time should roughly follow these diagrams, leaving a large amount of time for integration and unpredictable challenges.

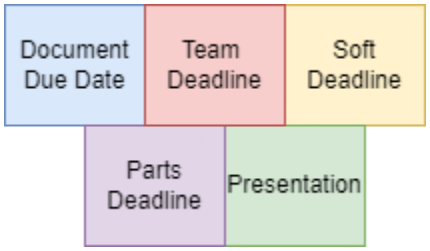


Figure 83. Timeline Legend

ABET Timeline

Figure 84. ABET Timeline

Radio club timeline

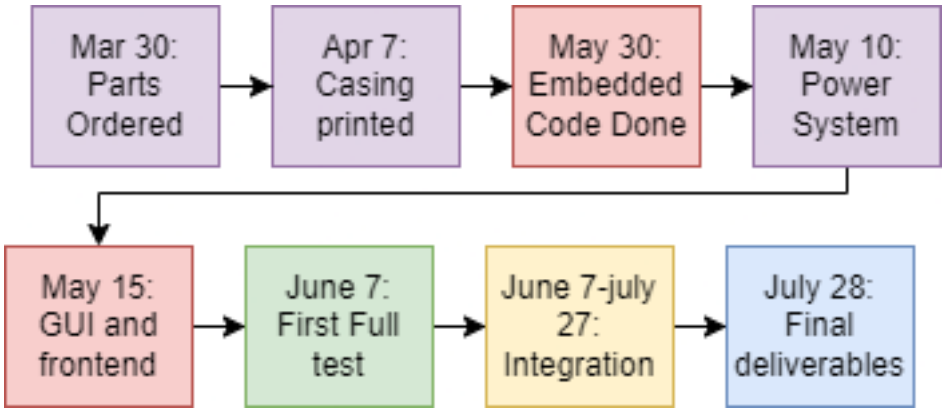


Figure 85. ARC@UCF Timeline

Combined Timeline

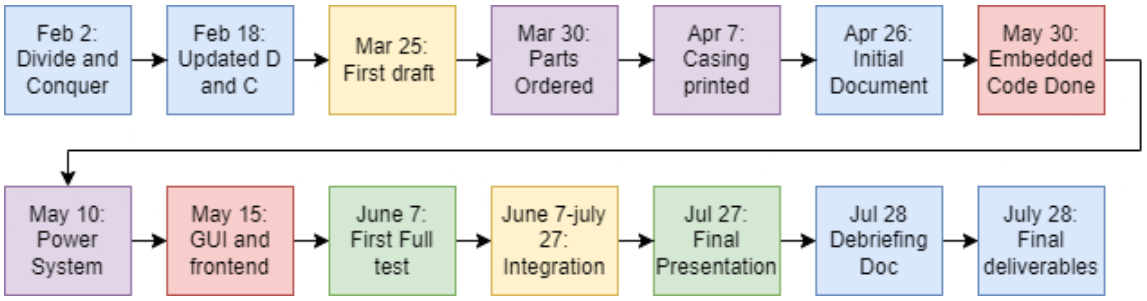


Figure 86. Comprehensive Timeline

During the entirety of Senior Design 1 we were able to meet most of the deadlines provided to us with the exception of one due to the shipment of some of our components being delayed. In order to achieve this, we followed strict weekly meeting times consisting of a virtual meeting on Tuesdays and Thursdays ranging from one to four hours per session. Depending on if we experienced any stumbling blocks throughout the phases of the project or not, a meeting on Saturday or Sunday was expected to catch up on any work that we are falling behind on. By following the timeline set forth by the administration (depicted above) and ourselves (Section 3.2.10: Software/Hardware Standards), we rarely found ourselves working at the last minute. Meetings that landed on the day on or before the submission deadline were spent reviewing and editing the document, ensuring that we are meeting page requirements and that our document is of professional quality.

As we reach the end of Senior Design 1, we have already entered the prototyping phase of the project and are ahead of schedule for Senior Design 2. We are planning to spend the time between the end of Senior Design 1 and the beginning of Senior Design 2 reviewing what we have done as of yet, completing the majority of our final PCB design, and integrating most of our components to the embedded system. Currently, we do not foresee any circumstances where the project cannot be completed and are confident that we will be able to meet all the milestones presented to us during Senior Design 2.

Since the second semester of the project requires hands-on assembly and testing, our usual meeting format cannot suffice and we will have to adjust accordingly. To counteract this we will be utilizing the facilities at the University of Central Florida and our homes to conduct meetings. There are expected to be situations where one or more team members are not able to meet in person during these times, so we will have an active voice chat and a team member recording all necessary information to ensure that everyone stays up to date with the current development progress.

6.2 Budget and Finance Discussion

The Club Faculty Advisor of ARC@UCF and CECS computer cluster researcher, Steven Dick, will be sponsoring this project on behalf of ARC@UCF as well as providing access to their laboratory and tools they have available. In addition, Steven will be doing his part to provide as many of the materials as possible so that we would not have to spend as much money. Steven set our development costs at around $500-$700, most of which would go towards the LimeSDR Mini and the Raspberry Pi Zero 2W. This budget excludes additional accessories such as GPS that can be implemented as a hardware extension.

Due to supply chain issues, we have had trouble ordering the LimeSDR Mini, as well as the Raspberry Pi Zero 2W, which are the most important as well as most expensive parts of our system. To get around this, we have been using a full LimeSDR, as well as a Raspberry Pi 3. This cut hundreds of dollars off of our development costs, and allowed us to come significantly under budget when selecting parts. In regards to the rest of the products we ordered, the list of products we ordered was approved by our sponsor since it was well under the $700 spending limit set on us at the beginning of Senior Design 1.

The Bill of Materials outlined below shows every component that we have ordered and plan on using in our project design. Additionally, we have a separate column for what the costs would be when these parts are ordered at a larger scale (1000+). Note that some of these parts are to be used purely during development, such as the LimeSDR, and will be replaced later on with different parts, such as the LimeSDR Mini.

Development costs are not an aspect of this design that we have to worry very much about, as our sponsor is going to keep any parts we ordered after the prototype is done, so no components will be tossed out.

Production costs are an important factor to consider when designing our project. Part of the intent behind this project was to make an open-source SDR radio which an average amateur radio enthusiast could easily reproduce on their own. Because of this, if any of the parts that we ordered were prohibitively expensive, it would defeat the purpose of our project. Our budget for production costs comes in at around $300-$500, so the estimated total price of our parts when ordered at a large scale must be within $500. Researching, we found that radios that generally perform similar functions to our SDR radio are significantly more expensive, often in the $5000 range. This means that the system we are designing would significantly lower the monetary barrier to entry for this specific part of the amateur radio hobby.

Keeping track of how much we were spending was important to our sponsor because everything that we were ordering was being paid for directly by their personal checking account. Because of this, we kept a thorough list of everything that we needed and we had the sponsor themselves order the parts and send them to us after they had looked at our list of required components and approved them. This allowed them to give us any suggestions if they believed that a specific component was overpriced or if we had chosen a component which they have had previous negative experiences with.

Overall, it was relatively easy to meet the financial constraints imposed on us by our sponsor. The development costs came up hundreds of dollars under budget due to supply chain issues preventing us from getting what we even wanted, and the amount allocated for production costs was a generous estimate. As this product is further developed after our share of development, I can see the production costs lowering significantly, especially if some kind of deal could be made with the producers of the LimeSDR, as the LimeSDR is the main expenditure. Alternatively, this product could be sold as an add-on kit with everything except the LimeSDR, and likely come in at easily under $100.

Component	Description	Quantity	Vendor(s)	Estimated Cost
LimeSDR Mini	Software defined radio component, primary core of SDR radio	1	Limemicro LLC	$199
Raspberry Pi Zero 2 W	Embedded Control CPU	1	iUniker (Amazon)	$10
144MHz Amplifier	Radio Frequency Amplifier	1	eBay	$20
433MHz Amplifier	Radio Frequency Amplifier	1	eBay	$12
Antenna	SMA antenna for required bands	1	Luiton	$20
Touchscreen Display	Resistive based water resistant HUD for user to interact	1	JniTyOpt	$30
Speaker	Multipurpose speaker(s) for signal production	1	Degraw	$12
Microphone	Multipurpose	1	KISEER	$10

	microphone			
Battery/Battery Controller	Battery and casing system capable of supporting power to system	4	Mab	$7.49
Battery Charger	Charges batteries	1	EBL	$7.99
Battery Case	Holds batteries	2	Digikey	$6.35
Waterproof Casing	Protection of radio system and casing, 3D printed	1	-	$20
Gasket	Sealant for waterproof casing in case PLA is used	1	-	$12
MCP3008	8 Channel 10 Bit SAR ADC	1	Digikey/AdaFruit	$3.50
USB Hub	4-Port USB Hub to support multiple USB devices	1	Degraw	$7
Miscellaneous	Electronic components (Resistors, Capacitors, etc)	-	Mouser, Digikey, Etc	~$10
Total Estimated Cost	$395.50			

Table 20. Estimated Cost Table

6.3 Safety Precautions

Safety is one of our most important concerns when designing the project. As opposed to some senior design projects where the goal is to simply produce a device solely for the final demonstration, our project is going to be taken by the Amateur Radio Club and developed further. When they obtain the device, it must be functional and designed in a way where the device cannot cause any bodily harm to the user.

As stated multiple times throughout the project, the main component that is a safety hazard is the lithium-ion battery cells. There are several procedures we will undergo in order to inspect the quality of the lithium-ion battery cells before we turn them over to the Amateur Radio Club including: delivering the product with discharged batteries or advising them that there is charge in the device, providing new batteries that we did not prototype with, and ensure they are aware of proper lithium-ion battery disposal practices. By doing these tasks as well as any other standard procedures we mitigate the risk of injury and limit developer responsibility.

There are many things the user can do to safely use and ensure that the portable software defined radio will last long. First and foremost, as with all electronics, regular maintenance should be performed on the device including: regularly charging the battery cells with either a reputable charger or the one provided, removing any dust and debris from the crevices and speaker system, periodically checking cables and connections. Aside from general maintenance, the user is expected to store the device in a cool/humidity-free environment, avoid extreme weather conditions and large bodies of water if possible, and never use harsh chemicals on or near the device. It is also recommended but not necessary to use gloves and protective eyewear while working on the internals of the radio.

6.4 Troubleshooting

While developing the device we thought to consider some potential problems which might occur during development. A common problem during development is misattribution of errors to sources, so we decided to create a table to help associate common problems with common solutions. By making this, during development we will more easily be able to prevent ourselves from getting stuck because we think something is causing a problem when it is actually something else causing the problem. This allows for effective troubleshooting while at the same time helping accomplish the open source goals.

The general approach to solving an undefined Problem will be to first isolate the problem as much as possible. Then find all the relevant modules/ components. Then to check the relevant input of the component. Relevant input in the broad sense meaning any system that directly influences the performance or functionality, This could be something like the connections, the Power or the Logic. From here if a faulty input is not discovered the system itself will be

isolated and tested with a working system to ensure functionality. If this does not show attempts to diagnose a problem with this general approach will be applied on the system itself. Recurrently iterating until the problem cause is discovered or a fault component that can be replaced is found. If this system does not work most likely the module itself has issues with multiple interactions and needs to be fully replaced.

Problem	Solution
Touchscreen wont turn on	● Check Power connection ● Check HDMI connection
Submerged Device	● Immediately shut down device and place in an electronics safe water absorbing material
Low Audio Quality Distortion Hissing or Whistling Humming Clicking, Buzzing or Pops	● Check speakers for dust and debris ● Search for stronger signal ● Make sure you are connected to one signal ● Refer to Battery Malfunctions below ● Refer to Poor Signal Below
No audio Output	● Check 3.5mm cable connection ● Check power connection ● Make sure system is not on mute or is set to a low volume
Corrupted Memory	● Eject and factory reset the microSD card. Reinstall Raspberry Pi OS as well as the SDR software and any necessary drivers.
Battery Quickly Loses Charge Battery Malfunctions Device Wont Turn On	● Check discharged voltage of the battery cells to make sure they are not dead ● Check charged voltage to ensure battery cells are functioning properly ● Replace batteries ● Charge Batteries
Poor Signal	● Make sure the signal is not impeded by walls, hills, structures, or mountains. ● Set antenna vertical ● Tighten Antenna

Table 21. Troubleshooting Table

7. Conclusion

When this project first started, we were confident that everything would go smoothly as we were doing a funded project via the Amateur Radio Club. As we are nearing the end of Senior Design 1 we can confidently say that although there were some unforeseen circumstances that we will discuss later, we were able to design a portable software defined radio with minimal setbacks.

The requirements that the Amateur Radio Club have set for their device were fairly standard when compared to similar products available on the market today. These features included: casual listening to frequencies on multiple bands, a battery life that lasts for several hours, a display and control system to access features, T/R functionality so a single antenna can be used, 5W of transmission power on specific bands, and finally an open source design that is fully programmable from either a laptop or desktop. Similarly, there were optional features of the device that we could implement at our discretion as the expected project timeline as described by the sponsor stated that they planned to utilize two separate Senior Design 1 and 2 sessions so that they could implement additional features as well as perfect the system design. Alongside their list of optional features, we also had the opportunity to propose alternative system designs and features we thought would be beneficial to either the lifespan of the device or the end user's experience. These features included: GPS support, a touch screen module, a waterfall display/panadapter, data storage, internal battery charging and battery level indicator, expandable communication capabilities such as bluetooth and usb interfacing, and waterproofing measures.

Upon reviewing these requirements we decided on several features that we would support in the initial design of the device. The main features that comprise our portable software defined radio are: a touch screen and inputs to control both the hardware and software components, a LimeSDR to receive radio transmissions, a suitable amplifier, an integrated power system, and the Raspberry Pi Zero 2W for the embedded system to tie in all our components. Although we do not have any materials or modules to support GPS, waterproofing, or the panadapter, we will try to implement low cost solutions after the device is in an operational state.

While researching the necessary components we needed to design our portable software defined radio based on the requirements set forth, we had to keep several circumstances in mind. The most important drawback to our project is that with the budget that the Amateur Radio Club provided for prototyping and development we needed to order parts in varying quantities so that our construction of the device will not be hindered by having a crucial component malfunctioning, getting destroyed in prototyping, or being dead on arrival; and this would consume a vast majority of our budget. Not only would buying several units of the same component hinder us, we are currently in a semiconductor shortage. Due to this, there is a limited availability of components and many components are no longer in production. There were several times we had to purchase a part from a third-party retailer and the prices were inflated to several times of the original market prices.

When we first started this project, we were all confident in our capabilities as engineers to complete the assignment and meet the expectations of our sponsors and advisors. As such, we delegated the workload at the beginning of the project based on the skills that each person held and adjusted assigned tasks whenever necessary. Since all project team members have a general idea of what professional field we would like to enter post-graduation, we did not assign any tasks to an individual with the goal of strengthening their weaknesses. For example, those who are to work with electrical hardware will be doing minimal programming and those who are to be software oriented won't develop core hardware solutions. Instead, we delegated assignments based on the individual's strengths in order to develop the best product we could as well as to hone in on our skills in our desired fields.

Due to our open source design, we have decided to select programming languages that either already have massive community engagement/documentation or can be learned relatively easily so that the average person can start their amateur radio experience. Out of all the languages that are available to us, we have chosen python to be the core of our software program as it has the easiest syntax, smallest learning curve, and one of the largest active communities out of all the programming languages. Although we intended to create most of our program in the python language, we have to use more complex languages such as C/C++ in order to communicate with the radio receiver module. Despite this, users are not expected to change the source code we have as we will provide most of the functionality a basic user would need; alongside this, the more experienced user should have the skills and knowledge to edit the integration code with the aid of online documentation.

For prototyping we were able to verify multiple system modules. Firstly we showed the LimeSDR's functionality with a loopback test we were also able to confirm the LimeSDK works with the Pi because the LimeSuite tool uses that specific SDK. from here we also verified the functionality of the PI for the Following: Firstly the communication with the LimeSDR was verified through the API in the loopback test. Next the communication with the Screen was verified through the HDMI and touch screen test. Lastly the functionality of the amplifiers was partially tested with the attenuators test.

The first iteration of the PCB was designed to match the requirements and specification given in the design section. The RF section was given adequate isolation and shielding for the required bands. The power supply traces were designed to provide the maximum required current continuously. Adequate heat dissipation was accounted for for all components with issues. This board should more than meet the design specifications however if needed additional iterations are a possibility.

During our research and prototyping phases this semester, we have identified some challenges that we might run into during the development portion of Senior Design 2. While prototyping the LimeSDR receiver and Raspberry Pi Zero, we found that the drivers given to us might not support everything that we had hoped to complete from the requirements list. Due to this, we will have to create either our own drivers or an additional driver interface that is able to

communicate with the pre-existing drivers so that we can get the best input signals. We also might have to upgrade the embedded system we are using to a more robust model such as a Raspberry Pi model 3 B+ or a BananaPI since the one we are currently designing our device around comes with very limited resources and Input/Output options.

On the hardware side, we will run into efficiency and power issues in the beginning but after identifying the problems on the first iteration of the PCB we will be able to solve them. Primarily, we are trying to reach a goal that consists of the battery life for the device lasting at least five to six hours. Standard 18650 lithium–ion batteries come in a variety of milliamp hour capacities and a simple battery upgrade could suffice, but since we have not recorded the power draw of the full system at maximum performance we will not know how much of a power difference we will need to compensate for. The RF components of the radio receiver system will also generate a tremendous amount of heat energy that we will not experience throughout the short periods of use during testing. We will have to wait until we have reached considerable progress in our development to see how much heat is generated and if it has the ability to affect the performance of the neighboring components.

Throughout the development of our project there have been multiple instances where we have considered what additional impacts our device will have to the amateur and hobbyist radio communities. As of right now our projected impacts can be condensed into three categories: social, political, and environmental.

For our social impact we expect that our product will be used and maintained by the Amateur Radio Club and that they will produce more devices to be used by radio enthusiasts. Since we also offer a modular, open source design, we also intend to draw in users that have no prior interest in radios. They are able to pick and choose what features are a part of their device and can add software functionality as needed so the possibilities are endless. Our device features an Embedded System capable of running many different operating systems and is able to connect to the internet via WIFI; so users could even create an entire computer system with a fully integrated radio onboard.

Since we are making a device that has similar products already available on the market we had to create selling points that would make our portable software defined radio more attractive to the Amateur Radio Club than the option of already buying a prebuilt system for their applications. Our product is expected to cost considerably less than the standard portable software defined radio and should have a growing user base. If the Amateur Radio Club decides to market the device there is the potential that other companies might take notice of the product and produce a device that shares our open source design. The companies we chose our components from might also restrict us from purchasing large quantities of their items; which will result in a redesign of our device with new components.

Our software defined radio has the potential to last years if properly maintained and as such, it will have very little negative impact on the environment. Users are expected to charge the provided lithium-ion battery cells

and dispose of them at an approved electronic materials recycling center after about 300-500 recharge cycles where the battery cells will be deconstructed and all usable parts will be sent back into production. Similarly the components we chose came from reputable suppliers and are also expected to have a long life. For minor components such as the microphone, one with superior quality can be chosen prior to assembly to increase performance and overall device life.

8. Appendix

Works Cited

[1] “About — blender.org.” *Blender*, https://www.blender.org/about/. Accessed 25 April 2022.

[2] “AD603 Voltage Adjustable Gain Amplifier.” https://ebay.to/3MYfD7L.

[3] “Adafruit I2S MEMS Microphone Breakout.” *Adafruit Industries*, https://www.adafruit.com/product/3421. Accessed 9 April 2022.

[4] “Balun One Nine v1 - Tiny Low-Cost 1:9 HF Antenna Balun and Unun with Antenna Input Protection.” *Amazon.com*, https://www.amazon.com/NooElec-Balun-One-Nine-Applications/dp/B00R09WHT6/ref=psdc_667846011_t3_B01N2NJSGV. Accessed 9 April 2022.

[5] “BK-18650-PC4 MPD.” *Digikey*, https://www.digikey.com/en/products/detail/mpd-memory-protection-devices/BK-18650-PC4/2330513. Accessed 9 April 2022.

[6] Brooks, Emilie. “Lithium Extraction Environmental Impact.” *Eco Jungle*, 31 December 2021, https://ecojungle.net/post/lithium-extraction-environmental-impact/. Accessed 9 April 2022.

[7] “CaribouLite RPi HAT.” *Crowd Supply*, https://www.crowdsupply.com/cariboulabs/cariboulite-rpi-hat. Accessed 26 April 2022.

[8] “CEA 861-F-2013 (ANSI) - A DTV Profile for Uncompressed High Speed Digital Interfaces.” *ANSI Webstore*,

https://webstore.ansi.org/standards/cea/cea8612013ansi. Accessed 9 April 2022.

[9] “CHASDI Omnidirectional 3.5mm Dongle Microphone.” https://www.amazon.com/CHASDI-Omnidirectional-Microphone-Controller-Mirrorless/dp/B07QJ9KDGS/.

[10] “COBOL Report Apr60.” *Wikimedia Commons*, https://en.wikipedia.org/wiki/File:COBOL_Report_Apr60.djvu. Accessed 25 April 2022.

[11] “Degraw DIY Speaker Kit.” *Amazon.com*, https://www.amazon.com/Degraw-DIY-Speaker-Kit-Amplifier/dp/B07CRVRG83/. Accessed 10 April 2022.

[12] “Design and Analysis of a Continuously Tunable Low Noise Amplifier for Software Defined Radio.” https://www.researchgate.net/publication/331735915_Design_and_Analysis_of_a_Continuously_Tunable_Low_Noise_Amplifier_for_Software_Defined_Radio.

[13] “Diamond HT Antenna 2m/70cm, BNC, 15in.” *Amazon.com*, https://www.amazon.com/Bundle-Diamond-Antenna-Guides-Reference/dp/B00WOLPBSO/. Accessed 9 April 2022.

[14] “DIY Bluetooth Speaker Box Kit.” https://www.amazon.com/Bluetooth-Speaker-Electronic-Sound-Amplifier/dp/B0871G4166/.

[15] “DSP-Driven Self-Tuning of RF Circuits for Process-Induced Performance Variability.” DSP-Driven Self-Tuning of RF Circuits for Process-Induced Performance Variability.

[16] “Enable Embedded Mode.” *Microsoft Docs*, 4 October 2021, https://docs.microsoft.com/en-us/windows/iot/iot-enterprise/os-features/embedded-mode. Accessed 26 April 2022.

[17] “433MHz RF Bandpass Filter Module PCB Double Sided Board BPF Module Electronic.” *ebay*, https://www.ebay.com/itm/313811380030?hash=item49109da73e:g:LmQAAOSw8IFhy~5v.

[18] “Geekstory BN-880 GPS Module.” https://www.amazon.com/Geekstory-Navigation-Raspberry-Aircraft-Controller/dp/B078Y6323W/.

[19] “Geekstory BN-880 GPS Module U8 with Flash HMC5883L Compass + GPS Active Antenna Support GPS Glonass Beidou Car Navigation for Arduino Raspberry Pi Aircraft Pixhawk APM Flight Controller.” *Amazon.com*, https://www.amazon.com/Geekstory-Navigation-Raspberry-Aircraft-Controller/dp/B078Y6323W. Accessed 26 April 2022.

[20] “GPS Module GPS NEO-6M.” https://www.amazon.com/Microcontroller-Compatible-Sensitivity-Navigation-Positioning/dp/B07P8YMVNT/.

[21] “Ham Radio History.” *ARRL*, http://www.arrl.org/ham-radio-history. Accessed 9 April 2022.

[22] “HDMI Audio Extractor.” *Amazon.com*, https://www.amazon.com/Extractor-Converter-Adapter-Splitter-Compatable/dp/B084RN22MW/. Accessed 9 April 2022.

[23] “HDSDR - Screenshots.” *hdsdr*, http://www.hdsdr.de/screenshots.html. Accessed 9 April 2022.

[24] “HiLetgo 0.1-2000MHz RF WideBand Amplifier 30dB High Gain Low Noise LNA Amplifier.” *Amazon.com*, https://www.amazon.com/HiLetgo-0-1-2000MHz-WideBand-Amplifier-Noise/dp/B01N2NJSGV/. Accessed 9 April 2022.

[25] “HONKYOB USB Mini Speaker.” *Amazon.com*, https://www.amazon.com/HONKYOB-Speaker-Computer-Multimedia-Notebook/dp/B075M7FHM1/. Accessed 9 April 2022.

[26] Infineon. “OPTIREG™ switcher TLS4125D0EPV50.” *Infineon Technologies*, 5 February 2021, https://www.infineon.com/dgdl/Infineon-TLS4125D0EP%20V50-DataSheet-v01_00-EN.pdf?fileId=5546d46270c4f93e01710b608b523b33. Accessed 25 April 2022.

[27] “The Invention of Radio Technology.” *ThoughtCo*, 10 May 2019, https://www.thoughtco.com/invention-of-radio-1992382. Accessed 9 April 2022.

[28] “JniTyOpt 3.5 inch Display.” https://www.amazon.com/JniTyOpt-Raspberry-Interface-Resolution-Supports/dp/B0931Z7MX5/.

[29] “KISEER USB 2.0 Mini Microphone.” https://www.amazon.com/KISEER-Microphone-Desktop-Recording-YouTube/dp/B071WH7FC6/.

[30] “LimeSDR vs LimeSDR Mini.” *Lime Microsystems*, https://limemicro.com/products/boards/limesdr-mini/. Accessed 26 April 2022.

[31] “LoveRPi 4 Port MicroUSB to USB Hub.” *Amazon.com*, https://www.amazon.com/LoveRPi-MicroUSB-Port-Black-Raspberry/dp/B01HYJLZH6/. Accessed 9 April 2022.

[32] “Maxmoral SMA Male to Female Connector RF Coaxial Adapter.” *Amazon.com*, https://www.amazon.com/Maxmoral-Female-Connector-Coaxial-Adapter/dp/B07239N158. Accessed 9 April 2022.

[33] “MIC Amplifier AFT09MS007NT1 NXP USA.” *Digikey*, https://www.digikey.com/en/products/detail/nxp-usa-inc/AFT09MS007NT1/4461952. Accessed 9 April 2022.

[34] “MIC Amplifier NPA1003QA.” *MACOM*, https://www.macom.com/products/product-detail/NPA1003QA. Accessed 9 April 2022.

[35] “MIC Amplifier TAT7472A1F.” *Qorvo*, https://www.qorvo.com/products/p/TAT7472A1F#parameters. Accessed 9 April 2022.

[36] “Mini HDMI Adapter.” https://www.amazon.com/ULT-WIIQ-Standard-Aluminum-Raspberry-Camcorder/dp/B0924KF735/.

[37] “Nagoya NA-320A Triband HT Antenna 2M-1.25M-70CM (144-220-440Mhz) Antenna SMA-Female for BTECH and BaoFeng Radios.” *Amazon.com*, https://www.amazon.com/Nagoya-NA-320A-2M-1-25M-70CM-144-220-440Mhz-BTECH/dp/B01K10B9XK/. Accessed 9 April 2022.

[38] “NASA Technical Standard: Soldered Electrical Connections.” *The NASA Electronic Parts and Packaging Program*, 18 January 2001, https://nepp.nasa.gov/docuploads/06AA01BA-FC7E-4094-AE829CE371A7B05D/NASA-STD-8739.3.pdf. Accessed 9 April 2022.

[39] “NEO-6 series | u-blox.” *U-blox*, https://www.u-blox.com/en/product/neo-6-series. Accessed 26 April 2022.

[40] “Omega2 – Onion.” *Onion.io*, https://onion.io/omega2/. Accessed 26 April 2022.

[41] “Operating system images – Raspberry Pi.” *RaspberryPi.com*, https://www.raspberrypi.com/software/operating-systems/. Accessed 26 April 2022.

[42]“Pimoroni Speaker pHAT.” *Adafruit Industries*, https://www.adafruit.com/product/3401. Accessed 9 April 2022.

[43] “Raspberry PI OS Lite vs Desktop: comparison between the 2 distributions.” *peppe8o*, https://peppe8o.com/raspberry-pi-os-lite-vs-desktop/. Accessed 26 April 2022.

[44] “Raspberry Pi Zero 2 W – Raspberry Pi.” *RaspberryPi.com*, https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/. Accessed 26 April 2022.

[45] “RF Amplifier Ultra Wideband Gain 20dB Medium Power Amplifier Board 5M-6GHz.” https://ebay.to/3CLVugK.

[46] “RF Power Amplifier 20M-512MHz 5W Linear Amplifier for FM Radio Remote Interphone.” *eBay*, https://www.ebay.com/itm/174362238519. Accessed 9 April 2022.

[47] “RS-UVPA 5W UHF/VHF Power Amp.” *HobbyPCB*, https://www.hobbypcb.com/index.php/products/uhf-vhf-radio/rs-uvpa. Accessed 9 April 2022.

[48] “SDRSharp | Amateur Radio – PEØSAT.” *Amateur Radio – PEØSAT*, https://www.pe0sat.vgnet.nl/sdr/sdr-software/sdrsharp/. Accessed 9 April 2022.

[49]“6 Types of Anti Patterns to Avoid in Software Development.” *GeeksforGeeks*, 15 January 2021, https://www.geeksforgeeks.org/6-types-of-anti-patterns-to-avoid-in-software-development/. Accessed 10 April 2022.

[50]“Tangxi Mini Stereo Speaker.” https://www.amazon.com/Tangxi-Stereo-Speaker-Pillow-Player/dp/B07SGK8T8W/.

[51]Texas Instruments. “LM2621Low Input Voltage,Step-UpDC-DC Converter.” November 2015,

https://rocelec.widen.net/view/pdf/y3iuum7fts/snvs033d.pdf?t.download=true&u=5oefqw. Accessed 10th April 2022.

[52]Texas Instruments. “TPS63070 2-V to 16-V Buck-Boost Converter With 3.6-A Switch Current.” *ti.com*, Texas Instruments, March 2019, https://www.ti.com/lit/ds/symlink/tps63070.pdf?ts=1650925377945&ref_url=https%253A%252F%252Fwww.google.com%252F. Accessed 21st April 2022.

[53]“13 Raspberry Pi Zero Alternatives.” *It's FOSS*, 30 October 2020, https://itsfoss.com/raspberry-pi-zero-alternatives/. Accessed 10 April 2022.

[54]“3D Printing with Polypropylene.” *Simplify3D*, https://www.simplify3d.com/support/materials-guide/polypropylene/. Accessed 10 April 2022.

[55]“Tips for 3D Printing with PETG.” *Simplify3D*, https://www.simplify3d.com/support/materials-guide/petg/. Accessed 10 April 2022.

[56]“Tips for 3D Printing with PLA.” *Simplify3D*, https://www.simplify3d.com/support/materials-guide/pla/. Accessed 10 April 2022.

[57]“UL1642 Certification of Lithium-ion Battery.” *LiPol Battery's*, https://www.lipolbattery.com/UL1642_Certification_of_Lithium-ion_Battery.html. Accessed 9 April 2022.

[58]“Ultimaker Cura: Powerful, easy-to-use 3D printing software.” *Ultimaker*, https://ultimaker.com/software/ultimaker-cura. Accessed 25 April 2022.

[59]“Ultimate 3D Printing Materials Guide.” *Simplify3D*, https://www.simplify3d.com/support/materials-guide/. Accessed 10 April 2022.

[60]“Used Lithium-Ion Batteries | US EPA.” *US Environmental Protection Agency*, https://www.epa.gov/recycle/used-lithium-ion-batteries. Accessed 9 April 2022.

[61]“Vega Barebones - Ultra Low-Noise VGA Module for RF & SDR.” https://www.amazon.com/Vega-Barebones-Low-Noise-30MHz-4000MHz-Capability/dp/B08LNYKHSM/.

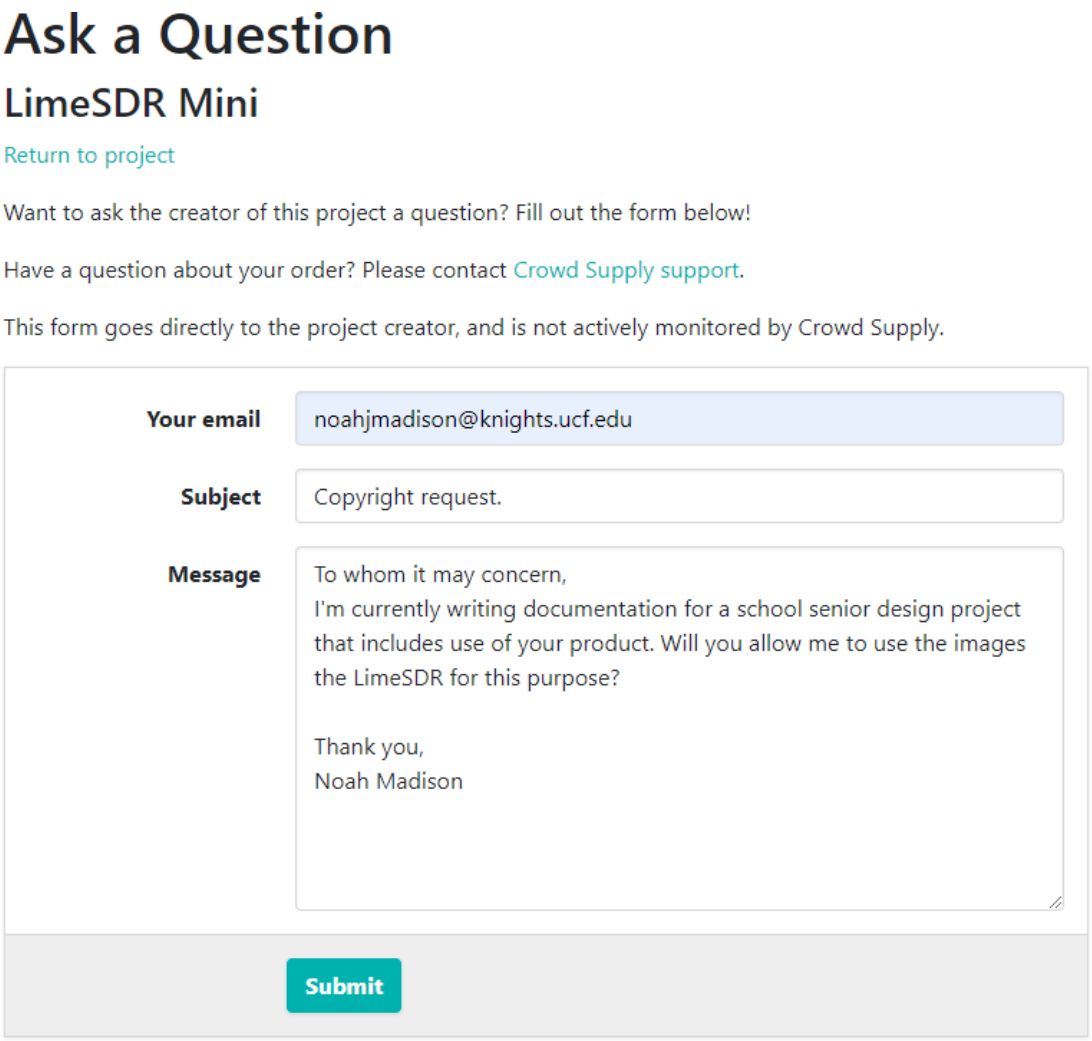
[62]“VK2828U7G5LF GPS Module.” *Cytron.io*, https://www.cytron.io/p-vk2828u7g5lf-gps-module. Accessed 26 April 2022.

[63]“Walkie Talkie Antenna for Baofeng UV-5X3 Antenna Upgrade Triband HT Antenna 144-220-440Mhz Antenna SMA-F.” *Amazon.com*, https://www.amazon.com/2M-1-25M-70CM-144-220-440Mhz-SMA-Female-Compatie-LUITON/dp/B0793JRVS7/. Accessed 9 April 2022.

Copyright

The following is a list of all relevant permissional requests, licensure, and copyright needed for the completion of this project.

Copyright Permissions



Ask a Question

LimeSDR Mini

Return to project

Want to ask the creator of this project a question? Fill out the form below!

Have a question about your order? Please contact Crowd Supply support.

This form goes directly to the project creator, and is not actively monitored by Crowd Supply.

Your email noahjmadison@knights.ucf.edu

Subject Copyright request.

Message

To whom it may concern,
I'm currently writing documentation for a school senior design project that includes use of your product. Will you allow me to use the images the LimeSDR for this purpose?

Thank you,
Noah Madison

Submit

Figure 87. LimeSDR Mini Permission

Ask a Question

CaribouLite RPi HAT

Return to project

Want to ask the creator of this project a question? Fill out the form below!

Have a question about your order? Please contact Crowd Supply support.

This form goes directly to the project creator, and is not actively monitored by Crowd Supply.

Your email noahjmadison@knights.ucf.edu

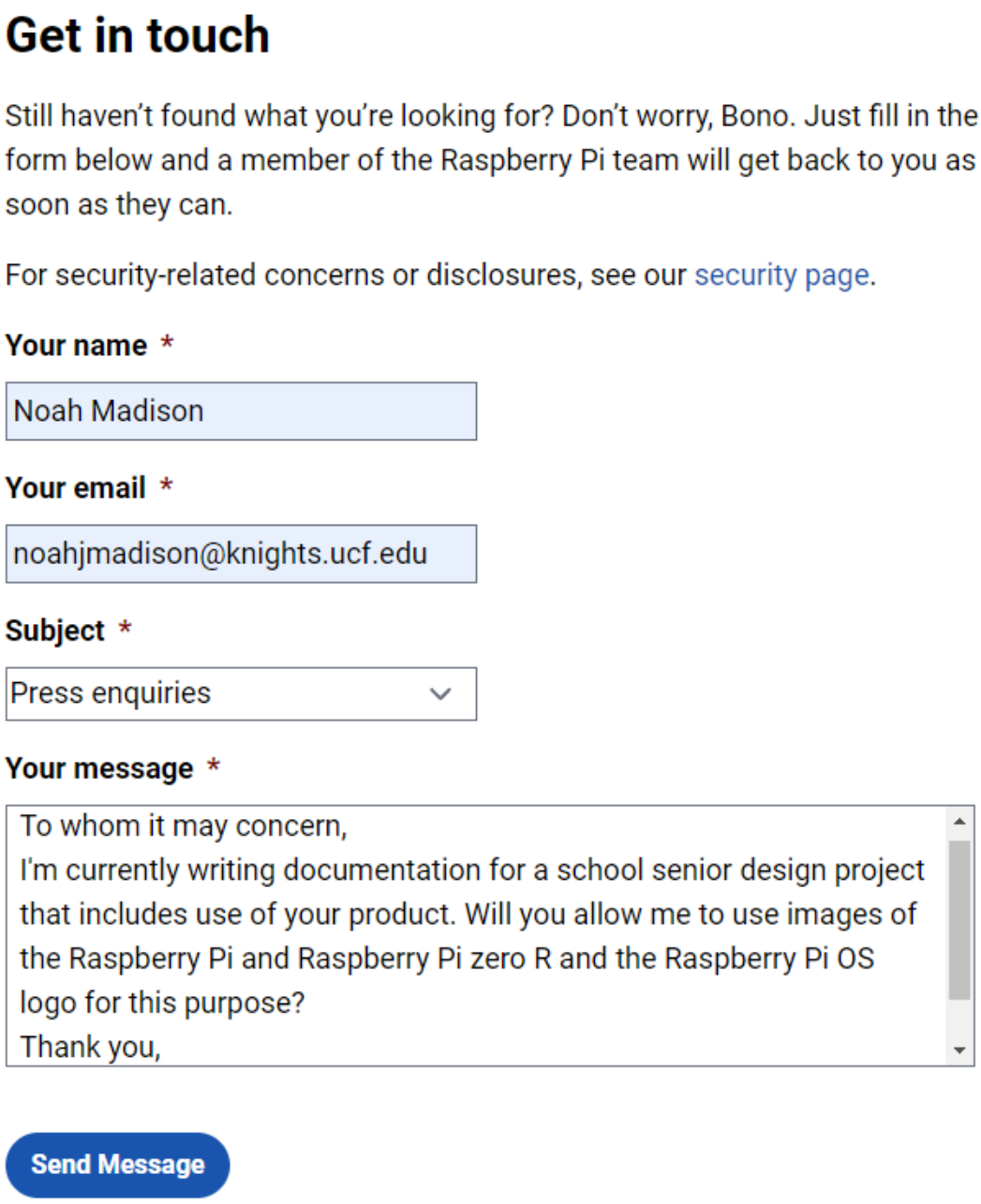
Subject Copyright request.

Message

To whom it may concern,
I'm currently writing documentation for a school senior design project that includes use of your product. Will you allow me to use the images the CaribouLite SDR for this purpose?
Thank you,
Noah Madison

Submit

Figure 88. CaribouLite RPi Hat permissions



Get in touch

Still haven't found what you're looking for? Don't worry, Bono. Just fill in the form below and a member of the Raspberry Pi team will get back to you as soon as they can.

For security-related concerns or disclosures, see our security page.

Your name *

Noah Madison

Your email *

noahjmadison@knights.ucf.edu

Subject *

Press enquiries

Your message *

To whom it may concern,
I'm currently writing documentation for a school senior design project that includes use of your product. Will you allow me to use images of the Raspberry Pi and Raspberry Pi zero R and the Raspberry Pi OS logo for this purpose?
Thank you,

Send Message

Figure 89. Raspberry Pi Permissions

The Banana M2 Images have an image release and an open source license:

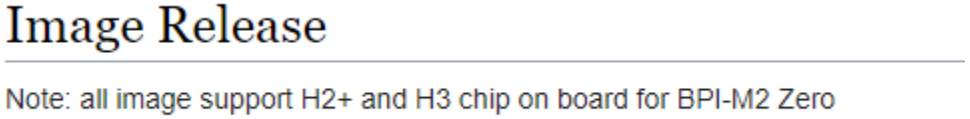


Image Release

Note: all image support H2+ and H3 chip on board for BPI-M2 Zero

Figure 90. Banana M2 Permissions

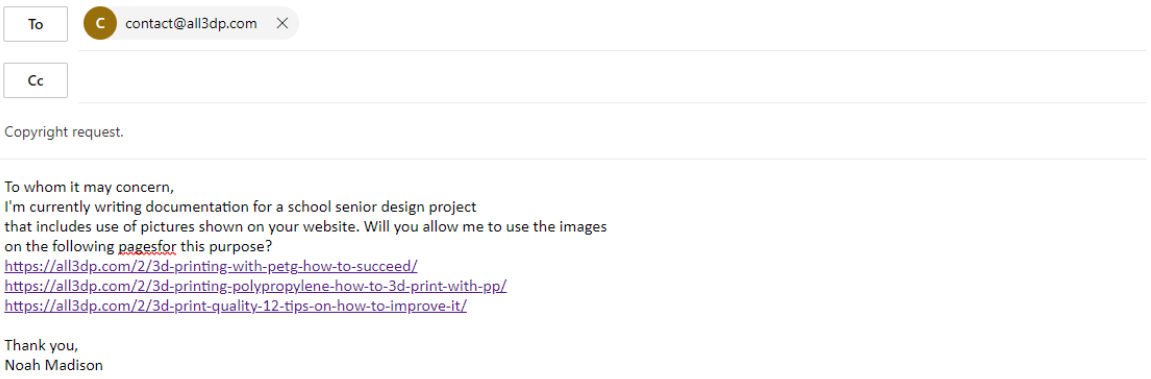


Figure 91. 3D Printing Permissions

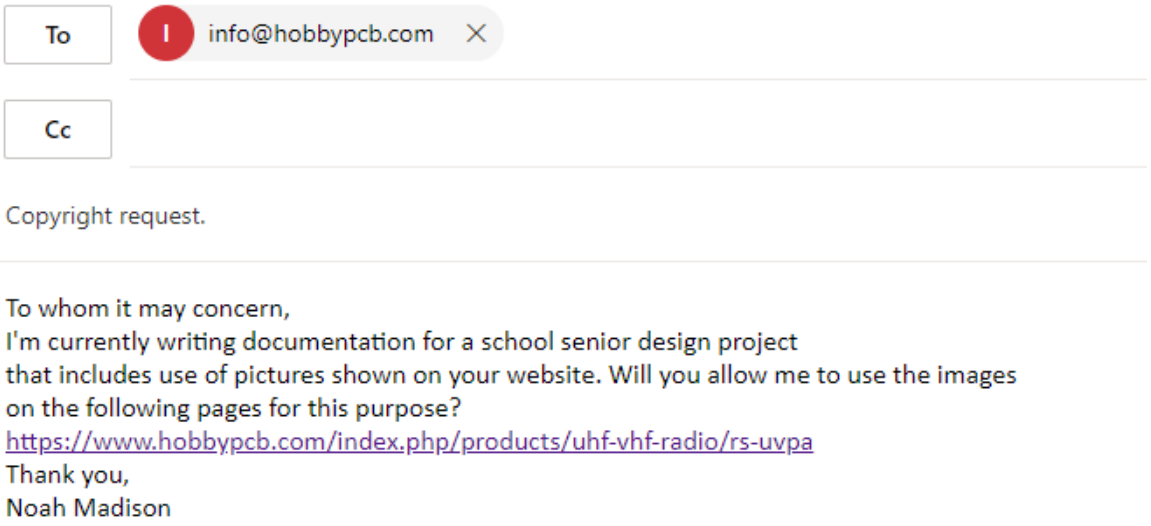


Figure 92. Radio Permissions

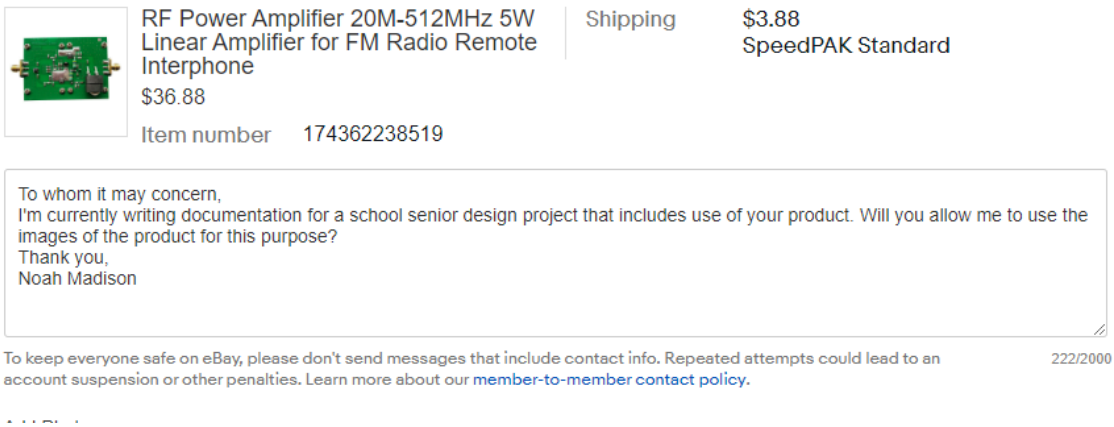


Figure 93. Amplifier Permissions

CONTACT QORVO

*Required Fields

General Information

Noah	Madison
noahjmadison@knights.ucf.edu	UCF
United States of America	State
orlando	32826
4077259833	

To whom it may concern,
I'm currently writing documentation for a school senior design project that includes use of your product. Will you allow me to use the images of the product linked here: https://www.qorvo.com/products/p/TAT7472A1F
Thank you,
Noah Madison

Figure 94. Qorvo Permissions

Have a Product Inquiry?

Contact Information

First Name *	Country *
Noah	United States
Last Name *	City
Madison	orlando
Company *	State *
UCF	Florida
Email *	Zip Code *
noahjmadison@knights.ucf.edu	32826
	Phone *
	4077259833

Other Information

Product Category *

GaN Power Amplifiers > 5W

Product Number *

NPA1003QA

Requested Information *

the NPA1003QA for this purpose?
Thank you,
Noah Madison

Figure 95. NPA Permissions

The following general format was used for the amazon images:

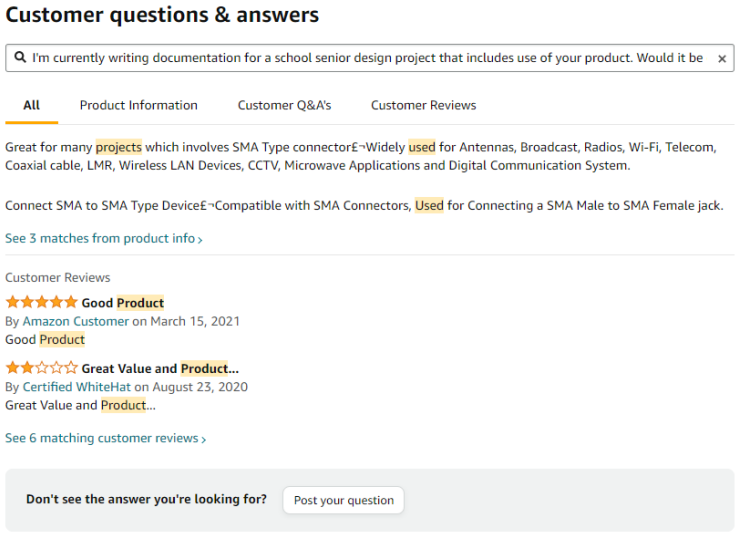


Figure 96. Amazon Permissions

Python

The python programming language is a completely free open source development tool that has no restrictions on private or commercial use. All of the modules and add-ons that we have selected as of right now are all either part of the GNU License, MIT License, Apache 2.0 License, or are simply just open source and free to use to the public and companies.

Raspberry Pi

Raspberry Pi embedded systems utilize a series of licensing and copyrights including the Creative Commons Attribution-ShareAlike 4.0/3.0 International, Creative Commons Attribution-NoDerivatives 4.0 International, BSD 3-Clause, MIT License, and General Public License. Under these documents, we are able to copy, share, and distribute certain files as wella s create devices as we see fit.

LimeSDR

The LimeSDR suite of tools is open source under the Apache 2.0 License and can support a wide range of communication standards and frequencies. The Apache License ensures that we can develop drivers and use the LimeSDR suite of tools without running into the issue of patent infringement. Our only restriction from the Apache License that affects us is that we are not allowed to use the