INEXPENSIVE UHF TRANSCEIVER LEVERAGING COTS COMPONENTS

Owen Chiaventone, Kyle Avola, Stetson Tuschhoff
Electrical and Computer Engineering Department
Missouri University of Science and Technology
Rolla, MO 65401
Faculty Advisor:
Dr. Kurt Kosbar

ABSTRACT

This paper describes the design of an inexpensive UHF transceiver which leverages some of the recently developed commercial off-the-shelf (COTS) components. The initial goal is to implement digital voice transmit and receive function, although the design can accommodate a wide range of digital communication and telemetry applications. The handheld transceiver transmits 5 watts of power in the 430-435 MHz UHF band. A 1.2 kHz wide GFSK modulation format is used, generated by a Silicon Labs radio chip. The recently released Raspberry Pi Zero processor implements a low bit rate audio coding which conforms to the Codec2 standard. The transceiver fits in a 3 cm x 8 cm x 14 cm volume. It is powered by two 18650 lithium ion cells, and draws approximately 1 watt of power during receive, and 6 watts during transmission.

Key Words

UHF GFSK Codec2 DSP

INTRODUCTION

This paper describes a cost effective digital handheld two-way radio suitable for a broad range of applications. FM and AM two-way handheld radios have been around for a very long time, have batteries capable of handling a day of continuous use, and a real world range exceeding 30km on 5 watts of power. A digital mode UHF/VHF radio would have several advantages over an analog radio, but its primary benefit is signal quality. With a quality codec and implementation, the digital radio has a greater range (A real world example of greater than 45 km for the same power) for the power input. Instead of a fading out effect as the user goes beyond range, the digital radios exhibit consistent voice quality until the connection is dropped, giving the user time to finish their statements as they go out of range. This radio aims to provide a ready-to-use radio or a software-modifiable platform for modifying the unit however the end user requires.

To make this a viable project, the radio must include a microprocessor with sufficient processing power to decode the voice data stream and a radio of sufficient sensitivity to decode very weak signals, such as those -100dBm and below. These two requirements must then be weighed against the ever present restriction of the battery life able to be given to a hand held project, the ability to fit easily in one hand. The radio must have a volume no more than 3x8x14cm, and should not weigh much more than a typical cellular or current hand held device, also imposing restrictions on the battery. The radio must be able to decode weak signals as interruption in decoding capability will cause the cessation of communication, and should be sufficiently powerful to provide communications over the specified range. The microprocessor should be able to handle the continual conversion of the microphone data stream using the DSP-intensive codec.

Our design does not rely on a proprietary codec or encoding chip, focusing on using off-the-shelf technology and an open hardware and software design. Our radio focuses on an open source codec which is processed on the radio's processor, saving the cost and extra power draw of an encoding chip. Looking to other examples of radios built using this codec, this radio will use prepackaged ICs whenever possible, saving the time, development cost, and board area common with a discrete approach.[4]. In contrast, our team wished to diminish development time and get a mature product to the market faster at the cost of some additional features that were likely to be unused outside of the experimental community. Finally, for the experimental community (or for those who require more control over their radio), our radio is still easy to work with in terms of source availability and hardware compatibility.

Our approach centers around a ready built radio chip with a built in microcontroller being controlled by a low power ($<$500mW) ARM processor capable of providing the digital signal processing. An off the shelf approach allows us to keep our time investment very low. Instead of using discrete components, we were able to use a single, low cost chip with a small amount of supporting circuitry to conserve board space and component cost, versus a digital synthesizer and oscillator chip approach. With a ready made radio front end, our team was able to focus on programming the radio and ensuring the signal chain was correct from the microprocessor to the radio. Because of the decision to incorporate the codec into a standard, well documented ARM chip, our team could simplify the board layout, as well as conserving the battery by using one less chip. In addition, this provided one more opportunity for meeting the cost savings goal, as a free codec with sufficient ability greatly reduces the cost of the end unit, with the codec being a major expense in the radio hardware. Focusing on off-the-shelf and open source technology allowed the team to greatly compact and dramatically lower the cost of the final product.

SPECIFICATIONS AND REQUIREMENTS

In order to be a practical, viable radio in the market, it was agreed upon that the unit must be able to conform to the following criterion: Power, Sensitivity, Runtime, Clarity, Cost, and Dimensions. First and foremost, in order to compete, the radio had to approach or exceed 5W of power on the UHF 430 MHz ISM band. While an advanced codec is able to decode at a lower signal receive power and a higher signal to noise ratio[5], 5W was set at the target output to maximize the range while maintaining runtime. The radio also had to be capable of receiving

signals fainter than existing products. The Si4464 chips used in the second and third prototypes are capable of decoding to -126dBm[3]. Runtime was a must have, and our target was for 120 minutes of continuous use (Assuming a 20% transmit usage cycle). In order to meet this goal, a pair of Lithium 18650 cells were used. With the power and range handled, the codec would be at its optimum performance, and we wished for clarity of speech to remain excellent to the very edge of its range. The open source Codec2 was selected for its excellent properties, including weak signal performance and low bandwidth constraints. [2] Our original target market, the local amateur radio community, was polled for the cost constraint. It was found that a sub-75 USD radio would be a very popular item, and every effort to remain below the 75 USD parts per unit price was made. Finally, the dimensions (both volume and weight) were in our minds as we made our prototypes, however the final prototype made to date does not yet follow these dimensions.

TECHNICAL APPROACH

SOFTWARE

Rowetel's Codec2 is at the heart of the radio design. The codec provides communications quality digital transmission at bitrates between 700 and 3200 bits per second. While not cellphone quality audio, a radio powered by Codec2 would allow for legible communications in many usage scenarios (such as construction, operations, or navigation) where users could reasonably expect noisy backgrounds and external interference to make perfect quality audio unnecessary. As communications requirements grow and frequencies become congested, efficient use of bandwidth is a requirement. The team selected a 1.2kHz channel with GFSK, providing more than the 700 bits per second requirement with room to fit forward error correction. A simulated channel was created with the ability to vary the noise floor, providing an opportunity to tune the codec before finishing the hardware (as seen in Figure 1).

PROCESSOR AND RADIO

The selection of the hardware was based around the codec and modulation scheme chosen to make sure no lock-in products were selected, leaving the decision as platform-agnostic as possible. The processor would handle the combination of the various user input sources (keypad, microphone, battery life remaining, USB charge/data) and output (Speakers, LCD Display, USB Data), as well as the radio I/O buffer. The processor would also encode and decode the microphone input stream, and the radio buffer input stream. The radio could then be any data chip with a byte I/O buffer.

USER I/O

To simplify the operation of the radio, a simple user interface was selected, composed of a frequency change mechanism, an on/off/volume potentiometer, and a USB charging system. The frequency change mechanism consists of a series of pushbuttons, a pair for up and down frequency select, and a pair for up and down band select. An LCD screen displays the users current frequency. The on/off rotary switch disconnects the battery from the regulation circuitry to save power, while

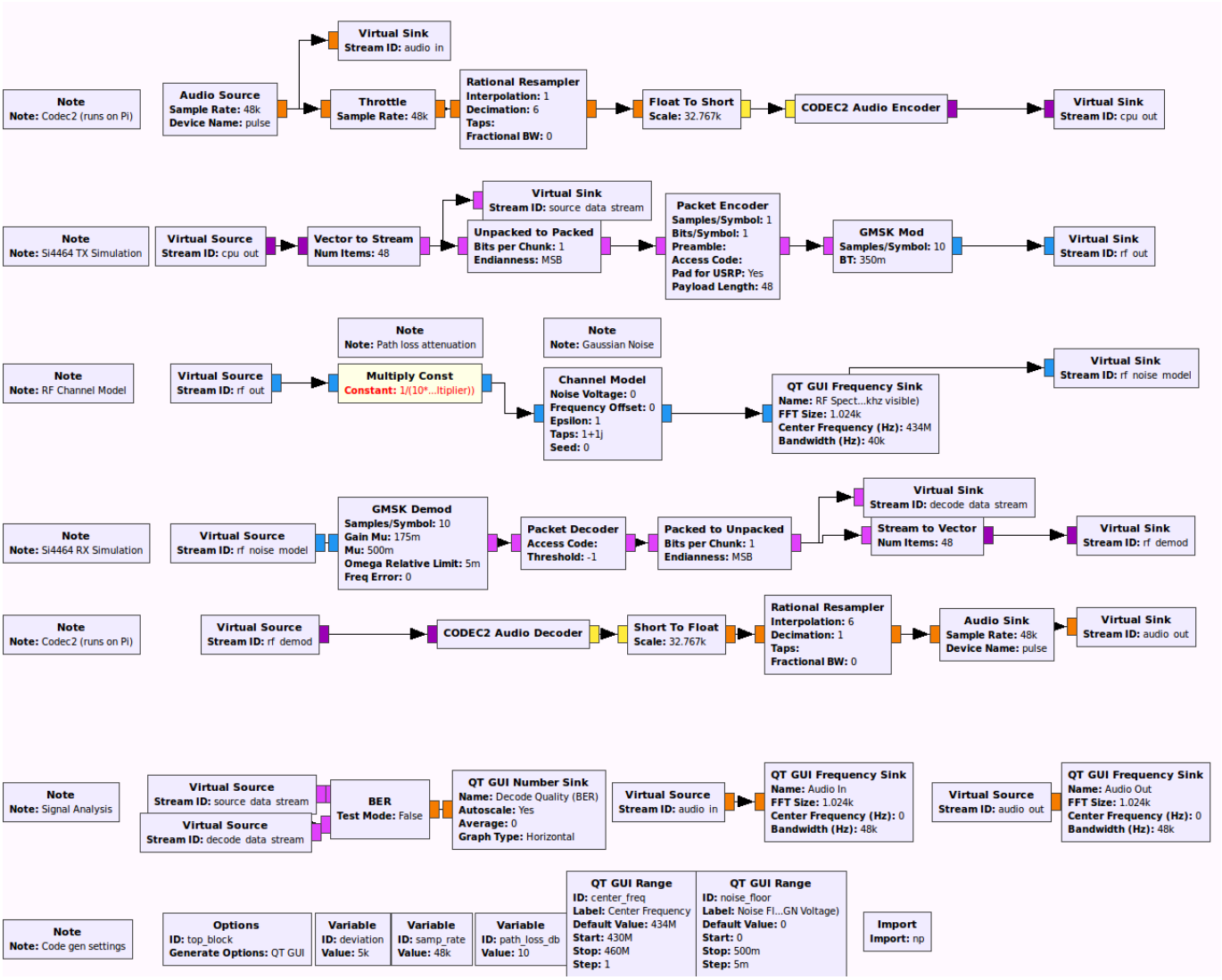

Figure 1: GNU Radio Flow Graph

the integrated volume potentiometer is connected to the processor via a 10 bit ADC. The USB port is capable of serial communication and fast charge for programming applications.

SUPPORTING CIRCUITRY

The battery charge and regulation system was selected to provide a constant 5V/3.3V source through a series of buck and boost converters. The USB port provides power to the charging buck converter to charge the battery, which is boosted from 3.2-4.2V to the running voltage of 5V, with a seperate 3.3V buck path.

FIRST REVISION AND REDESIGN

PROCESSOR, RADIO CHOICE

The initial task was the selection of the processor and radio. In order to get the signal to noise ratio required for long range communications, our choice fell to two modulation schemes: GFSK and QPSK. QPSK-16 was chosen due to its excellent performance in terms of power per bit ($\frac{E_b}{N_0}$), which left us with only one radio front end chip available at the price point and external interface complexity we were willing to expend. The On AX5043 offered excellent frequency range, very good sensitivity at a high data rate, and QPSK-16 support. An STM F4 series chip was original selected as our processor, providing low power draw and high performance. A development board was made for the radio, and a discover board was bought for the processor.

RESULTS

The results from this combination of hardware were not greatly encouraging in the first round. Two major issues prevented us from going further with this development track, time and complexity. Approximately 5 months was available to complete the project, leave a great short of time available for working through complex issues. The STM chip, while powerful, was a complex chip to work with, requiring either in-depth knowledge of the API and feature set exposed, or the time to develop low-level drivers for the chip, time our team did not have. The AX5043 shipped with a driver and configuration for the 8051, however, it had a difficult to work with feature set as well, and little if any community support for which to draw help. Eventually, the team was able to get a carrier from the combination, but the chip was seen as a lost cause to get the setup working within the time frame.

PITFALLS

We learned several lessons from this loss, but the most important being what to look for in selecting components. Instead of looking solely for a chip based on its technical capabilities, it was important to also look at support available, tools available, and overall time to complete a task for the given platform. If no tools were available, or were difficult to use, more time would be spent working with the toolset than developing for our application. If no support was available, the team would quickly find itself burning valuable time attempting to understand low level hardware than the issues which it set out to solve. If either of these grew to great, we would not be able to

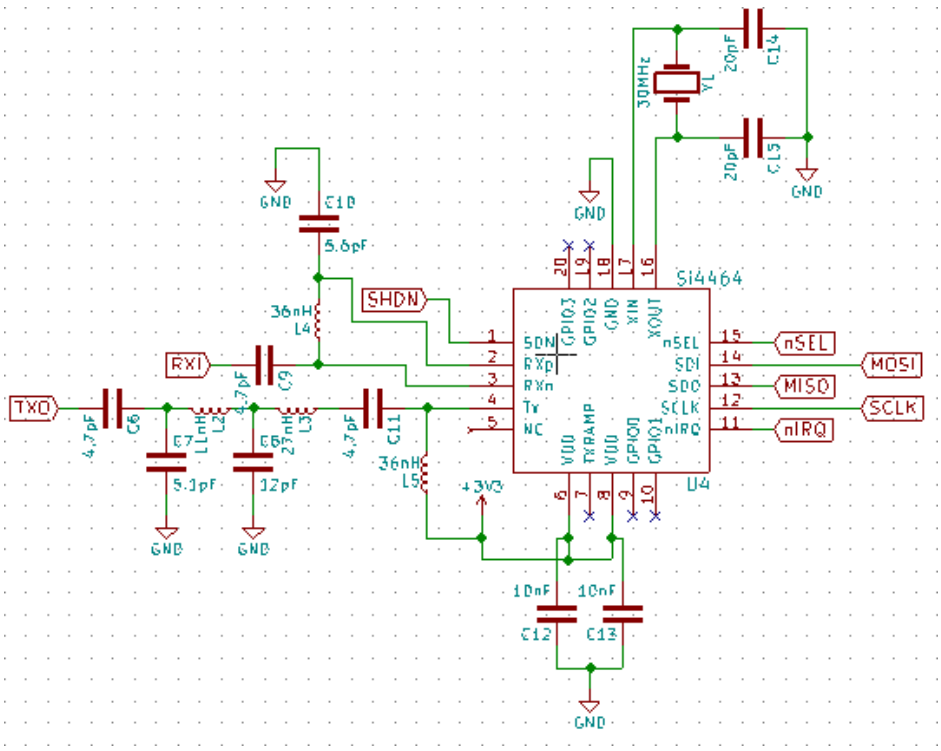

Figure 2: Radio and Matching Network

complete the project.

SECOND & THIRD REVISIONS

PROCESSOR, RADIO CHOICE

With the previous failure to make the radio and the processor communicate effectively, it was decided to simplify as much as possible to produce a working product. A Raspberry Pi was selected for its ease of GPIO control and heavy processing capability, which we hoped would allow for quick and easy turnaround on our software development. Our standby choice, an Si4464 radio, was selected to take over the radio front end. The Si4464 had excellent community support, prebuilt drivers for the raspberry Pi, and would be easy to build a radio platform for. Selecting the Pi, however, forced us to use a processor that was not power efficient for the task, and the Si4464 was not able to support QPSK-16. It was decided that these were worthwhile trade-offs to get a running prototype. In order to ”breadboard” the radio in a way suitable for use with the Raspberry Pi, a development board was built, which included the radio, supporting circuitry and matching networks, and the volume control knob (Figure 2). In revision three, a preamplifier, final amplifier, and RX/TX path select switch was added to the matching network front end (Figure 3).

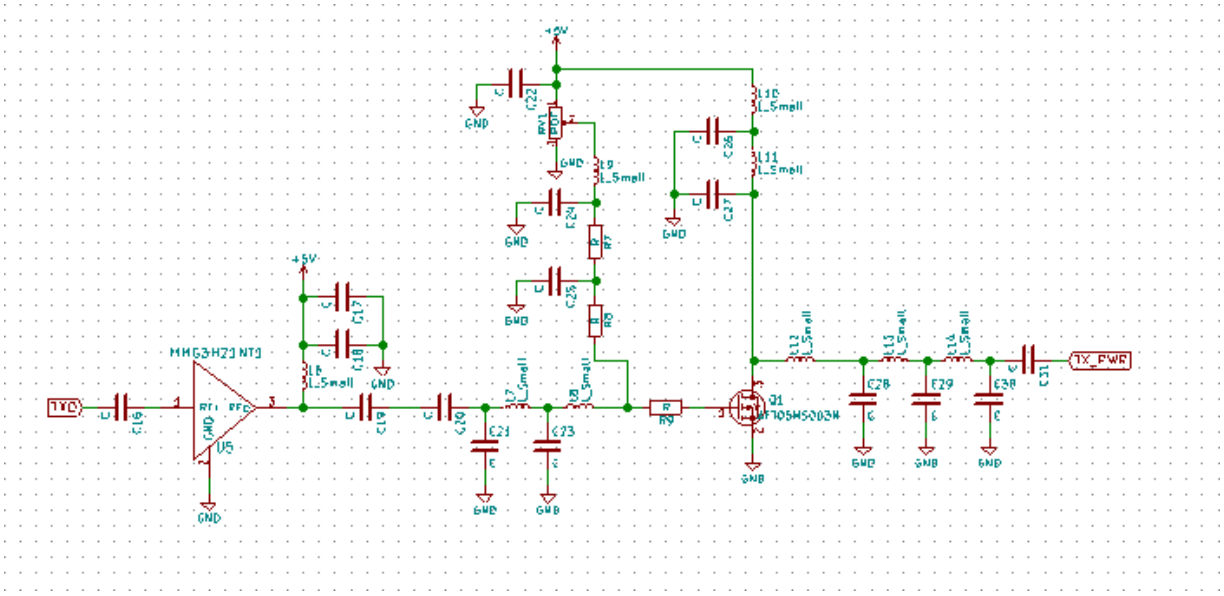

Figure 3: Amplifier Chain

BATTERY MANAGEMENT

The decision was made early on that the radio would be charged through a standard 5V USB micro B port. Such a charging method would enable quick, convenient charging with a charger that almost everyone would already have. To extend the capacity of the 18650 cells, the two 3.7V cells were placed in parallel. This allowed a standard buck converter to reduce the 5V input to a 4.2V charging voltage for the batteries (Figure 4). On the output side of the converter, a boost converter was used to supply 5V power to the rest of the system (An NPN MOSFET allowed for boost cutout when the USB charger was plugged in and the unit was operating at low load), followed by a buck converter to provide 3.3V to the rest of the system.

USER INTERFACE

The User Interface was comprised of a volume control knob, a series of pushbuttons for frequency selection, a Push-To-Talk switch and a LCD Screen. It was decided to keep the user interface simplistic to ensure the radio would not become too difficult to learn, a common problem with modern digital radios which include a great deal of features, filling never ending menus. Instead, volume was easily controlled, a frequency was able to be selected (and later, power output), and then one simply had to transmit.

RESULTS

The second and third prototype boards were greatly encouraging compared to the original choices. The user interface was complete, and a battery management system was demonstrated and capable of providing charging and output power to the system. While drivers were built for the Pi, a welcome improvement, they were for a previous version and had to be modified for the Pi

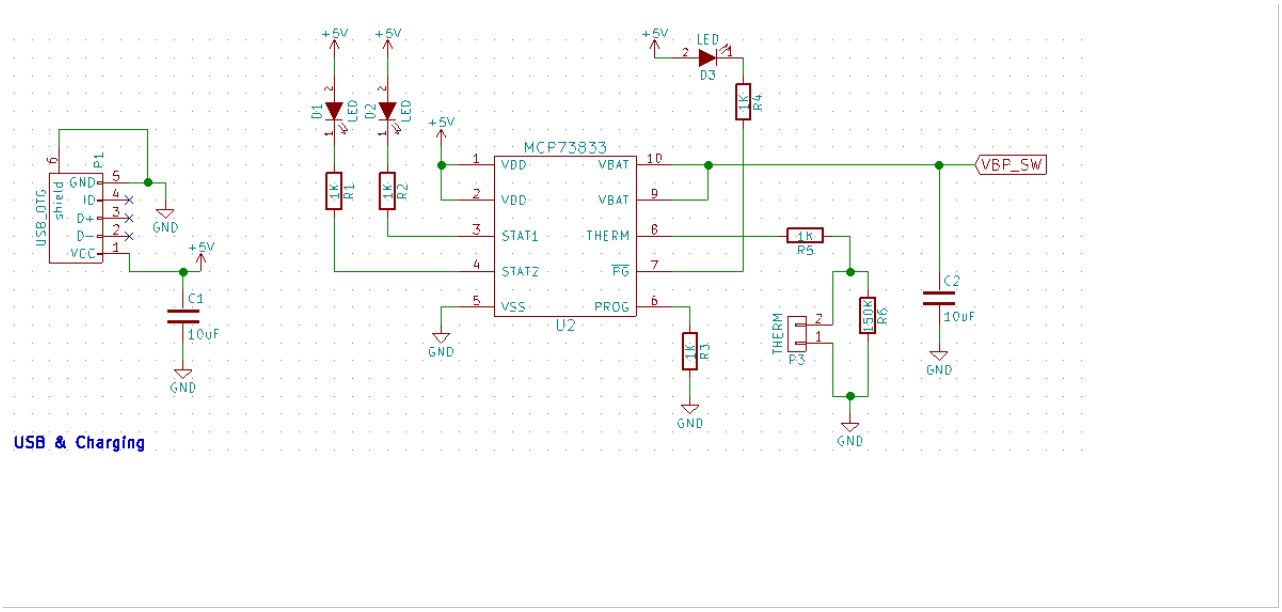

Figure 4: Battery Charging from USB

v3. Unfortunately, the Si4464 used an asynchronous serial select line, which is not handled by the Linux kernel. We were unable to get the Pi and the radio to talk with one another, requiring instead the use of an Arduino for testing purposes. As the Arduino was unable to handle codec processing of the microphone input, a demonstration was able to be made of a computer handling encoding of the data, passing the data to the Arduino, and then sending the data (received by an SDR).

FUTURE WORK

TRANSMISSION AND RECEPTION

As the team begins to wrap up its work, the next milestone is the successful transmission and reception of voice data packets. For this milestone to be met, a slight addition will need to be made to the radio to overcome the asynchronous serial issue met during testing of the third revision. Depending on microprocessor choice, the raspberry Pi processor will require an external processor which is capable of acting as a buffer and I/O controller for the Pi (Assuming we wish to continue with the Linux kernel), or a switch back to a dedicated low-power DSP capable ARM chip will have to be made. The benefit of the raspberry Pi processor is the vast array of drivers already included with the kernel, and the ease of accessing input and output for prototype use and changing. However, we eventually will have to migrate to a different microprocessor, and until then, the Linux kernel will act as an intermediary, slowing communications and requiring an IO processor to handle the radio. If the switch to a more complex chip were made at this time,

effort would be spent early on to port drivers and handle more of the underlying hardware instead of providing a quick prototyping environment, but we would be more knowledgeable of the end product when the time comes to switch to the new chip.

FULL INTEGRATION

Following a successful transmission and reception of voice data, the radio must be integrated to form a seamless experience. At this time, the prototype is bulky and over size requirements. To finalize the product, the various subunits of the product must be designed to work with one another and the unit must be compacted. For the subunits to be designed together, the power management must be added to the processing board, which should be able to snap on in a low-profile manner to the radio board (which has been moved off the processing chip for concerns of signal and noise issues). Following the resolution of those items, the unit will be a full-featured radio.

CONCLUSIONS

While it is clear that there is a great deal of work necessary to get this radio in a state suitable for use in emergency and commercial services, our team has built a solid base on which to work on this project in the future. With a method of simulating encoding and decoding, adding noise and processing times, the future of the project will not rely as greatly on any one bottleneck issue. With the user interface in place and integrated with the radio drivers and processing front end, future work will be easier and quicker to implement, allowing the team to spend more time on adding features and cleaning up issues as they arise. Future work will also involve a better management of resources, including money, time, and personnel. A digital voice mode radio using the Codec2 codec is not far away.

REFERENCES

[1] ”DVSI home page,” in DVSI, 2016. [Online]. Available: http://www.dvsinc.com/index.htm. Accessed: Dec. 13, 2016.

[2] D. Rowe, ”Codec 2,” in Rowetel, 2016. [Online]. Available: http://www.rowetel.com/?page_id=452. Accessed: Dec. 13, 2016.

[3] ”SI4464/63/61/60 Datasheet,” in Silicon Labs, 2016. [Online]. Available: https://www.silabs.com/Support%20Documents/TechnicalDocs/Si4464-63-61-60.pdf. Accessed: Dec. 13, 2016.

[4] david rowe, ”SM2000 Part 1 – Open VHF digital voice radio,” in Rowetel, 2015. [Online]. Available: http://www.rowetel.com/?p=4694. Accessed: Dec. 13, 2016.

[5] Stallings, Data and Computer Communications, 10th edition, pgs 98-103

[6] RadioHead: RadioHead Packet Radio library for embedded microprocessors, RadioHead: RadioHead Packet Radio library for embedded microprocessors. [Online]. Available: http://www.airspayce.com/mikem/arduino/RadioHead/. [Accessed: 02-Mar-2017].