AM Audio Transmission to Demonstrate Signal Attenuation Characteristics

Trevor Wolf, Colorado School of Mines

***Abstract*—Signal transmission and signal losses can be a hard concept to grasp. Demos can help to increase class comprehension of course content. An in class demo using real antennas and signals can help demonstrate the effects of antenna tuning, orientation, polarization, and gain in an easy to interpret format.**

I. Introduction

LECTURE demonstrations should be designed in such a way that students can easily grasp. RF signal attenuation is not an easy concept to grasp, as it involves complex relationships between antenna designs, their positioning, their rotation, signal strength, and more. This project outlines a demonstration for a class lecture for an introduction to antennas course. This system is to demonstrate the effects of the aforementioned variables that effect signal transmission in an easy to grasp way. This demonstration is to incorporate two elements: a transmitter and a receiver. The transmitter sends a signal that is made known to the students. The receiver picks up and displays this signal so it can be juxtaposed with the original.

A. Design Requirements

- Demonstration system must incorporate a transmitting and receiving end with SMA connectors to swap out antennas as needed.
- Transmitting end must be mobile enough to demonstrate a noticeable loss in signal strength due to distance.
- Receiving end must be able to display to an HDMI projector screen so the class can see the received signal waterfall.
- A signal encoding method should be chosen such that the apparent quality of the signal received must drop proportionally to the drop in signal strength.
- Must be easy to set up within a few minutes.
- Must be able to change carrier frequencies quickly and easily to support differently tuned antennas.

II. Design

A. Transmitter Module

The final design topology is laid out visually in Fig. 1. The mobile transmitting unit consists of a Raspberry Pi with a touchscreen display, a HackRF software defined radio, and a transmitting antenna. The Raspberry pi is running a python script built with the GNU Radio software. This script takes an audio file and encodes it [1] with amplitude modulation at a specified carrier frequency.

This paper is part of a research project for the EENG386 course
Course instructed by Atef Elsherbeni at the Colorado School of Mines
Manuscript received December 5, 2022

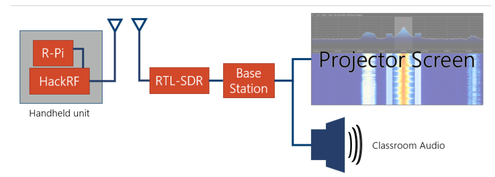

Fig. 1. Block Diagram of demonstration elements.

B. Receiver Module

The receiving end consists of an RTL-SDR chosen for its small size, low cost, and absence of a need to transmit. This is connected to a laptop which is running SDR# or GQRX (decoding software, see A.2 and A.3) depending on which is more convenient for the operating system of the chosen laptop. SDR# should be used with windows devices and GQRX is preferable for Mac and Linux devices. The decoding software shows the signal spectrum and waterfall, and decodes the AM audio to be played over the classroom speakers. Ideally the laptop being used as the base station will have an HDMI connection to display the spectrum and waterfall on the classroom projector screen for easy viewing by the entire class.

C. Software Use

Once the python script is run, a GUI will appear and the audio file will begin playback at the default carrier frequency of 433MHz. The graphical user interface of this script displays the transmitted signal waterfall, allows frequency selection with a series of radio buttons, and allows adjustment of the amplitude of the audio signal relative to the carrier wave within the modulation. This last adjustment gives a pseudo volume control.

D. Hardware Configurability

The hardware allows for antennas to be swapped out at will, as long as they use SMA connectors. A 5V amplifier can be added or removed between the HackRF and the transmitting antenna to increase signal strength.

III. Software

The python script run on the Raspberry Pi was generated with GNU Radio using GNU Radio's flowgraph companion.

This flowgraph contains a set of blocks that route samples from source to sink. In this case the source is a ”WAV File Source” block [1] and the sink is an ”osmocom sink” block. The samples pass through a repeat block to increase the sample rate from the audio sample rate to a rate suitable for the HackRF, as well as a multiply block to adjust the relative amplitude as determined by the UI ”volume” slider. The samples are then offset by 1 to avoid negative amplitudes and converted to complex samples [2] with a null complex component. This complex sample stream is multiplied with a signal source block [3] of the cosine form, oscillating at the set carrier frequency. This signal is then passed to a GUI sink block and the osmocom sink. Additionally, the osmocom sink can be disabled and a set of blocks enabled that adds noise before decoding the signal again and passing it to an audio sink [4]. This allows software testing without any attached hardware.

IV. Testing

Testing started with an unmodulated carrier wave to validate the data path between GNU Radio, the HackRF, and the receiving station. The flowgraph was then adapted to slowly change the carrier frequency by modulating it with a low frequency (0.1Hz) wave to very that the amplitude modulation was functioning correctly. The audio component was then introduced and after various tweaks, the transmitted music was audible on the receiving end. In Fig. A.1 you can see the amplitude modulated audio signal being transmitted at 433MHz. Fig. A.2 shows the audio being received at 915MHz. Note the considerable reflections in the spectrum at regular intervals above the carrier frequency. These appeared at higher frequencies when the 5V amplifier was installed. The decoded audio could also be heard at these frequencies, while with less clarity. Testing showed a distinct drop in audio quality when actions that would be expected to drop the signal quality were performed. This includes rotating linearly polarized antennas out of phase, increasing the distance between antennas, and changing the transmitter power by adding or removing the amplifier.

V. Conclusion

Concluding testing, the demo is complete and ready for a class presentation. Testing showed that the drop in audio quality was a good indicator of the drop in signal strength, which also lines up with a visual drop in signal strength on the receiver’s spectrum readout. The transmitting module is not completely untethered at the moment, it is tied to an outlet by the power adapter of the Raspberry Pi display. This can be easily remedied with a battery pack and a USB-A to USB-C cable. This may not be necessary as the transmitting module is composed of many parts and is not convenient to hold and may be better suited to a semi-mobile cart anyway. The cables from the SDR to the antenna still allow it plenty of mobility to demonstrate signal fallof via distance and orientation.

Appendix A
Images From Testing

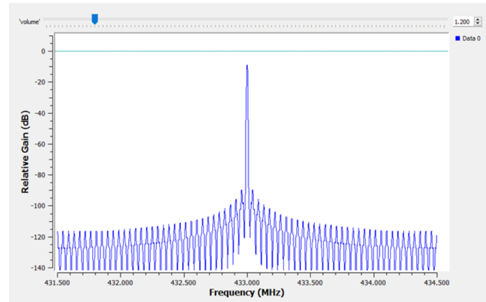

Fig. A.1. Spectrum from transmitter showing encoded signal at 433MHz.

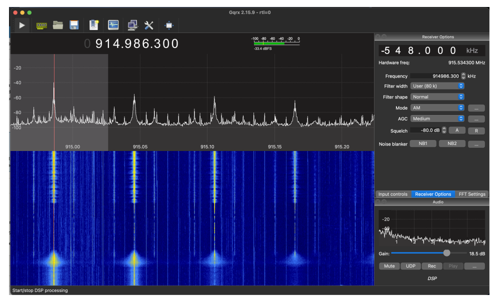

Fig. A.2. Signal Reception at 915MHz using an RTL-SDR and GQRX (Kobe Prior)

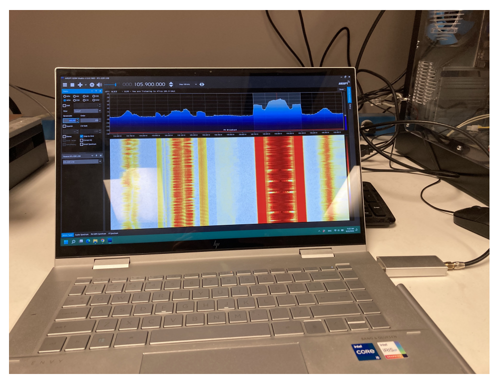

Fig. A.3. FM Signal Reception using SDR# on a Windows Laptop with an RTL-SDR

Acknowledgment

The author would like to thank Kobe Prior for providing assistance with testing the transmitter and receiver simultaneously, as well as Robert Jones and Matthew Dodd for their assistance in debugging GNURadio flowgraphs.

References

[1] GNU Radio Wiki Editors, “Simulation example: AM transmitter and receiver - GNU Radio,” Mar. 2021. [Online]. Available: https://wiki.gnuradio.org/index.php?title=Simulation_example:_AM_transmitter_and_rec

[2] M. Steele, “Building an AM Transmitter with HackRF One,” May 2020. [Online]. Available: https://www.dlineradio.co.uk/articles/building-an-am-transmitter-with-hackrf-one/

[3] GNU Radio Wiki Editors, “Signal Source - GNU Radio,” Aug. 2022. [Online]. Available: https://wiki.gnuradio.org/index.php/Signal_Source
[4] S. Katz, “Using GNU Radio Companion: Tutorial 3. Receiving AM Signals,” publisher: California State University, Northridge. [Online]. Available: http://www.csun.edu/ skatz/katzpage/sdr_project/sdr/grc_tutorial3.pdf

Trevor Wolf was born in Louisville, CO. He is a student at the Colorado School of Mines pursuing a bachelors degree in electrical engineering. He is a member of the ARC Research Group at the Colorado School of Mines. He has interned in the research and development department of Bridge Diagnostics Inc. for two years. Research interests include software defined radio, satellite communications, microcontrollers, and sensor systems.