Data Transmission over Audible Spectrums with OOK

Miles Macchiaroli, Spencer Bourassa, Adam Beauchaine, Subin Bastola, and Mira Yun
Department of Computer Science and Networking
Wentworth Institute of Technology
Boston, MA 02115, USA
{macchiarolim, bourassas, beauchainea, bastolas, yunm}@wit.edu

***Abstract*—By utilizing on-off keying (OOK) modulation, we build a data transmission system in which two nodes are able to communicate via an audible AM radio band. In this paper, we focus primarily on both the implementation of an OOK system and an analysis of its merits in the modern world of wireless networking technology, which are largely limited to simple specialized tasks in requiring high spectral efficiency.**

Keywords*—*Wireless Communication System; On-Off Keying; Amplitude Modulation; Hands-on Example;

I. INTRODUCTION

In the current state of network engineering, the use of efficient and reliable wireless communication standards is paramount to the success of any modern organization. Because of the growing demand for wireless professionals in all network engineering fields, many schools introduce wireless networks and applications to undergraduates. Wireless communication, like other networking courses, can be too dry and abstract for undergraduates without hands-on learning experience [1]. Thus, off-the-shelf devices, open-source software, and third-party firmware including wireless routers, Raspberry Pi, Arduino, Kismet [2], DD-WRT [3], BATMAN-adv [4], and so on, have been introduced into undergraduate classrooms [1][5].

In order to understand whole wireless transmissions, it is beneficial to construct one's own systems for wireless transmission. In this paper we demonstrate how to build a wireless transmission system with the on-off keying (OOK) modulation scheme, and amplitude modulation (AM) radio broadcasting technology. OOK is the simplest form of amplitude-shift keying (ASK) modulation that represents binary one or zero with carrier presence [6]. Although OOK is not widely used, as it has poor noise sensitivity, the simplicity of the protocol aids it in being very spectrally efficient [7]. This characteristic alone makes the protocol a great foundation for development both as a protocol, and a platform for highly specialized applications. This paper aims to outline the methods of creating a simple communication system that is capable of transmitting information wirelessly from one host to another. This system withholds the ability to parse, modulate, transmit, receive, demodulate, and convert the received information into a copy of the original data.

The rest of this paper is organized as follows: Section 2 outlines the system modules and functionalities. Section 3 describes our implementation details. Section 4 presents our findings and improvements from system performance. Finally, we conclude our work in Section 5.

II. DATA TRANSMISSION WITH OOK

Our data transmission system is designed with modularity in mind, meaning all parts should be easily obtained or substituted for materials of adequate equivalence. This not only makes the system easy to troubleshoot but also opens the doors to anyone interested in the methods developed to implement individually into their own projects. By segregating the project into six sections A - F each system will be independently variable for fine-tuning.

A. Convert text or a file into its binary equivalent

In order to modulate the data, it must first be parsed to its binary equivalent. The importance of this step is to create a singular string of raw data that can be easily interpreted by a program.

B. Modulate parsed data

Once the parsed string is received, it may be modulated. This act requires a program that will be able to both interpret the string of binary and generate an audio file in which the presence of ‘0’ results in the absence of a carrier wave, and the presence of ‘1’ results in the presence of a carrier wave.

C. Transmission of modulated data

Once the data has been successfully modulated into an audio format, it needs to be transmitted over an audible spectrum. This process requires an AM radio broadcast transmitter. The audio file generated has to be

played or relayed to the broadcast transmitter, which broadcasts the data to a predetermined frequency. The length of each bit also must be consistent for the demodulation to be successful.

D. Receiving of modulated data

In order to receive the transmission of data, a receiver unit must be configured to the predetermined frequency. This receiver needs to be connected to a computer capable of recording the broadcast transmission and saving the transmission to disk. This process prepares the transmission for demodulation.

E. Demodulation of data

To demodulate the received audio file, a program needs to be created that interprets the audio file. This program must ingest the file, slice it into segments of a predetermined length, then read the amplitude of each segment. If the segment has an amplitude over a certain threshold, then the program writes a '1' to the output string, if the segment amplitude is below the threshold, the program concatenates a '0' to the output string.

F. Conversion of data into a message or file

After demodulating the transferred file, the received data needs to be converted back into its original message, be it a string, or a photograph. The application must be capable of collecting the output string from the demodulation program and convert it to a copy of the original message. This program has to interpret the string bit by bit to reconstruct the original contents.

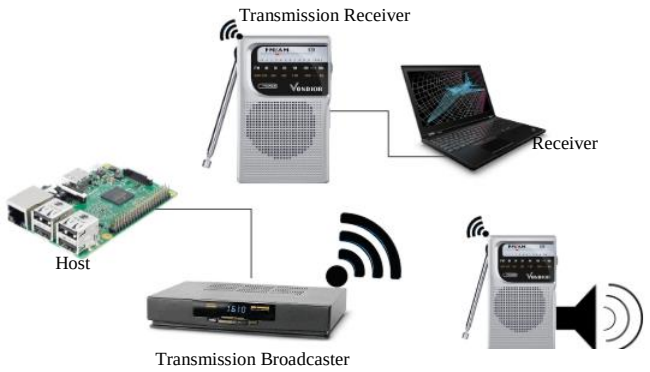

Fig. 1. System Architectural Layout

III. SYSTEM IMPLEMENTATION

Our system is constructed to parse, modulate, transmit, demodulate, and convert data into its final formation. This system utilized a host node, receiver node, transmission broadcaster, and transmission receiver as shown in Figure 1. Software that was used to obtain results is developed in Python, with the inclusion of default Unix programs.

A Raspberry Pi model 3B running Raspbian (Release 2019-04-08) is used as the host. The Linux command *xxd* [8] is used to convert our source files into their binary equivalent. *xxd* creates a hex dump of a given file or standard input, the flag of '-b' was used to switch the output of the program to bits, rather than providing the default hex dump. As shown in Figure 2, the program outputs binary bits as groups of octets that can then be read into the modulation program.

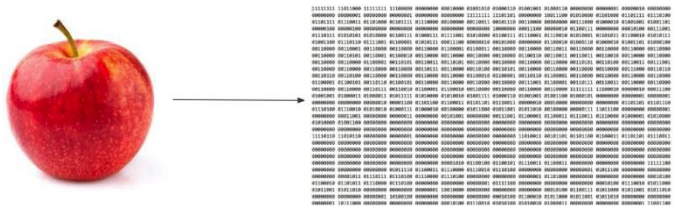

Fig. 2. Image of an apple with its equivalent binary dump

A modulation program is constructed using python. The program makes use of the PyAudio Library [9]. The program follows the CD-DA standard [10] when calculating the bitrate with the equation *bitrate = sample rate* x *bit depth* x *channels*. The program by default works much like Morse code. Each bit has a length of 50ms in Figure 3, whether it is filled with radio silence or a blip. The software modulates the equivalent of 550Hz with a wpm of 120, and a Farnsworth speed of 105 by generating a waveform as shown in Figure 4.

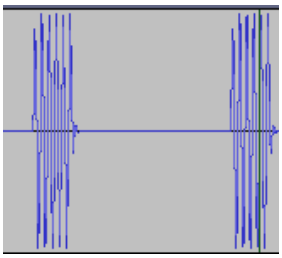

Fig. 3. Example of modulation output.

```
#generating wawes
for x in xrange(NUMBEROFFRAMES):
 WAVEDATA = WAVEDATA+chr(int(math.sin(x/((BITRATE/FREQUENCY)/math.pi))*127+128))

for x in xrange(RESTFRAMES):
 WAVEDATA = WAVEDATA+chr(128)
```

Fig. 4. Snippet of waveform function.

Data transmission is handled by an external AM broadcast transmitter. The transmitter used is equipped with an 8' indoor antenna, with an RF output power of 100mW. The operating frequency for this experiment was 1650kHz, with a stability of +/- 30Hz [11]. Audio

receivers used are generic handheld AM/FM radios with the addition of a 3.5mm Tip Ring Sleeve (TRS) phone connector for audio output.

Retrieval of the transmitted data utilizes the client, in this case, a laptop computer, that has audio recording software running, and the microphone input connected to the phone output of the handheld AM/FM radio receiver. Audio received needs to be trimmed to the start and end of the transmission by the end-user and saved as a windows audio (WAV) file.

Demodulation of the audio back into a binary form is to be handled by a python program based on the *pydub* project [12]. This program completes a number of tasks. First, it segregates the audio file into sections of a predetermined length. In this installment, the sections were 50ms in length. The program then performs an analysis on each segment, capturing the amplitude of each section as shown in Figure 5. The amplitude is compared to a predetermined threshold of -80dB. If the recorded value is above this threshold, the program will write a '1' to file, if it is below the threshold, a '0' will be appended to the output as shown in Figure 6. When the script is finished, a string is returned to the user that is ready for final conversion.

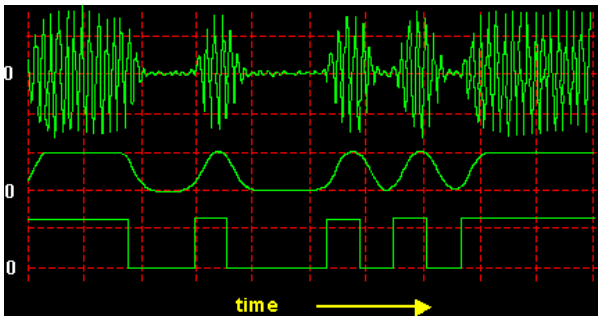

Fig. 5. Example of slices made to an audio file by the demodulation script

```
print()
for x in range (0,len(bit_array)-1):
    if bit_array[x].dBFS > 0:
        sys.stdout.write('1')

    else :
        sys.stdout.write('0')
```

Fig. 6. Example of demodulation script writing each case to file.

Conversion of the output string back into the original message can be handled via various methods and tools. For ease of use, the methods utilized in this project were readily available web applications. Depending on the filetype of the original data, the received data was processed differently. For plaintext, the binary octets were fed directly into a binary to text converter. For image processing, additional steps were used. Before converting an image from its bit form, the octets are converted into their base64 equivalent. From there the base64 data was able to be processed as a file and converted into its original container. This process is time-consuming yet allows for the validity of data to be verified against the original data.

IV. Findings and Improvements

Transmission of the data was completed utilizing an external AM broadcast transmitter, as to maintain modularity of the project. There are multiple factors that can improve the transmission quality of the data. One method is to conduct a preliminary spectrum scan of the operating area, as to locate the frequency that has the least interference for that operating area. Additionally, the gain for both the input device and the RF antenna should be adjusted as to avoid clipping and distortion to the data transmission as shown in Figure 7.

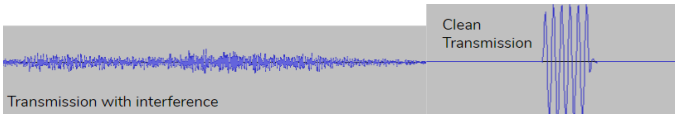

Fig. 7. Comparison between a transmission that contains interference (left), and the original audio transmission (right)

Demodulation of the data was handled by a python script based on the Pydub Library. This method for interpreting data was effective but very inefficient. The program is required to segregate, or slice, each section of audio and copy it into a new file. The software then is required to open and check every single file it had created. This method works for small text string transmissions, however, for larger files, the program would make thousands of files and must scrub through each one individually. Other methods are advised for larger transmissions; however, the process would benefit from some of the previously stated alterations to the project. Additional changes to be made to this segment would include a more accurate threshold for the demodulation, possibly comparing the section of audio to others. Implementing more careful timings would also benefit the process, as the program only splices in 50ms sections. This requires the audio file to be prepared very carefully prior to processing it, as the slightest misalignment would prove to throw off the returned data.

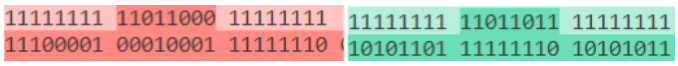

Fig. 8. Comparison between the octets of a transmission that contains the expected received bits (left), and the demodulated bits after transmission (right)

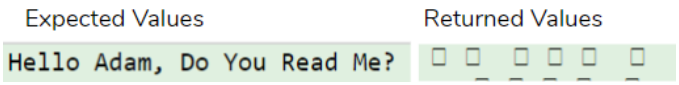

Fig. 9. Comparison between data sent (left) and interpreted data (right)

The system as a whole is inefficient and inaccurate, given all the listed improvements. Transmissions made over the AM spectrum proved to be full of interference, causing for constant misinterpreted bits as shown in Figure 8. Because of the inaccuracy, text transmissions sometimes return invalid characters rather than the expected information, as shown in Figure 9. When working with other filetypes, the transmission would yield a file the client computer cannot interpret as shown in Figure 10. Or when working with a similar file structure, as shown in Figure 11 we can find the addition of corrupted segments in the reconstructed images.

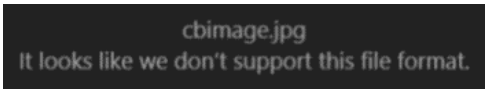

Fig. 10. Demodulated file that, after converting, was uninterpretable by the client machine

Fig. 11. Comparison between original image (left) and reconstructed file that has suffered from corruption (right)

V. Conclusion and Future Work

This paper presents how to create a system that allows the examination of OOK and radio transmissions, and how they operate in real-world implementations. Our system showed the ability to parse, modulate, transmit, receive, demodulate, and convert the received information into a copy of the original data. We firmly believe that experience of building a complete system motivates students to learn actively and encourages them to seek out wireless technologies and improve them.

Future incarnations of this project will contain a more robust solution for each section *A – F*. As the system is modular, each project section can be worked on separately, and swapped into the production system. Additionally, redundancy could be implemented by simply adding more modules in parallel to the existing systems.

References

[1] Mira Yun, Charlie Wiseman, and Leonidas Deligiannidis, "802.11 Wireless Networks: Incorporating Hands-On Learning Experience into the Undergraduate Classroom", In Proc. of the 2013 International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS'13), pp.140-146, Las Vegas, USA, July 2013.

[2] Kismet, https://www.kismetwireless.net/

[3] DD-WRT, https://dd-wrt.com

[4] BATMAN-Adv, https://www.open-mesh.org

[5] Mira Yun, Magdy Ellabidy, and Bowu Zhang, "Project-based Learning Example: Wireless Mesh Networks for Undergraduates", The Journal of Computing Science in Colleges, Vol 30:2, pp.52-59, December 2014.

[6] Tom McDermott, "Wireless Digital Communications: Design and Theory", Tucson Amateur Packet Radio Corporation, Tucson, Arizona, 1996

[7] "A 15-Gb/s Wireless ON-OFF Keying Link." IEEE Access, Access, IEEE, 2014, p. 1307. EBSCOhost.

[8] Linux man page: https://linux.die.net/man/1/xxd

[9] PyAudio Library: https://people.csail.mit.edu/hubert/pyaudio/

[10] Schouhamer Immink, Kees. (2007). Shannon, Beethoven, and the Compact Disc, pp.43-44, April 2019.

[11] TalkingHouse / I A.M. Radio Transmitter: http://www.talkinghouse.com/pdfs/talking-house-i-am-radio-manual.pdf

[12] Pydub Library: http://pydub.com/