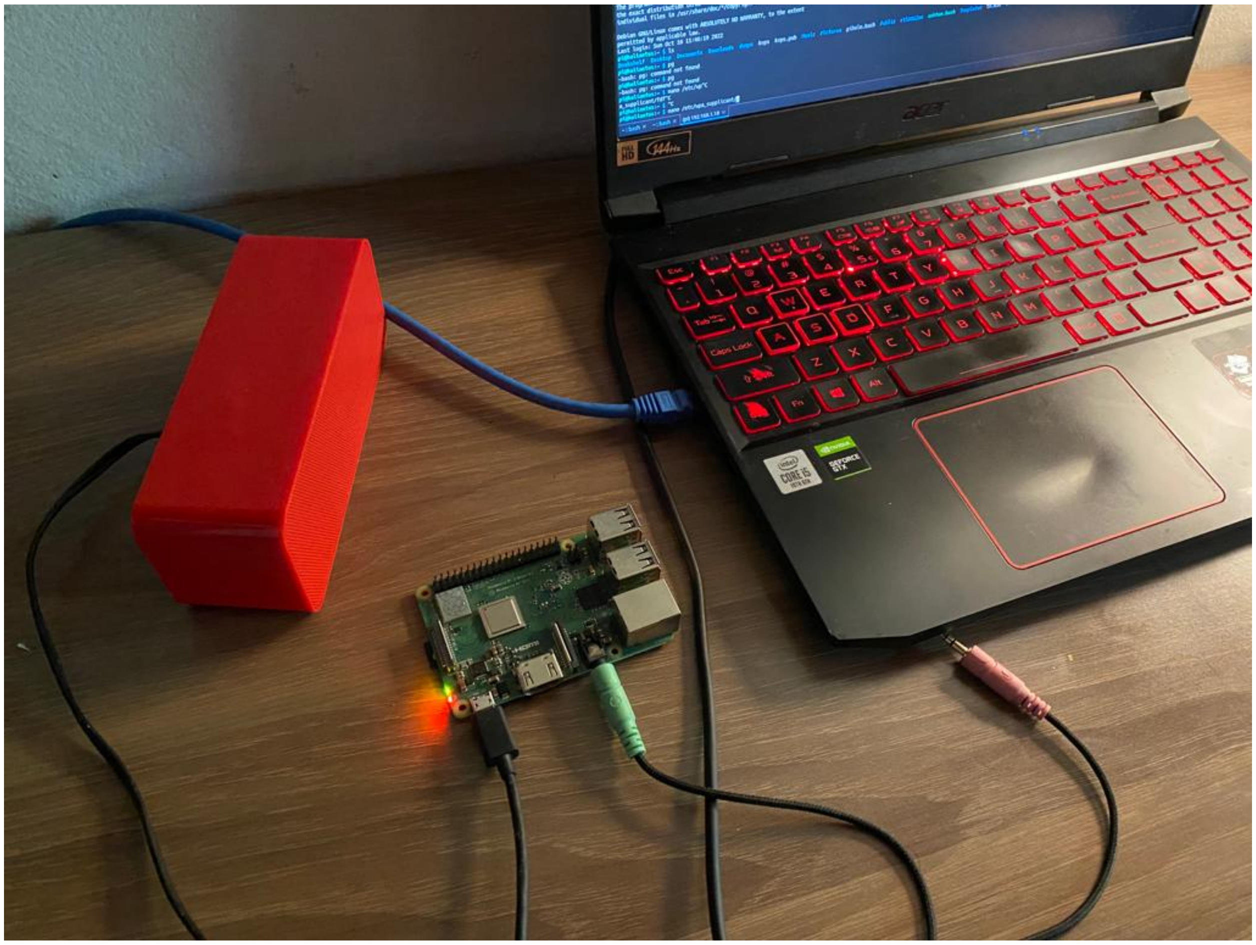
Sonar Transmission Protocol

Rev. 10/16/2022



Matthew Harmon

Introduction

In modern times, the overwhelming majority of wireless communication has been based on the electromagnetic spectrum. This is undoubtedly for good reason, since low frequencies (such as radio waves) travel at the speed of light, through obstacles, and cannot be noticed by the human body. However, in an age where RF protocols dominate, if not completely represent, the domain of wireless communication, it would be beneficial to have another system, fundamentally different in nature, yet still capable of transmission even when traditional methods fail or are unavailable. Enter the Sonar Transmission Protocol, a proof-of-concept data transmission method capable of sending data through sound!

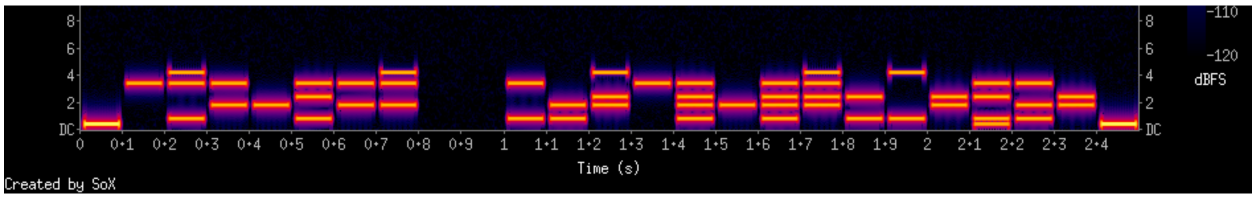
Specifications

Data Transmission Medium: Sound

Data Transmission Mode: Digital

Maximum Data Transmission Rate: Reliably tested up to 50 bits / second, more is possible

At its core, the Sonar Transmission Protocol (STP) functions like any other digital system of communication. It is capable of transmitting data on multiple consecutive frequencies (all while avoiding interference), very similarly to MIMO technology used by modern cellular carriers. The sections below outline the basic functionality of this protocol, including its inner workings, shortcomings, and use cases:



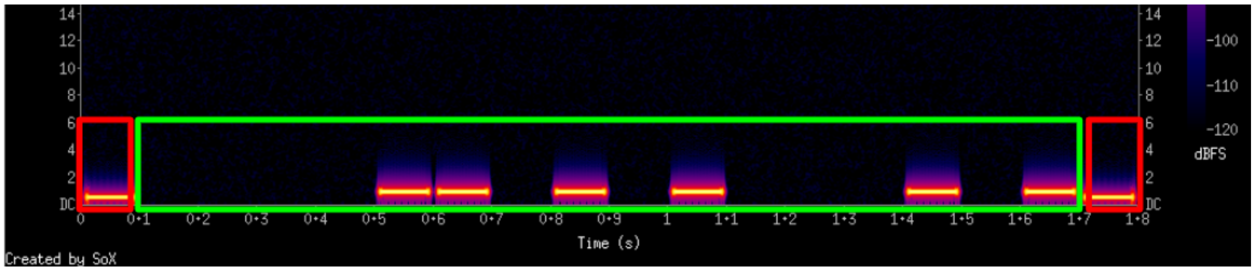
A 5-band, 0.1s pulse length "Hello, world!" spectrogram

Encoding

In order to encode a message, the STP uses the command-line "SOund eXchange" audio editing program, commonly abbreviated as SoX. Widely available on Linux (which was the platform of creation for this protocol), SoX allows for the creation of tones, the manipulation of waveforms, and the visualization of audio through spectrograms, some of which are illustrated here. The encoder program creates a .wav file based on the input message provided, which can be played and then subsequently decoded.

Since the STP uses a file's binary bits to transmit data digitally, all kinds of files, including binary and text, can be transmitted through it.

Components of Encoding



The components of a single-band message. The data stream is shown in the middle, and Prime Tonal Identifiers on the sides.

I. The message

Every bit in a message is allotted a specific time frame, or **pulse duration**. If a bit is 1, a beep will be played during that pulse duration. If it is 0, no sound will be played. In the image shown below, the message is enclosed in the green portion of the spectrogram shown above, and its bits would read "0000110101000101".

II. The PTI

In order for the decoder to identify the beginning and end of the message, two **Prime Tonal Identifiers**, or PTIs, are played at 400Hz for one pulse duration at the beginning and end of a message. They are marked in red in the above spectrogram.

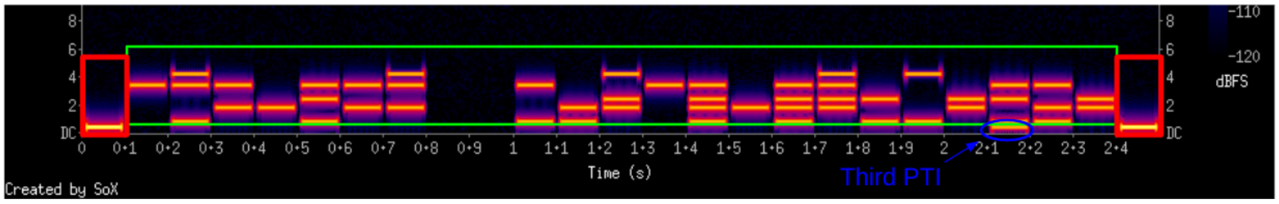
Multi-band messages

In order to achieve faster transmission rates, the Sonar Transmission Protocol supports sending multiple data streams at once. Each "**band**", as is denominated, occupies a previously assigned frequency, as shown below. In an attempt to reduce harmonic noise in higher bands from lower bands, frequencies are not evenly spaced between each other.

Band #	Operating Frequency (hz)
1	800
2	1800

3	2400
4	3400
5	4200

Due to household microphone limitations, the STP can currently encode anywhere from 1 to 5 bands. If microphones with better high-frequency reception were utilized, this number could probably be increased.

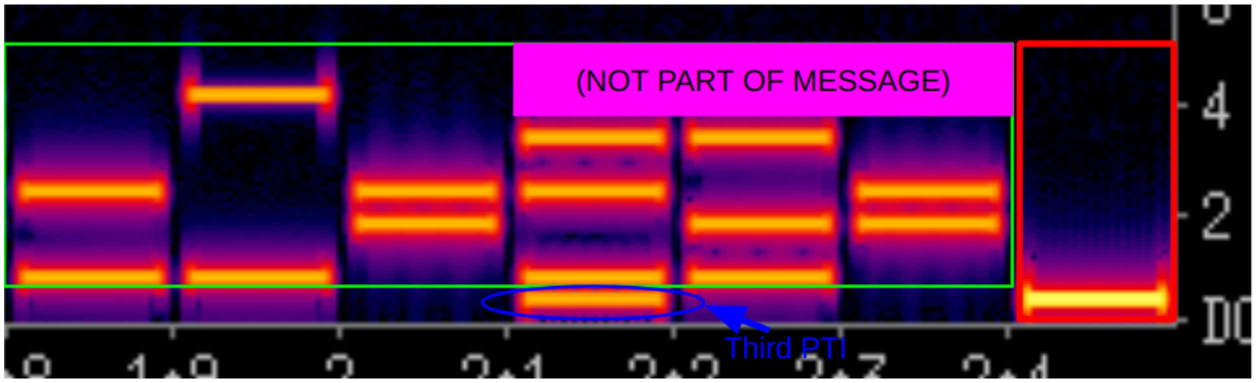


A 5-band message, with the data section marked in green, the first and second PTIs marked in red, and the third PTI circled in blue (explained below).

The Third PTI

During encoding, the input message is split into several equal parts: one for each band. However, in certain circumstances, the last band will have less bits than every other band in the message, if the number of total bits is not a multiple of the number of bands. If this was the case, the decoder would read one or more extra zeros at the end of the message, and without knowing the total length of the message, it would be impossible to know how many (if at all) were superfluous.

Therefore, in multi-band messages, a third PTI is added if the number of bits in a message is not a multiple of the number of bands. This PTI signals the end of the last band of data to the encoder:



Any part of the last band that is during or after the pulse duration of the third PTI is not part of the message.

Decoding

Components of Decoding

Sensitivity

In order to counteract noise and differing sound levels during a message, the decoder has an adjustable sensitivity. All sustained sound above a certain threshold of sensitivity will be counted as a pulse.

Parsing

During decoding, small sections of an input audio file (specifically half a pulse length in duration) are taken and analyzed. If their amplitude (or strength of sound) is stronger than the set sensitivity, it will be counted as a valid pulse. Otherwise, it will not.

Analyzing sections of audio half as long as a pulse length will always guarantee that the full intensity of a pulse will be read at some point, even if the message does not begin exactly at a multiple of the pulse length.

PTI Discovery

The decoder begins by isolating all sound at 400Hz, the frequency at which PTIs are played. It then parses through the message until it finds a sound strong enough to be counted as a PTI.

Once this occurs, it establishes a **frame of reference**, a section of the audio file half a pulse length long. Since the message follows an organized structure, all other pulse lengths any number of pulse lengths from that frame of reference (until the end of the audio file) must also be loud enough to be discoverable.

The decoder then continues searching through the file, finding the second, and if applicable, third PTIs. Their times are then recorded.

Message Decoding

Much like PTI discovery, the decoder begins analyzing the loudness of small sections of audio, half a pulse length in duration, starting from one pulse duration from the frame of reference, and in multiples of the pulse length. It does so beginning in the first band, after isolating its frequency from all other sounds in the audio. Amplitudes above a certain threshold are counted as a 1 bit, while all other quieter noises are counted as 0.

This is done for each band, until the second PTI (or third PTI for the last band of multi-band messages) is found. All bands are then concatenated back into one message, and an output file is created.

Future Plans

The Sonar Transmission Protocol is still in development. For simplicity, it is currently coded in Bash shell script, though this is by no means its final form. Below, intended changes and improvements are listed:

- Error correction
  - Due to time constraints, this has not been possible yet, but error correction is an important part of any communication protocol, especially if strict file integrity is required. An idea is to employ a Hamming Code-like parity check system.
- Higher Frequency Data Transmission
  - Currently, data transmission is extremely loud (and possibly annoying) to bystanders. If speakers and microphones capable of high-frequency (>20kHz) audio transmission could be obtained, this would not be a problem, and would facilitate the usage of more bands, as harmonics would be farther apart and there would be less low-frequency noise.
- More Bands
  - Higher frequencies and better transmission equipment would allow for more bands, which would increase the data transmission rate.
- Different Language
  - A different language, such as Python or C++ would allow more widespread adoption from devices not necessarily restricted to Unix.