Derelict Broadcasting Tower (solution)

by Celestine Lau

Solvers are presented with 8 .wav files each containing a high pitched noise (those with keen ears may be able to pick out two different, but close, pitches). The words “frequency”, “shifted” and “key” in the flavortext clue the type of encoding being used – Frequency Shift Keying (FSK), a technique used by modems and various electronic devices for transmitting digital data across a radio wave. This video (https://www.youtube.com/watch?v=ogJB5fiQ9kM) provides a good introduction.

> Note: To be precise, since the frequency modulation is being performed on audio, it is technically Audio Frequency Shift Keying (AFSK), an older technology used by telephone-line modems, but many of the principles are similar.

To analyze the audio file’s frequencies, Audacity (free to download) or perhaps other online spectrum analyzers like https://academo.org/demos/spectrum-analyzer/ can be used. If using Audacity, this guide (https://manual.audacityteam.org/man/spectrogram_view.html) helps to set up the Spectrogram view where the frequencies can be visualized. Unfortunately the default settings in Audacity are such that only frequencies from 0 to 8kHz are shown, so you would need to open the Spectrogram Settings menu and set the Max Frequency to a higher value (at least 11kHz) to be able to view the frequencies in the files.

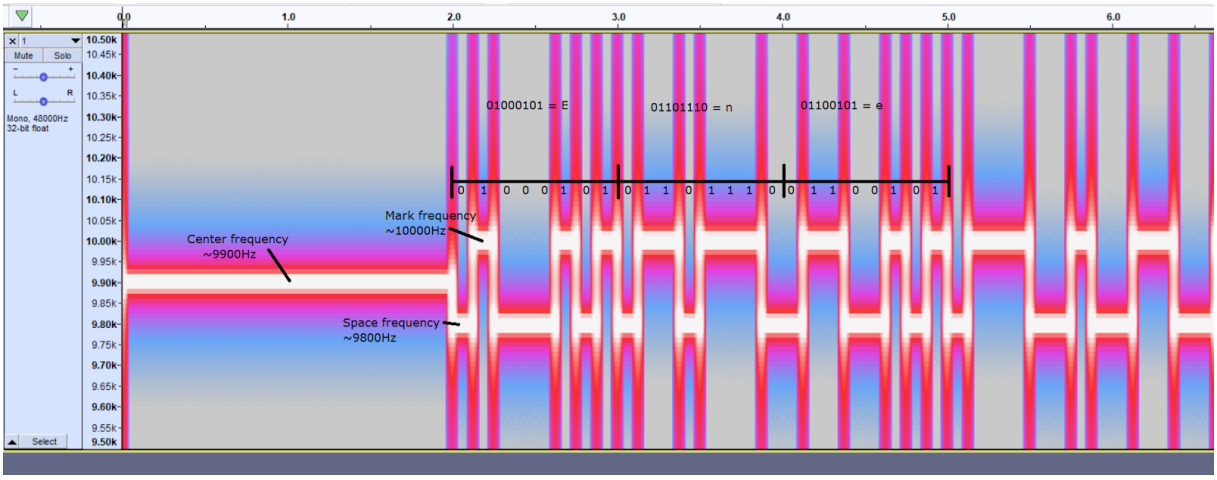
The following properties of each file should be noted:

- The length of the audio is a round number of seconds.
- The first and last 2 seconds of the audio have a constant frequency.
- In between the first and last 2 seconds, the frequencies observed are +100Hz and -100Hz of the constant frequency in the first 2 seconds. Also any frequency changes happen at multiples of ⅛ of a second (after 2s).

> Note: One of the settings in Audacity is “Window size”, this basically trades off between frequency precision and time precision, where a higher window size improves frequency precision but lowers time precision. This is a result of the uncertainty principle, which is not relevant to this puzzle.

Digital data sent using FSK are typically sent using a serial protocol, where the data is encoded as bytes and sent in chunks of 8 bits. This is represented in this puzzle by each byte occupying 1 second of time. The protocol used in this puzzle has 0 start bits, 0 stop bits, no parity bit, and bytes are sent in big-endian (the most significant bit of each byte is earlier in the audio file so the entire bit sequence can be read left to right visually). Endianness is not specified, but solvers should be able to try both and find that only a big-endian representation allows the data to consistently fit within the range of ASCII printable characters. The higher frequency is the *mark frequency* which represents a binary 1, and the lower frequency is the *space frequency* representing a binary 0. This allows groups of 8 bits to be read as a byte representing an ASCII character.

For example, here is the decoding of the first few seconds of 1.wav. Spectrogram settings are set to display 9500-10500 Hz and using a 4096 window size (other settings are default).



The 8 audio files each contain a word clue, which are as follows

Filename	Decoded ASCII characters	Answer
1.wav	Energy from the sun 5 5	SOLAR POWER
2.wav	Beck remix EP 4 3	HELL YES
3.wav	A meeting of tongues? 6 4	FRENCH KISS
4.wav	Cadbury dark chocolate bar 3 4	OLD GOLD
5.wav	One of the best in a field 5 5	WORLD CLASS
6.wav	Novel in four movements 8 8	NAPOLEON SYMPHONY
7.wav	Hole founder 8 4	COURTNEY LOVE
8.wav	Lincoln, Jackson or Madison 5 7	STATE CAPITAL

Solvers should now observe that the second word of each answer corresponds to the name of a radio station in Singapore. This ties in with the overall use of Frequency Modulation (or FM for short) and the presentation of the files as audio files. Each of these radio stations broadcasts at a specific frequency.

Observe also that the 8 audio files are at slightly different frequencies, but are consistent in the spacing between the mark, space, and *center frequency* (the frequency at the first and last 2 seconds of each file which lies exactly in between the 2 frequencies used for binary transmission). In order to accurately determine the center frequency, one tool available in Audacity is Plot Spectrum in the Analyze menu. This should be done while selecting the part of the audio to analyze, ideally the first or last 2 seconds. The displayed peak will correspond to the center frequency. The window size (as discussed above) can be increased for better precision. Once these are obtained, the frequencies in the file should be very close to a multiple of 100Hz above the broadcasting frequency of the associated radio station in each clue (after accounting for a 10^4 order of magnitude difference). As the *shift* (distance of the mark and space frequencies from the center frequency) is also 100Hz, this clues that the number of multiples of 100Hz is meaningful, and in this case, can be used as an index into

the other word.

Answer	Radio Station's frequency	Center frequency in audio file	Index	Extract
SOLAR POWER	98 MHz	9900 Hz	1	S
HELL YES	93.3 MHz	9530 Hz	2	E
FRENCH KISS	92 MHz	9500 Hz	3	E
OLD GOLD	90.5 MHz	9350 Hz	3	D
WORLD CLASS	95 MHz	10000 Hz	5	D
NAPOLEON SYMPHONY	92.4 MHz	9440 Hz	2	A
COURTNEY LOVE	97.2 MHz	10220 Hz	5	T
STATE CAPITAL	95.8 MHz	9880 Hz	3	A

This gives the final answer SEED DATA.

Constructor's Notes:

The encoding of data into a waveform is a technology enabling modern day high-speed communication networks, and I felt would be an interesting subject of a puzzle. One of the decisions I had to make was how to represent the binary data, as frequency shift keying is just the technique of encoding bits, but how these bits represent data varies depending on the protocol used – I eventually settled on one that hopefully is fairly intuitive to decode.

To generate the waveforms, I used https://github.com/kamalmostafa/minimodem, then manually cleaned it up using Audacity (applying high and low pass filters to get rid of unwanted harmonics), and manually added the "center frequencies" to the ends of the original audio. Minimodem automatically encodes data in little endian format, but it was not too hard to write a simple script to reverse the bit representation of the data I wanted to transmit so that it would appear correctly.

I hope this puzzle was a good "introduction" to FSK and also a nod to our local radio stations which are always nice to listen to on a long car ride.