Simulating a 300-baud Modem in Software

Les Kerr

8 April 2005
Revised, 16 July 2006

Dialup Morse requires a 300-baud modem, such as the old Radio Shack DCM-6, that's compatible with the Bell 103 modem specification. Modern modems may be able to connect at 300 baud, but they're designed to transmit ASCII characters, not marks and spaces of arbitrary length. To get around this problem, the Morse KOB program uses a *voice modem*[1] to send and receive audio waveforms over the phone line and it simulates the Bell 103 modem functions in software.

How the software works

The program starts by creating two 800-byte strings, corresponding to 100 ms of digitized sine wave at 8000 samples per second for the 1270 Hz mark and 1070 Hz space tones. (At this sampling rate, the waveforms of both tones repeat every 800 samples.) Each string is divided into ten 80-byte *frames*, with one frame being sent over the phone line every 10 ms.[2]

The modem is initialized with the following commands:

AT+FCLASS=8	Put modem into voice mode
AT+VIP	Initialize voice parameters
AT+VSM=128,8000	Configure for 8-bit linear compression, 8000 samples per second
AT+VLS=1	Connect to phone line and go off hook
AT+VTR	Start full duplex transmit and receive

To the best of my knowledge, these commands conform to ITU-T Recommendation V.253, which means the program has a reasonable chance of working with modems from various manufacturers. It does work with the Actiontec modem in my laptop computer and the Creative modem in my desktop computer.[3]

---

[1] Modems are generally classified as *data*, *data/fax*, or *data/fax/voice*. Voice modems are used in applications that provide telephone answering machine functions such as playing a prerecorded greeting, recording messages that come in over the phone line, etc.

[2] The 1070 and 1270 Hz sine waves are *phase coherent* with each other every 10 ms. This means that the program can safely shift between mark and space at 10 ms intervals without causing any discontinuities in the transmitted signal. In fact, the same property is true for 5 ms frames, but I found my computer couldn't handle such short timing cycles.

[3] The modem *must* support *full-duplex* voice transmission, meaning simultaneous transmit and receive. The Creative Modem Blaster PCI card, available for about $40, is one such product.

The program then launches a high-priority thread that runs every 10 ms, performing the following tasks with each iteration:

- Send the next 80-byte frame to the modem for the desired mark or space tone.
- Get an 80-byte string of input samples from the modem, determine whether the received tone is a mark or space, and drive the sounder accordingly.

The demodulation algorithm

Every 10 ms, an 80-byte string of input samples is read from the modem. Each byte represents a digitized sample of the input waveform, with a value from -128 to +127 (using linear compression). In practice, I've noticed that the amplitude of the tones from the remote modem can be much less than that, perhaps ranging only from -6 to +6. The sampling rate is 8000 samples per second.

The input string is processed in four ways. First it's compared against a precomputed 80-byte string corresponding to a 2025 Hz sine wave. The comparison is done by multiplying each byte of the input signal by the corresponding byte of the reference waveform. These 80 products are then added together, and the sum becomes the result of this first step of the analysis.[4]

The second step is identical to the first, except that the reference 2025 Hz waveform is shifted by a quarter wavelength. Then, the two numbers resulting from the first two steps are squared and added to each other. The resulting value (always positive) is a relative indication of the power of the input signal at the space frequency.

The third and fourth steps of the analysis are the same as the first two, except that the reference waveforms are 2225 Hz sine waves (the mark frequency).

Finally, the mark and space values are compared against each other, and the larger one determines how the input signal is to be interpreted.

[4] I actually do the analysis on just 79 of the 80 bytes in the input string, ignoring the 80th. This is because 9.875 ms (the duration of 79 samples at 8000 samples per second) is very close to a whole number of wavelengths at both 2025 Hz and 2225 Hz. My intuition tells me this might improve the behavior of the algorithm, but I haven't verified my assumption either analytically or empirically.