Digital Concepts

Here, we explore concepts of digital signal transmission. We'll see different digital modulation schemes like Frequency-Shift Keying (FSK), Phase-Shift Keying (PSK) and Quadrature Amplitude Modulation (QAM), and explain the difference between bit rate (bit/s) and Baud (Bd).

Morse to FSK

Let's start with Morse code and "transform" it until we can use it as a basic digital system.

Say we (arbitrarily) represented dots by 0s, and dashes by 1s, the letter "A" (•−) would then be represented as "01".

The **first** thing we need to fix is that the dashes last longer than the dots. In fact, to speed things up, "in 1910, Reginald Fessenden invented a two-tone method of transmitting Morse code. Dots and dashes were replaced with different tones of equal length."[1)]

For example, here's how "VE7" (•••− • −−•••) could sound like if dots were tones of 600 Hz and dashes tones of 800 Hz (with some dead space in between each note for us to hear the breaks):

The **second** thing we need to fix is that the number of 0s and 1s (number of bits) that a letter needs is different for different letters. For example: "V" has four bits (0001) where as "E" only has one (0) and "7" has five (11000).

Since the biggest number of bits in Morse Code is 5, it might be tempting to try and fix this problem by adding leading zeros to all the other letters, but this doesn't work because letters with less than 5 bits might map onto existing letters once we add the extra zeros. For example: we can't make "V" 00001, because that's already "4", and "E" can't be 00000 since that's already "5", and so on.

So in addition to having every bit be the same length, we'd also prefer an encoding method where every character has the same number of bits. To that end, we could use the Baudot Code, which can encode 32 characters of 5 bits each, or the ASCII Standard, which can encode 128 characters of 7 bits each.

The **third** thing we need to fix is to make the notes as short as they can possibly be. With morse code, the speed someone can receive is dependent on their skills, but if a computer had "perfect skills", would there be a theoretical limit to how short a note can be? It turns out there is! The shortest note has to be long enough to include a full cycle of the lowest frequency used. For example, as we'll see below, APRS uses 1200 Hz and 2200 Hz tones for its transmissions. One cycle of 1200 Hz is 1/1200th second (0.000833... s) and 2200 Hz has a cycle of 1/2200th second (0.0004545... s). That means that each "note" has to be at least 1/1200th second long.

The **fourth** and final thing we need to fix is to eliminate the spaces between each note so that the transmission is one long continuous two-note song. This is easy enough, but it leads to a technical issue that we'll address in a moment: how will the receiver know when a note starts and ends if they're all smooched together?

Before we answer that, let's have a look at a real example[2)] of a digital transmission that uses 1200 Hz and 2200 Hz tones:

AFSK_1200_baud.ogg

If we download the audio file and open it in Audacity, we can see the wave form visually:

The first half of the transmission contains the same pattern of one long wave (1200 Hz) followed by 12 short waves (2200 Hz) repeated 76 times. I separated these in red. Since each of these sections lasts 1/150th of a second and each note lasts 1/1200th of a second, each section contains 8 notes (separated here in yellow).

But why spend so much time sending essentially no information? Remember when we removed all the spaces between notes and wondered how the receiver would be able to know when a note starts and finishes? This part acts as a sort of clock to synchronize the receiver to the transmitter. By sending a series of “ticks”, the receiver now knows where each group of 8 notes starts.

Let's take a look at the beginning of the message that follows the synchronization period:

From here, there's a few ways to decide how we should encode the information: do we want each note to literally represent a 0 and a 1? or do we want the *change* in note to represent a change in bit? See this short but very instructive page for more details.

Summary

What we have created here is a Frequency-Shift Keying (FSK) system.[3] The characteristics of (A)FSK are that:

- 0s and 1s are represented by two different tones.
- Each character has the same number of bits.
- Each tone is as short as it can be (one cycle of the lowest tone).
- All notes are transmitted back to back without spaces in between so a sync method is required.

In a sense, FSK is the digital analog of FM, where as ASK (Amplitude-Shift Keying), which we're going to skip here, would be the analog of AM.

Now, there's a distinction we need to make that's not clear yet (but will be soon). There are two ways to describe the speed of the transmission:

- Bit rate (in bit/s, or bps) is the number of 0s and 1s that can be transmitted per second. In our case, it's 1200 bps.
- The Baud (Bd) is the number of notes that can be transmitted per second. In our case, that's also 1200 Bd.

So what's the difference? Well, with FSK, the bit rate is equal to the Baud so the distinction is not clear. But imagine there was a system where each note could somehow encode more than one bit. The bit rate would then be greater than the Baud. We'll see this soon...

PSK

Where as with FSK, the information was encoded in the change of frequency (using two different tones to represent 0s and 1s), PSK encodes the information in the phase. This is a little trickier to conceptualize because while the human ear can detect differences in frequency (as different notes) and differences in amplitude (as different volumes), it can't detect differences in phase.[4)]

In a nutshell, the phase of a wave describes where it starts. Let's take a look at an example:

Supposing we use the blue wave as a reference wave, the green wave is 90° out of phase, while the red wave is 180° out of phase. Notice that all three waves have the same frequency (the number of cycles per second) and the same amplitude (the height of the wave), but they start at different points. To the human ear, all three notes would sound exactly the same, but a computer can be made to detect the difference, which means we can use two of those waves to represent different bits.

Constellation Diagram

A useful way of representing the different “states” that the waves can be in is using a constellation diagram.[5)] For example, using the Blue and Red waves to encode a message, the constellation diagram would have the two dots on the horizontal axis (one on the right at (1,0) and one on the left at (-1,0)).

The distance between a particular point and the centre of the graph represents the amplitude of the wave. In this case, both dots have the same amplitude. The phase is represented as the angle the point makes with the right side of the horizontal axis. So the blue wave has a phase of 0°, while the red wave has a phase of 180°. Finally, we can assign each wave a bit (0 or 1).

Notice that this diagram doesn't tell us anything about the actual frequency of the waves because they are made to be the same. What changes here is the phase, not the frequency.

Before we continue, let's briefly review the difference between a bit rate and Baud:

- Bit rate (in bit/s, or bps) is the number of 0s and 1s that can be transmitted per second.
- The Baud (Bd) is the number of notes that can be transmitted per second.

Again, in this case, the bit rate is the same as the Baud since as with FSK, each note encodes a one bit. But this is about to change!

Higher order PSK

If instead of only using two different phases (0° and 180°) we used eight different phases (0°, 45°, 90°, 135°, 180°, 225°, 270°, and 315°) each “note” can now encode three bits instead of just one! This particular scheme is called 8-PSK.[6)]

The advantage is that while we might only be able to transmit 1200 notes per second (1200 Baud), the data rate is

now triple (3600 bps). That is, each note encodes 3 bits.

But why stop at eight states? Well, in reality, the more states we use, the closer together they become, which means that it can get more and more difficult to tell them apart, which leads to more errors.

Quadrature Amplitude Modulation (QAM)

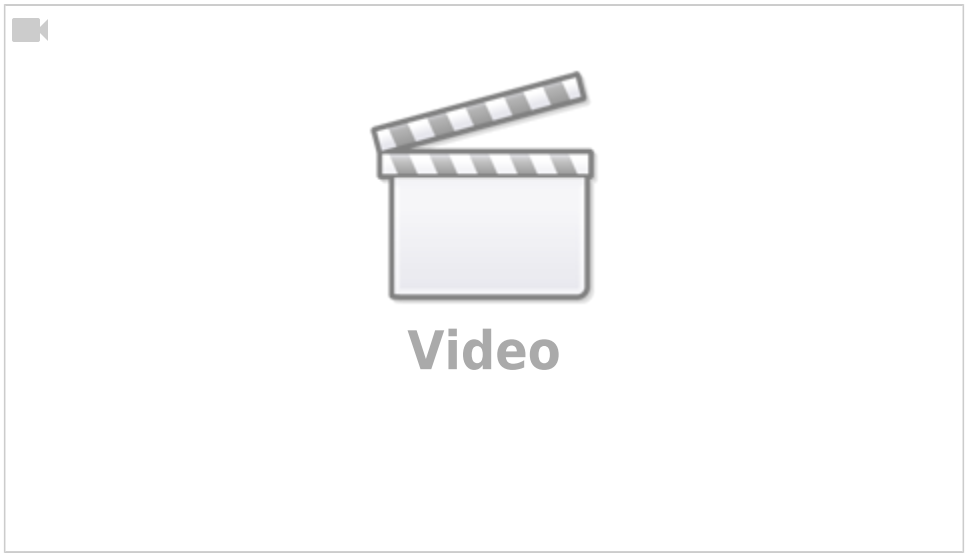
In 8-PSK, the eight states were all on the same circle because their amplitudes were all the same. One way to add more states while keeping them apart as much as possible is to spread them around over the area of a "square" instead of on the perimeter of a circle. This is called Quadrature Amplitude Modulation (QAM).[7)]

QAM is kind of a mix between PSK and ASK (Amplitude Shift Keying). That is, in addition to being able to vary the phase of the waves (where they start), we can also vary their amplitude (how strong the signal is). In the case of 16-QAM, each note can encode 4 bits. So a 1200 Baud signal has a bit rate of 4800 bps.

ADSL technology for copper twisted pairs uses a constellation size of up to 32768-QAM, equivalent to 15 bits per tone.[8)]

1)
Source: Wikipedia: Frequency-Shift Keying
2)
Audio file source: AFSK 1200 Baud file is from Wikipedia: Wikipedia: Frequency Shift Keying
3)
Technically, what we just built is an *Audio* Frequency-Shift Keying (AFSK) system. The difference between the two is that FSK changes the *radio* frequency directly while AFSK changes the *audio* frequency being transmitted. This matters if we're using FM to modulate the signal, but on SSB, changing the audio frequency is equivalent to changing the radio frequency. See Wikipedia: FSK for more details
4)
This is a bit of an over simplification. Two notes out of phase will sound different compared to two notes in phase if they are each played in different ears. That difference is subtle to hear. Some people can't hear the difference, and those who can find it very hard to describe. Phase information between both ears is part of how the brain processes direction (see Wikipedia: Stereophonic sound for more information).

If you listen to this clip with headphones, you'll hear a note (880 Hz tone) being played in phase for 5 seconds, then slowly get out of phase for the next 36 seconds (10° per second), then come back in phase for the last 5 seconds.



It's hard to describe but the out of phase section has a weird 3D depth quality to it. The point is that the human ear can't detect phase nearly as well as frequency (pitch) or amplitude (volume).

This is also the basic idea behind what W8JI calls stereo diversity, which is a super interesting subject in its own right!

5)

BPSK Constellation diagram was modified from Wikipedia

6)

8-PSK Diagram from Wikipedia: Phase-shift keying

7)

16-QAM animation is from Wikipedia: Quadrature Amplitude Modulation

8)

Source: Wikipedia: QAM