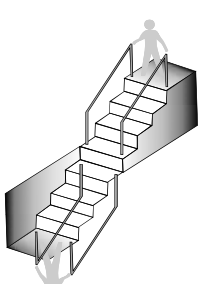
Step 1: The Big Picture

Software Receiver Design: *Build Your Own Digital Communications System in Five Easy Steps* is structured like a staircase with five simple steps. The first chapter presents a naive digital communications system, a sketch of the digital radio, as the first step. The second chapter ascends one step to fill in details and demystify various pieces of the design. Successive chapters then revisit the same ideas, each step adding depth and precision. The first functional (though idealized) receiver appears in Chapter 9. Then the idealizing assumptions are stripped away one by one throughout the remaining chapters, culminating in sophisticated receiver designs in the final chapters. Section 1.3 on page 12 outlines the five steps in the construction of the receiver and provides an overview of the order in which topics are discussed.



1 A Digital Radio

1.1 What Is a Digital Radio?

The fundamental principles of telecommunications have remained much the same since Shannon's time. What has changed, and is continuing to change, is how those principles are deployed in technology. One of the major ongoing changes is the shift from hardware to software—and **Software Receiver Design** reflects this trend by focusing on the design of a digital *software-defined radio* that you will implement in MATLAB.

"Radio" does not literally mean the AM/FM radio in your car; it represents any through-the-air transmission such as television, cell phone, or wireless computer data, though many of the same ideas are also relevant to wired systems such as modems, cable TV, and telephones. "Software-defined" means that key elements of the radio are implemented in software. Taking a "software-defined" approach mirrors the trend in modern receiver design in which more and more of the system is designed and built in reconfigurable software, rather than in fixed hardware. The fundamental concepts behind the transmission are introduced, demonstrated, and (we hope) understood through simulation. For example, when talking about how to translate the frequency of a signal, the procedures are presented mathematically in equations, pictorially in block diagrams, and then concretely as short MATLAB programs.

Our educational philosophy is that it is better to learn by doing: to motivate study with experiments, to reinforce mathematics with simulated examples, to integrate concepts by "playing" with the pieces of the system. Accordingly, each of the later chapters is devoted to understanding one component of the transmission system, and each culminates in a series of tasks that ask you to "build" a particular version of that part of the communication system. In the final chapter, the parts are combined to form a full receiver.

We try to present the essence of each system component in the simplest possible form. We do not intend to show all the most recent innovations (though our

presentation and viewpoint are modern), nor do we intend to provide a complete analysis of the various methods. Rather, we ask you to investigate the performance of the subsystems, partly through analysis and partly using the software code that you have created and that we have provided. We do offer insight into all pieces of a complete transmission system. We present the major ideas of communications via a small number of unifying principles such as transforms to teach modulation, and recursive techniques to teach synchronization and equalization. We believe that these basic principles have application far beyond receiver design, and so the time spent mastering them is well worth the effort.

Though far from optimal, the receiver that you will build contains all the elements of a fully functional receiver. It provides a simple way to ask and answer *what if* questions. What if there is noise in the system? What if the modulation frequencies are not exactly as specified? What if there are errors in the received digits? What if the data rate is not high enough? What if there are distortion, reflections, or echoes in the transmission channel? What if the receiver is moving?

The first step begins with a sketch of a digital radio.

1.2 An Illustrative Design

The first design is a brief tour of a digital radio. If some of the terminology seems obscure or unfamiliar, rest assured that succeeding sections and chapters will revisit the words and refine the ideas. The design is shown in Figures 1.1 through 1.5. While talking about these figures, it will become clear that some ideas are being oversimplified. Eventually, it will be necessary to come back and examine these more closely.

> The boxed notes are reminders to return and think about these areas more deeply later on.

In keeping with Shannon's goal of reproducing at one point a message known at another point, suppose that it is desired to transmit a text message from one place to another. Of course, there is nothing magical about text; however, .mp3 sound files, .jpg photos, snippets of speech, raster-scanned television images, or any other kind of information would do, as long as it can be appropriately digitized into ones and zeros.

> Can *every* kind of message be digitized into ones and zeros?

Perhaps the simplest possible scheme would be to transmit a pulse to represent a one and to transmit nothing to represent a zero. With this scheme, however, it is hard to tell the difference between a string of zeros and no transmission at

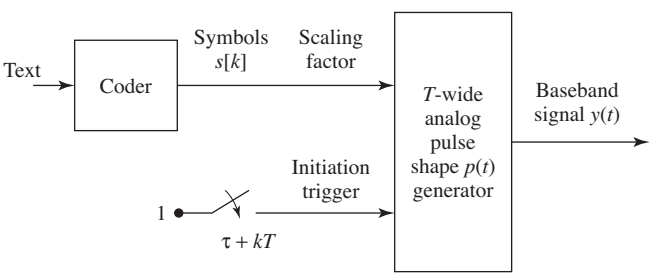


Figure 1.1 An idealized baseband transmitter.

all. A common remedy is to send a pulse with a positive amplitude to represent a one and a pulse of the same shape but negative amplitude to represent a zero. In fact, if the receiver could distinguish pulses of different sizes, then it would be possible to send two bits with each symbol, for example, by associating the amplitudes[1] of $+1$, -1, $+3$, and -3 with the four choices 10, 01, 11, and 00. The four symbols ± 1, ± 3 are called the *alphabet*, and the conversion from the original message (the text) into the symbol alphabet is accomplished by the *coder* in the transmitter diagram Figure 1.1. The first few letters, the standard ASCII (binary) representation of these letters, and their coding into symbols are as follows

letter	binary ASCII code	symbol string
a	01 10 00 01	$-1,\ 1,\ -3,\ -1$
b	01 10 00 10	$-1,\ 1,\ -3,\ \ 1$
c	01 10 00 11	$-1,\ 1,\ -3,\ \ 3$
d	01 10 01 00	$-1,\ 1,\ -1,\ -3$
$\vdots$	$\vdots$	$\vdots$

(1.1)

In this example, the symbols are clustered into groups of four, and each cluster is called a *frame*. Coding schemes can be designed to increase the security of a transmission, to minimize the errors, or to maximize the rate at which data are sent. This particular scheme is not optimized in any of these senses, but it is convenient to use in simulation studies.

> Some codes are better than others. How can we tell?

To be concrete, let

- the *symbol interval* T be the time between successive symbols, and
- the *pulse shape* $p(t)$ be the shape of the pulse that will be transmitted.

[1] Many such choices are possible. These particular values were chosen because they are equidistant and so noise would be no more likely to flip a 3 into a 1 than to flip a 1 into a -1.

For instance, $p(t)$ may be the rectangular pulse

$$p(t) = \begin{cases} 1 & \text{when } 0 \le t < T, \\ 0 & \text{otherwise,} \end{cases} \tag{1.2}$$

which is plotted in Figure 1.2(a). The transmitter of Figure 1.1 is designed so that every T seconds it produces a copy of $p(\cdot)$ that is scaled by the symbol value $s[\cdot]$. A typical output of the transmitter in Figure 1.1 is illustrated in Figure 1.2(b) using the rectangular pulse shape. Thus the first pulse begins at some time τ and it is scaled by $s[0]$, producing $s[0]p(t-\tau)$. The second pulse begins at time $\tau + T$ and is scaled by $s[1]$, resulting in $s[1]p(t-\tau-T)$. The third pulse gives $s[2]p(t-\tau-2T)$, and so on. The complete output of the transmitter is the sum of all these scaled pulses:

$$y(t) = \sum_i s[i]p(t-\tau-iT).$$

Since each pulse ends before the next one begins, successive symbols should not interfere with each other at the receiver. The general method of sending information by scaling a pulse shape with the amplitude of the symbols is called *Pulse Amplitude Modulation* (PAM). When there are four symbols as in (1.1), it is called 4-PAM.

For now, assume that the path between the transmitter and receiver, which is often called the *channel*, is "ideal." This implies that the signal at the receiver is the same as the transmitted signal, though it will inevitably be delayed (slightly) due to the finite speed of the wave, and attenuated by the distance. When the ideal channel has a gain g and a delay δ, the received version of the transmitted signal in Figure 1.2(b) is as shown in Figure 1.2(c).

There are many ways in which a real signal may change as it passes from the transmitter to the receiver through a real (nonideal) channel. It may be reflected from mountains or buildings. It may be diffracted as it passes through the atmosphere. The waveform may smear in time so that successive pulses overlap. Other signals may interfere additively (for instance, a radio station broadcasting at the same frequency in a different city). Noises may enter and change the shape of the waveform.

There are two compelling reasons to consider the telecommunication system in the simplified (idealized) case before worrying about all the things that might go wrong. First, at the heart of any working receiver is a structure that is able to function in the ideal case. The classic approach to receiver design (and also the approach of **Software Receiver Design**) is to build for the ideal case and later to refine so that the receiver will still work when bad things happen. Second, many of the basic ideas are clearer in the ideal case.

The job of the receiver is to take the received signal (such as that in Figure 1.2(c)) and to recover the original text message. This can be accomplished by an idealized receiver such as that shown in Figure 1.3. The first task this receiver must accomplish is to sample the signal to turn it into computer-friendly digital

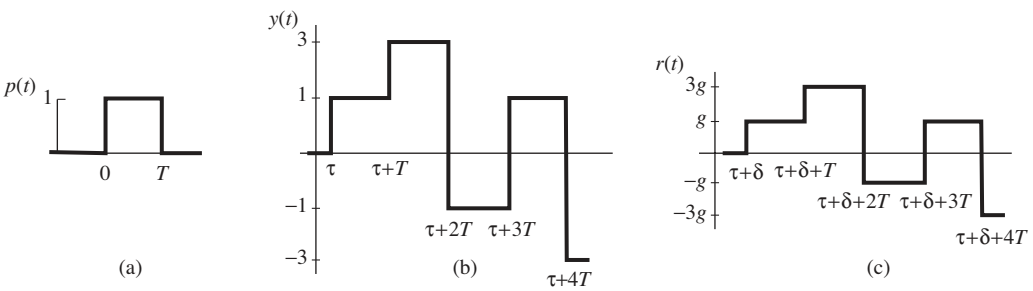


Figure 1.2 (a) An isolated rectangular pulse. (b) The transmitted signal consists of a sequence of pulses, one corresponding to each symbol. Each pulse has the same shape as in (a), though offset in time (by τ) and scaled in magnitude (by the symbols $s[k]$). (c) In the ideal case, the received signal is the same as the transmitted signal of (b), though attenuated in magnitude (by g) and delayed in time (by δ).

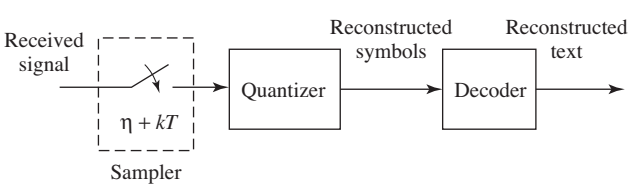


Figure 1.3 An idealized baseband receiver.

form. But when should the samples be taken? On comparing Figures 1.2(b) and 1.2(c), it is clear that if the received signal were sampled somewhere near the middle of each rectangular pulse segment, then the quantizer could reproduce the sequence of source symbols. This quantizer must either

1. know g so the sampled signal can be scaled by $1/g$ to recover the symbol values, or
2. separate $\pm g$ from $\pm 3g$ and output symbol values ± 1 and ± 3.

Once the symbols have been reconstructed, then the original message can be decoded by reversing the assignment of letters to symbols used at the transmitter (for example, by reading (1.1) backwards). On the other hand, if the samples were taken at the moment of transition from one symbol to another, then the values might become confused.

To investigate the timing question more fully, let T be the sample interval and τ be the time at which the first pulse begins. Let δ be the time it takes for the signal to move from the transmitter to the receiver. Thus the $(k+1)$st pulse, which begins at time $\tau + kT$, arrives at the receiver at time $\tau + kT + \delta$. The midpoint of the pulse, which is the best time to sample, occurs at $\tau + kT + \delta + T/2$. As indicated in Figure 1.3, the receiver begins sampling at time η, and then samples regularly at $\eta + kT$ for all integers k. If η were chosen so that

$$\eta = \tau + \delta + T/2, \tag{1.3}$$

then all would be well. But there are two problems: the receiver does not know when the transmission began, nor does it know how long it takes for the signal to reach the receiver. Thus both τ and δ are unknown!

Somehow, the receiver must figure out *when* to sample.

Basically, some extra "synchronization" procedure is needed in order to satisfy (1.3). Fortunately, in the ideal case, it is not really necessary to sample exactly at the midpoint; it is necessary only to avoid the edges. Even if the samples are not taken at the center of each rectangular pulse, the transmitted symbol sequence can still be recovered. But if the pulse shape were not a simple rectangle, then the selection of η would become more critical.

How does the pulse shape interact with timing synchronization?

Just as no two clocks ever tell exactly the same time, no two independent oscillators[2] are ever exactly synchronized. Since the symbol period at the transmitter, call it T_{trans}, is created by a separate oscillator from that creating the symbol period at the receiver, call it T_{rec}, they will inevitably differ. Thus another aspect of timing synchronization that must ultimately be considered is how to automatically adjust T_{rec} so that it aligns with T_{trans}.

Similarly, no clock ticks out each second exactly evenly. Inevitably, there is some jitter, or wobble in the value of T_{trans} and/or T_{rec}. Again, it may be necessary to adjust η to retain sampling near the center of the pulse shape as the clock times wiggle about. The timing adjustment mechanisms are not explicitly indicated in the sampler box in Figure 1.3. For the present idealized transmission system, the receiver sampler period and the symbol period of the transmitter are assumed to be identical (both are called T in Figures 1.1 and 1.3) and the clocks are assumed to be free of jitter.

What about clock jitter?

Even under the idealized assumptions above, there is another kind of synchronization that is needed. Imagine joining a broadcast in progress, or one in which the first K symbols have been lost during acquisition. Even if the symbol sequence is perfectly recovered after time K, the receiver would not know which recovered symbol corresponds to the start of each frame. For example, using the letters-to-symbol code of (1.1), each letter of the alphabet is translated into a sequence of four symbols. If the start of the frame is off by even a single sym-

[2] Oscillators, electronic components that generate repetitive signals, are discussed at length in Chapter 3.

bol, the translation from symbols back into letters will be scrambled. Does this sequence represent a or X?

$$\begin{array}{c} \overbrace{}^{a} \\ -1,\ -1,\ 1,-3,-1 \\ \underbrace{-1,-1,\ 1,-3,}_{X}-1 \end{array}$$

Thus proper decoding requires locating where the frame starts, a step called frame synchronization. Frame synchronization is implicit in Figure 1.3 in the choice of η, which sets the time t ($= \eta$ with $k = 0$) of the first symbol of the first (character) frame of the message of interest.

How to find the start of a frame?

In the ideal situation, there must be no other signals occupying the same frequency range as the transmission. What bandwidth (what range of frequencies) does the transmitter (1.1) require? Consider transmitting a single T-second-wide rectangular pulse. Fourier transform theory shows that any such time-limited pulse cannot be truly bandlimited, that is, cannot have its frequency content restricted to a finite range. Indeed, the Fourier transform of a rectangular pulse in time is a sinc function in frequency (see Equation (A.20) in Appendix A). The magnitude of this sinc is overbounded by a function that decays as the inverse of frequency (peek ahead to Figure 2.11). Thus, to accommodate this single-pulse transmission, all other transmitters must have negligible energy below some factor of $B = 1/T$. For the sake of argument, suppose that a factor of 5 is safe, that is, all other transmitters must have no significant energy within $5B$ Hz. But this is only for a single pulse. What happens when a sequence of T-spaced, T-wide rectangular pulses of various amplitudes is transmitted? Fortunately, as will be established in Section 11.1, the bandwidth requirements remain about the same, at least for most messages.

What is the relation between the pulse shape and the bandwidth?

One fundamental limitation to data transmission is the trade-off between the data rate and the bandwidth. One obvious way to increase the rate at which data are sent is to use shorter pulses, which pack more symbols into a shorter time. This essentially reduces T. The cost is that this would require excluding other transmitters from an even wider range of frequencies since reducing T increases B.

What is the relation between the data rate and the bandwidth?

If the safety factor of $5B$ is excessive, other pulse shapes that would decay faster as a function of frequency could be used. For example, rounding the sharp corners of a rectangular pulse reduces its high-frequency content. Similarly, if other transmitters operated at high frequencies outside $5B$ Hz, it would be sensible to add a lowpass filter at the front end of the receiver. Rejecting frequencies outside the protected $5B$ baseband turf also removes a bit of the higher-frequency content of the rectangular pulse. The effect of this in the time domain is that the received version of the rectangle would be wiggly near the edges. In both cases, the timing of the samples becomes more critical as the received pulse deviates further from rectangular.

One shortcoming of the telecommunication system embodied in the transmitter of Figure 1.1 and the receiver of Figure 1.3 is that only one such transmitter at a time can operate in any particular geographical region, since it hogs all the frequencies in the baseband, that is, all frequencies below $5B$ Hz. Fortunately, there is a way to have multiple transmitters operating in the same region simultaneously. The trick is to translate the frequency content so that instead of all transmitters trying to operate in the 0 and $5B$ Hz band, one might use the $5B$ to $10B$ band, another the $10B$ to $15B$ band, etc. Conceivably, this could be accomplished by selecting a different pulse shape (other than the rectangle) that has no low-frequency content, but the most common approach is to "modulate" (change frequency) by multiplying the pulse-shaped signal by a high-frequency sinusoid. Such a "radio-frequency" (RF) transmitter is shown in Figure 1.4, though it should be understood that the actual frequencies used may place it in the television band or in the range of frequencies reserved for cell phones, depending on the application.

At the receiver, the signal can be returned to its original frequency (demodulated) by multiplying by another high-frequency sinusoid (and then lowpass filtering). These frequency translations are described in more detail in Section 2.6, where it is shown that the modulating sinusoid and the demodulating sinusoid must have the same frequencies and the same phases in order to return the signal to its original form. Just as it is impossible to align any two clocks exactly, it is also impossible to generate two independent sinusoids of exactly the same frequency and phase. Hence there will ultimately need to be some kind of "carrier synchronization," a way of aligning these oscillators.

> How can the frequencies and phases of these two sinusoids be aligned?

Adding frequency translation to the transmitter and receiver of Figures 1.1 and 1.3 produces the transmitter in Figure 1.4 and the associated receiver in Figure 1.5. The new block in the transmitter is an analog component that effectively adds the same value (in Hz) to the frequencies of all of the components of the baseband pulse train. As noted, this can be achieved with multiplication by a "carrier" sinusoid with a frequency equal to the desired translation. The new

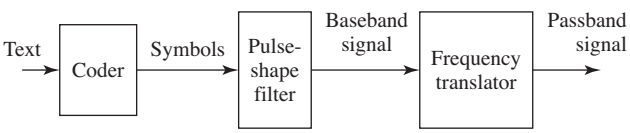


Figure 1.4 "Radio-frequency" transmitter.

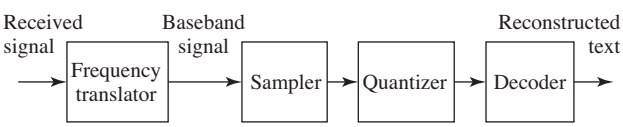


Figure 1.5 "Radio-frequency" receiver.

block in the receiver of Figure 1.5 is an analog component that processes the received analog signal prior to sampling in order to subtract the same value (in Hz) from all components of the received signal. The output of this block should be identical to the input to the sampler in Figure 1.3.

This process of translating the spectrum of the transmitted signal to higher frequencies allows many transmitters to operate simultaneously in the same geographical area. But there is a price. Since the signals are not completely band-limited to within their assigned $5B$-wide slot, there is some inevitable overlap. Thus the residual energy of one transmitter (the energy outside its designated band) may interfere with other transmissions. Solving the problem of multiple transmissions has thus violated one of the assumptions for an ideal transmission. A common theme throughout **Software Receiver Design** is that a solution to one problem often causes another!

> There is no free lunch. How much does the fix cost?

In fact, there are many other ways in which the transmission channel can deviate from the ideal, and these will be discussed in detail later on (for instance, in Section 4.1 and throughout Chapter 9). Typically, the cluttered electromagnetic spectrum results in a variety of distortions and interferences:

- in-band (within the frequency band allocated to the user of interest)
- out-of-band (frequency components outside the allocated band such as the signals of other transmitters)
- narrowband (spurious sinusoidal-like components)
- broadband (with components at frequencies across the allocated band and beyond, including *thermal noise* introduced by the analog electronics in the receiver)
- fading (when the strength of the received signal fluctuates)
- multipath (when the environment contains many reflective and absorptive objects at different distances, the transmission delay will differ across different paths, smearing the received signal and attenuating some frequencies more than others)