Module 10: Bits over the Air

© 2019 Christoph Studer (studer@cornell.edu); Version 0.1

In this module, you will finally design an amplitude modulation (AM) communication system that transmits bits over the air. To make it work, we will use everything you have learned so far: AM modulation, simultaneous playback and recording in MATLAB, synchronization, AM demodulation, and detection. The goal of this module is to successfully transmit a few bits over a short distance. ***Remember: Please ask us in case you have questions or problems—it may appear easy to transition from our AWGN channel models to a real over-the-air experiment, but you will see that it is trickier than it may seem!***

10 Bits over the Air

In essence, we will now replace the MATLAB AWGN channel with the loudspeaker and microphone and transmit digital information over the air. To this end, we will be using simultaneous playback and recording as learned in Module 5. While it may appear straightforward to simply replace the channel model with a real over-the-air channel, some new difficulties will appear. First, acoustic channels introduce echoes—the transmit signal is reflected from surrounding objects and multiple copies of the same signal arrive at the microphone at different time instants. Second, depending on the distance between the loudspeakers and the microphone, and the loudness setting of your loudspeakers, the received signal can have a different amplitude. Third, other groups are talking and performing experiments nearby and at the same time, which can cause interference. All of these effects combined may cause transmission errors. We emphasize that each of these problems also arrise in practical wireless communication systems that operate over the electromagnetic spectrum. Radio signals are reflected at physical objects, the distance between the transmitter and receiver can change, and other communication services are transmitting nearby. Hence, you are experiencing a scenario that very much resembles a real wireless communication system. Fortunately, we have designed an extremely robust wireless communication system that uses digital amplitude modulation. If everything is set up properly, error-free transmission should be possible!

10.1 Overview of the Entire Communication System

Figure 25 illustrates the entire communication system. As you will see, we have already implemented each of these building blocks in the previous modules. Each building block is detailed next:

1. We create a vector `bits` that contains the digital information we want to transmit. In this example, we transmit eight bits $[1, 0, 0, 1, 0, 1, 1, 1]$; note that the first bit was set to be a binary 1 to simplify synchronization at the receiver.

2. We use digital amplitude modulation (AM) to modulate the digital information onto the amplitude of sine waves at carrier frequency `FC` of length `tx_len`. In this example, we use a carrier frequency of 1,760 Hz and each sine wave consists of 1000 samples.

3. We simultaneously transmit and receive the AM signal over the loudspeakers and record it with the microphone. The received signal is attenuated and contains noise; in addition, the receiver was not synchronized properly.
4. We rectify the signal.
5. We perform synchronization and crop the signal so that we only have the noisy transmit signal.
6. We compute the average amplitudes for each transmit bit.
7. We perform detection: large values are assigned to a bit 1 and smaller to a bit 0.

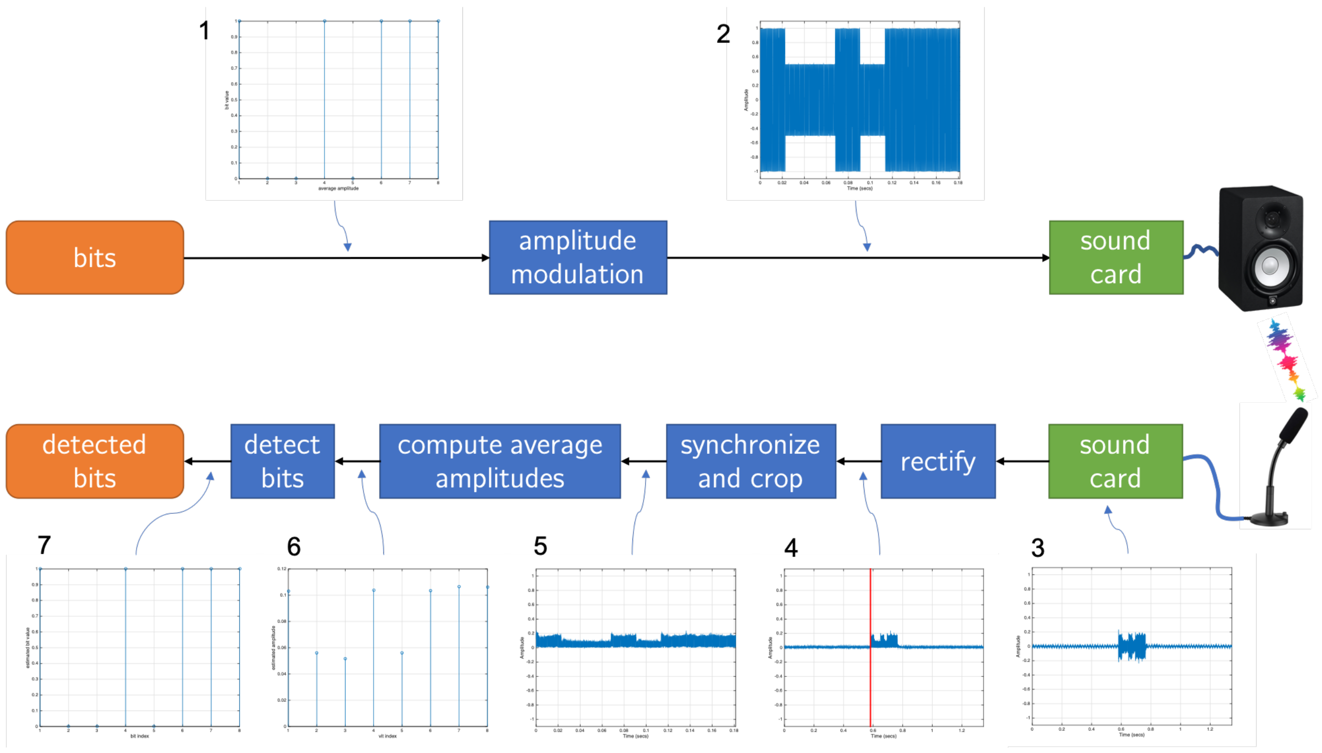


Figure 25: Overview of the final digital ommunication system. Top: transmitter; bottom: receiver.

The key components of this digital AM communication system have been designed in Modules 8 and 9. In principle, you only need to replace the AWGN channel model function `channel_AWGN_delay` with the simultaneous playback and recording example we have used in Module 5.

Activity 39: Implement the communication system and try it out!

Implement the entire communication system as illustrated in Figure 25. You should have each of the individual tasks implemented already. Also, synchronization should be working well by now. To make sure that everything is working, place the microphone close (a few inches) to the loudspeaker and use the plot functions to visualize all intermediate signals (as done in Figure 25). ***Visualizing intermediate results is extremely helpful in finding errors!***

Important: It is always good practice to have a look at the signals and vectors you generated, transmit, and receive—programming mistakes can occur everywhere! Also, make sure that synchronization is working properly; this is a common cause of transmission errors. Also make sure that the received signal

is not too loud or too weak, which can cause issues as well. Furthermore, be careful with the amount of pause before recording, as the delay of the computer and your MATLAB version may vary. You should play with the pause time parameter to ensure that you have recorded the entire transmit signal!

10.2 Exploring the Main System Parameters

Your wireless communication system contains a number of parameters we have fixed but can be changed. To gain more intuition about their impact on the performance of the communication system, you should first guess what will happen when changing a certain parameter and then try it out to see whether you were right. If you were wrong, you can try to explain why. Here are a few ideas:

- Change the carrier frequency `FC`. Try to understand what will happen if you increase or decrease this parameter. Also look at the spectrum and spectrogram of the generated digital AM signal.
- Reduce the length of each sine wave `tx_len`. How small can you make it without having errors? What happens to the spectrum and spectrogram of the signal? Can you explain why the system will stop working if you reduce this parameter by too much?
- Change the distance between the loudspeakers and the microphone. Change the orientation of the loudspeakers (e.g., pointing away from the microphone). Turn down the volume of the loudspeaker. What do you observe? How far can you transmit bits without having errors?
- What happens if you speak in the microphone during transmission. Will the system still work?
- What happens if you change the sine wave into a square wave? Does the communication system still work? What happens to the spectrum and spectrogram of the output signal?

We would like to reiterate that this acoustic communication resembles very much a real wireless communication system that operates over the wireless spectrum. Everything you observe here also happens with Wi-Fi or LTE cellular communication with the exception that these systems contain a number of very clever tricks to transmit even more bits in shorter time while maintaining very high reliability.

Activity 40: Play with Parameters

Perform some of the experiments listed above and try to understand what is going on. The goal is to understand each of the building blocks of the communication system and the associated parameters. ***Talk to us if you have questions or if you are interested in more detailed explanations on each of the effects you are observing!***

10.3 Transmitting Data from one Computer to Another

So far, we have transmitted data from the loudspeakers to a microphone on the same computer. Can you set up the same system where you transmit the data from one computer and successfully receive the data from another computer? To do this, work with the other group you are assigned to. MATLAB on one computer needs to implement only the transmitter, including playing back the digital AM signal. MATLAB on the other computer only needs to implement the receiver.

Activity 41: Transmit data from one computer to another

Team up with the other group you are assigned to. Open MATLAB on two neighboring machines. Try to transmit information from one computer to the other using the loudspeakers of one machine and the microphone of the other machine. If that is working, try to transmit information from your group's MATLAB code to a receiver implemented by the other group. The goal of this activity is to check compatibility between the two *standards* that both groups have implemented. Note that successful transmission will only work if you use the same carrier frequency, the same length of the sine waveform, and if you know how many bits are transmitted. Make sure that both systems are compatible!

Real-world communication systems are designed based on *standards*. For example, IEEE 802.11n is a wireless communication standard that describes how to transmit data for Wi-Fi (also known as wireless LAN). By paying a fee to IEEE, one can get access to the standard document, which precisely describes how bits are modulated and transmitted—interestingly, standards do not describe how to build the receiver. Note that it is sufficient to describe the modulation process, as the demodulation process essentially has to reverse the modulation operation. In practice, there are many ways to perform demodulation. In our example, we used rectifying and filtering for AM demodulation. (For AM demodulation other, and actually much better, methods exists, but these methods are quite complicated and would be part of a university-level digital communications course.) Every manufacturer of wireless devices implements their transmitters exactly according to this standard and builds receivers that are compatible with the way data is transmission. This approach enables reliable communication between devices even though they are from different manufacturers. The above activity demonstrates the importance of having exactly the same system parameters to enable interoperability among devices.

10.4 Transmitting Text over the Air

So far, we have been transmitting hand generated bits or randomly created information. In practice, one would like to transmit a text message or an image, for example. You will now learn how to send a short text message over the air. Before we do so, make sure that your communication system is working reliably!

Text in a computer is represented as binary-valued information. Each character of a string can be represented by eight bits; the mapping from character to bit is defined by the well-known ASCII standard. For example, the character "A" is represented in ASCII by the bit string $[1, 0, 0, 0, 0, 0, 1, 0, 1]$ which corresponds to 65 in decimal representation. We can now transmit a MATLAB string over the air using two functions we provide. The first function

```
bits = string2bits('This is a test!')
```

generates a vector of bits that represent the string "This is a test!" Note that we prepend a binary 1 to the beginning of the bit string to simplify synchronization. We also provide the reverse function that takes a vector of bits and converts it back to a string:

```
string = bits2string(bits)
```

You should obtain the original string from this function when used with the argument `bits`. You can now conveniently define a string, transmit the associated vector of bits over the air, and convert the received bits in the vector `bits_detected` back to a string. In case a transmission error happened, the transmitted string will contain wrong characters.

Activity 42: Transmit text over the air

Use the functions `string2bits` and `bits2string` together with your MATLAB code to transmit text over the air. Longer strings will take longer to transmit—shorter strings can be transmitted more quickly. How many seconds does it take you to reliably transmit 280 characters? What is the data rate (number of bits per second) of your communication system? What are the parameters of your system that can be used to increase the data rate? How high can you make the data rate without having errors in the transmitted text?

10.5 Transmitting Images over the Air

You will now learn how to transmit images over the air using your communication system. In a computer, images are represented as digital information in a similar fashion as audio signals. However, instead of sampling values in time, for images one samples pixels in space. How dense one samples is known as the resolution of an image—you may have heard of digital cameras that have a resolution of several megapixels. In addition to fixing a resolution, one also has to determine how many bits are used to represent each color (or intensity) of a pixel. In what follows, we will be transmitting relatively small simple black-white images. This simplifies the conversion of an image into bits that we can transmit and also does not take too long to transmit them over the air. In addition, we can directly visualize the received images even if transmission errors happened. To simplify implementation, we provide functions to load 32×32 pixel grayscale images and convert them into binary, black-white images. You can then transmit the associated bits using your communication system. We also provide a function to convert the binary string back to an image. If you have questions on how these functions work, feel free to talk to us!

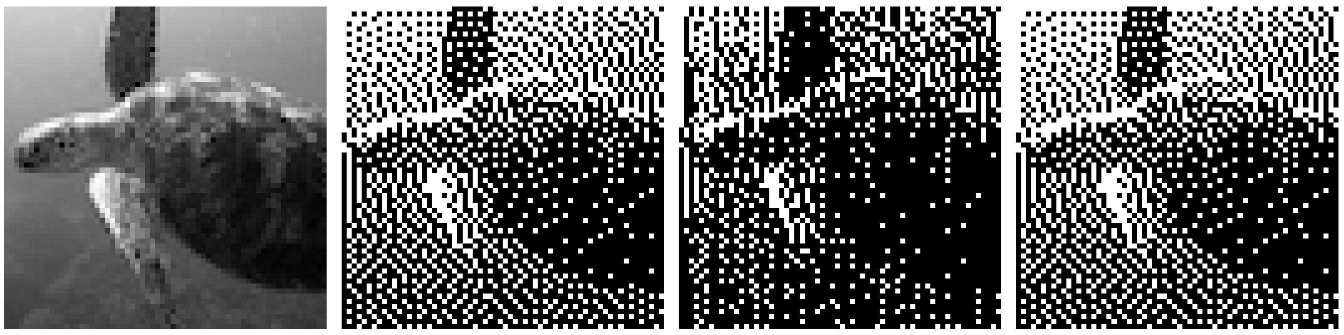
Use the MATLAB function

```
bits = load_image('images/turtle.tiff');
```

to load the image "turtle.tiff" from the `images` folder. The same function also displays the image and converts it into a black-white image and then a vector of bits. Note that this folder contains other images as well. We add a binary 1 to the beginning to simplify synchronization. The MATLAB function

```
show_bits(bits_detected);
```

takes the detected bit vector `bits_detected` and converts it to a black-white image.



(a) Grayscale image. (b) Black-white image. (c) Received image: 516 errors. (d) Received image: 0 errors.

Figure 26: Grayscale image (a) converted to black-white image (b) that can easily be transmitted over the air. Received image with 561 errors (c); received image without any transmission errors.

Figure 26 shows an example of over-the-air transmission of the turtle image. The first image shows the original 64×64 pixel image. The second image shows the black-white image that we will transmit. Each pixel of this image will be transmitted over the air. The third image shows a received image where 561 errors happened (out of the 4097 transmitted bits). This is an error rate of about 13% (obtained by dividing the number of errors by the number of transmitted bits). Interestingly, one can still see the turtle despite the errors. The last figure shows the result if no transmission happened.

Activity 43: What is the worst error rate?

An error rate of 0% indicates perfect transmission; an error rate of 50% means that every other bit is wrong (on average); an error rate of 100% indicates that all bits are transmitted in error. Ask yourself: What is worse, having 50% errors or 100% errors? Do not jump into quick conclusions!

Activity 44: Transmit an image over the air

Use the functions `load_image` and `show_bits` together with your MATLAB code to transmit an image over the air. Try to transmit an image without any errors. Feel free to transmit some of the other small images in the `images` folder. Increase the data rate of your system until errors happen. How many errors can you tolerate to still identify the original transmitted image?

Activity 45: Would video transmission be possible?

Now assume that you want to transmit a video. Videos are basically a sequence of images (or frames) that are recorded at a certain rate (known as the frame rate). Assume that you would like to transmit a 10 second black-white video of 64×64 pixels at a frame rate of 15 frames per second. How long would it take to transmit this video over your communication system? Would it be possible to watch the video in real-time (i.e., without downloading it first)?

10.6 Advanced Modulation and Demodulation Schemes

So far, you have learned how to build a rudimentary communication system and what the key components are to make it work. Practical systems (such as Wi-Fi, LTE, or Bluetooth) use significantly more advanced technologies and nowadays, AM is only used for terrestrial radio transmission (e.g., emergency broadcast). There exist many ways to improve your communication system. If your group made fast progress and if you are interested, we can explain to you some of the methods that you can try to implement to achieve higher data rates. Some of these techniques are summarized as follows:

- Map two bits to four distinct amplitudes. This can potentially double the data rate without increasing the bandwidth, but requires more sophisticated methods for detection.
- Use better filters at the receive side to block interference from other systems communicating nearby.
- Occupy multiple frequencies at once to transmit multiple bits in parallel. This can significantly increase the data rates but also occupies more bandwidth.
- In addition to using amplitude information to encode information, you can also use phase information. This requires complex receivers but increases the data rates without increasing bandwidth.
- Add redundancy to the transmitted bits to prevent errors. This approach is known as *coding* and can further improve robustness of your system.

Remember: In this research project, you have already learned the most important basics of signal processing and digital communication. A dedicated digital communications course would rigorously

explain the concepts behind the methods we have used here, and also cover some of the advanced topics discussed above. Cornell, for example, has the course ECE 4670 on Digital Communication System Design, where you would build a communication system that uses the same modulation strategy as Wi-Fi. As it turns out, the method is related to what you have seen here, but is able to achieve much higher data rates while being more robust to noise and able to tolerate multi-path propagation.