AC 2011-249: A LAB DEVELOPMENT FOR REAL TIME COMMUNICATION SYSTEMS WITH PCS

Min-Sung Koh, Eastern Washington University

MIN-SUNG KOH obtained his B.E. and M.S. in Control and Instrumentation Engineering in the University of ULSAN, South Korea, and his Ph. D in Electrical Engineering and Computer Engineering at Washington State University. He was with KEPCO (Korea Electric Power Co.) for 9 years before enrolling in the Ph. D. program at Washington State University. In KEPCO, he worked at the NPP (Nuclear Power Plant) as a nuclear engineer. In the Fall '02 quarter he joined the department of Engineering and Design at Eastern Washington University, where he has taught several courses in Computer Engineering Technology and Electrical Engineering. Currently, he is an associate professor of Electrical Engineering at Eastern Washington University. His research interests are in the areas of speech and image signal processing, signal processing in communication systems, photoacoustics, and embedded systems.

Esteban Rodriguez-Marek, Eastern Washington University

Prof. Rodriguez-Marek is an Associate Professor at Eastern Washington University. His research interests are in signal processing and engineering education.

Claudio Talarico, Eastern Washinton University

Claudio Talarico is an Associate Professor of Electrical Engineering at Eastern Washington University. Before joining Eastern Washington University, he worked at University of Arizona, University of Hawaii and in industry, where he held both engineering and management positions at Infineon Technologies, IKOS Systems (now Mentor Graphics), and Marconi Communications. His research interests include design methodologies for integrated circuits and systems with emphasis on system-level design, embedded systems, and complex SOCs. Talarico received a PhD in Electrical Engineering from the University of Hawaii at Manoa, and he is a member of IEEE. Contact him at ctalarico@ewu.edu

A Lab Development for Real Time Communication Systems With PCs

Abstract

Communication system classes have been traditionally taught with a lecture-only format. However, the proliferation of new concepts and algorithms in communication systems makes it increasingly hard for students to master them only through mathematical derivations. Furthermore, without a hands-on demonstration of how the algorithm is used in real-life applications, students without strong mathematical skills can become frustrated and generate a retention problem in EET/CET/EE programs. To overcome this problem, the theory taught in lecture has been complemented with laboratory experiments and class projects. However, many traditional communication systems' laboratory experiments are related to various hardware and software, many of which are not easily available outside of the lab environment. In this paper, we introduce a lab that allows students to experience real-time communication systems with a PC and the ubiquitous MATLAB package and, thus, can be done in the comfort of home. The lab allows the verification through experimentation of various communication system concepts in real time, processing signals obtained through the PC's sound card from input devices such as microphones, iPods, etc. This paper shows that transmitter/receiver models can be set up in real time with PCs. Further, it describes how this laboratory experiment can be extended with the use of a personal computer to many other labs or projects that require real-time processing.

1. Introduction

Our daily life is closely connected to communication devices such as iPods, smart phones, Netbooks, laptops, etc., only to name a few. The rapidly changing modern world has driven many people to base their social agendas on communication devices. These devices are so ubiquitous that is getting harder and harder to even imagine our lives without communication devices. Hence, as a communication systems engineer/technologist it is essential to have an understanding of communication systems' basic concepts and principles. To provide this understanding in the classroom, theoretical concepts are introduced in the typical communication systems class. Among the various topics, it is essential to learn modulation and demodulation of analog/digital communication systems because many other subsystems such as synchronization, encryption, and error correction, etc. can be implemented on top of basic modulation/ demodulation. These basic modulation/demodulation concepts have been introduced with mathematical derivations and their complexity has been frustrating for students not having a strong mathematical background, not an uncommon occurrence in EET/CET/EE programs. Furthermore, tedious mathematical derivations used to explain theories have led to retention

problems. Hence, it is essential complement theory with laboratory experiments, so that students get to visualize the concepts and can thus take more ownership of their learning. Many lab equipment for communication systems [1] are commercially available, and provides good practice to experiment first-hand the theory learned in lecture. However, these lab environments are not portable, as many benchtop devices are required. Thus, the student is confined to working in a school laboratory, thereby precluding easy access to learning instruments. In an attempt to overcome this problem, simplified communication labs have been attempted based on software/hardware combinations such as Labview with a DSP board [2], MATLAB/Simulink [5] or MATLAB/Simulink and/or C/C++ with a DSP board [3] [4]. In this paper, a new approach for simplified laboratory experiments for communication systems is introduced, which uses real-time processing capabilities of MATLAB and sound cards available in most PCs. Since the developed lab does not use any boards (except for sound card in a PC) or hardware, it can be completed anywhere, as long as the necessary MATLAB toolboxes have been installed. Note that a sound card is available in just about every modern personal computers. Students can not only experience traditional communication theory using their own laptop or desktop computers, but they can also build more advanced subsystems to go above and beyond the expectations for their class projects. The setup of a PC as a transmitter (Tx) and a receiver (Rx) are explained in Sections 2 and 3, respectively. A sample real-time Tx/Rx model using PCs is introduced in Section 4. Conclusions followed in Section 5.

2. Setup as a Transmitter

In this section, it will be explained how to setup a PC as a transmitter using MATLAB. As a necessary first step, MATLAB's "Data acquisition toolbox" [6] must be available in the PC, so that we can send and receive data through the sound card. Although the "Data Acquisition Toolbox" is not included in the student version of MATLAB, it can be downloaded as a trial version at the site given in [7] or it can be added on the student version of MATLAB for additional cost. Once the toolbox is installed in the PC, then the PC can be set up as a transmitter using the code shown in Figure 1.

****

```
clear all; close all; clc;

%----- Setup a Sound card in a PC-----
Fs=22050;
ai = analoginput('winsound',0);
chan1 = addchannel(ai, 1);
ao = analogoutput('winsound',0);
chan2 = addchannel(ao,1);
%----- Setup a sound card to read continuously ------
set(ai,'SampleRate',Fs);
set(ai,'SamplesPerTrigger',inf);
set(ao,'SampleRate',Fs);
%-----Sample a signal and send it continuously to a speaker ------------
for k=1:1:inf
    if k==1
       start(ai)
```

```
            end
            Data = getdata(ai,Fs);
            %-- Include your modulator here
            plot(Data);
            ModData = Data;
            %--Send it to a speaker
            putdata(ao,ModData);
            if k==1
               start(ao)
            end
            %--Comment out this part not to see event log for output
            events = ao.EventLog;
            {events.Type}
            %--End to see events.
        end
        delete(ai)
        clear ai
        delete(ao)
        clear ao
**************************************************************************
```

Figure 1. MATLAB codes to setup a PC as a transmitter

In Figure 1, notice that the built-in functions, "analoginput", "addchannel", etc. are included in the "Data Acquisition Toolbox". The sampling frequency in the example, *Fs*, is set as 22050 Hz but it can vary depending on the possible sampling frequencies allowed by the sound card. Built-in function "analoginput" creates an analog input object, "ai", for a sound card (i.e., adaptor input option is "winsound") with an identification number 0 (i.e., ID input option = 0). Similarly, an analog output object, "ao" is created. A hardware channel is added into the created analog objects, "ai" and "ao" by the built-in function "addchannel" as "chan1" and "chan2". Other parameters such as sampling rate and samples per trigger are also set in the code shown in Figure 1. In setting the "SamplesPerTrigger" parameter, "inf" implies that the data will be obtained continuously until a stop function issued or an error occurs [6]. The main parts of the transmitter are repeated as an infinite loop until the program of transmitter is forced to be stopped, or it can be iterated for a certain amount of time, if desired. In each iteration, the data is obtained from a sound card by the built-in function "getdata" and the obtained data will be modified by students' algorithms to be implemented. The modified data (i.e., ModData in Figure 1) can be transmitted through a wired line to another computer (i.e., a receiver) by putting data with the built-in function "putdata." Note that in Figure 1 ModData is identical to the data obtained from the sound card, as no algorithm is being tested yet. Notice that the "plot(Data)" command in Figure 1 is used to send data continuously to the sound card without "stop(ao)". Two command lines, "events=ao.EventLog;" and "{events.Type}" are used to print the events of output channel into the MATLAB command window for troubleshooting. The "Stop" message will be displayed on the command window of MATLAB if the output device is stopped for some reason (e.g. an error, etc.). The same process is repeated infinitely or for the desired time depending on the number of iterations. The rest of the code in Figure 1 is used to end the program properly after it has iterated the desired time. Students can put their own code in the spot indicated by "Include your modulator here." Although a modulator is called for, any other advanced subsystems such as

filter designs, encryption, error correction coding, synchronization, etc. can be used, depending on the goal of the experiment. An added benefit of this setup is that students have to take running time into consideration in the design of their systems, as a low-running system will suffer loss of data. A simple example of a student experiment is double sideband amplitude modulation (AM), where students can multiply a cosine (i.e., the carrier signal) with the input data to obtain the modulated signal. For the two-computer model, the computer set for transmission can have an iPod (or a microphone) connected into its mic input and the headphone output will have the transmitted data. This data should be connected into the mic input of the other computer designated as the receiver.

3. Setup as a Receiver

When two PCs are available, the second can be assigned as the receiver and must be connected to the transmitter through a wired line. A simplified version of the lab is to use a single computer having a transmitter and a receiver. However, it is a more proper setup to utilize two PCs as the transmitter/receiver model mimicking a real-world communication environment. The computer set as a receiver must have the code shown in Figure 2.

****

```
clear all; close all; clc;

%----- Setup a Sound card in a PC-----
Fs=44100;
ai = analoginput('winsound',0);
chan1 = addchannel(ai, 1);
ao = analogoutput('winsound',0);
chan2 = addchannel(ao,1);
%----- Setup a sound card to read continuously ------
set(ai,'SampleRate',Fs);
set(ai,'SamplesPerTrigger',inf);
set(ao,'SampleRate',Fs);
%-----Sample a signal and send it continuously to a speaker ------------
for k=1:1:inf
    if k==1
       start(ai)
    end
    Data = getdata(ai,Fs);
    %-- Insert demodulator here
    plot(Data);
    DeModData = Data;
    % Send it to a speaker
    putdata(ao,DeModData);
    if k==1
       start(ao)
    end
    %--Comment out this part not to see event log for output
    events = ao.EventLog;
    {events.Type}
    %--End to see events.
end
delete(ai)
clear ai
delete(ao)
clear ao
```

****

Figure 2. MATLAB codes to setup a PC as a receiver

Basically, the code in Figure 2 is the same as that in Figure 1 except for *Fs* and the application code, which should be written by the students. Students must implement the demodulator code in the section indicated by "Insert demodulator here". Note that any other advanced algorithms such as decryption, error corrections decoding, synchronization, etc. can also be included in this segment. The counterpart of the example described in Section 2 is a demodulator for double side band AM modulation, where students need to multiply cosine (i.e., the carrier signal with the same frequency as in the modulator) and path through a proper low pass filter to retain only the baseband signal. Even in the simple example of double sideband AM demodulator, students will clearly see why a filter design is required. Moreover, students will notice the time delays caused by filtering, hopefully previously explained in lecture or in the textbook. Students should understand how to deal with the time delays. When students implement their algorithms for each part in the transmitter and the receiver, they realize how the theory covered in lecture works in real-time and in a practical environment. Furthermore, they will be exposed to many practical issues such as delay, filtering, synchronization, etc. which must be considered carefully in practical environments. If the transmitter and the receiver are not designed correctly, students will be able to easily hear the distortion, loss of data, noise, etc. in real-time. Many interesting concepts can also be addressed by this experiment. For example, laptops connected to the grid may display audible distortion due to the frequency of the distribution system. This noise does not show up in a laptop running on battery power.

4. Real-Time Tx/Rx Example

The double sideband AM modulation/demodulation is further explored in this section. Using the basic setup given in Figures 1 and 2, students add their own codes for the theory of AM modulation where indicated in Sections 2 and 3. Figure 3 shows the block diagrams of AM.

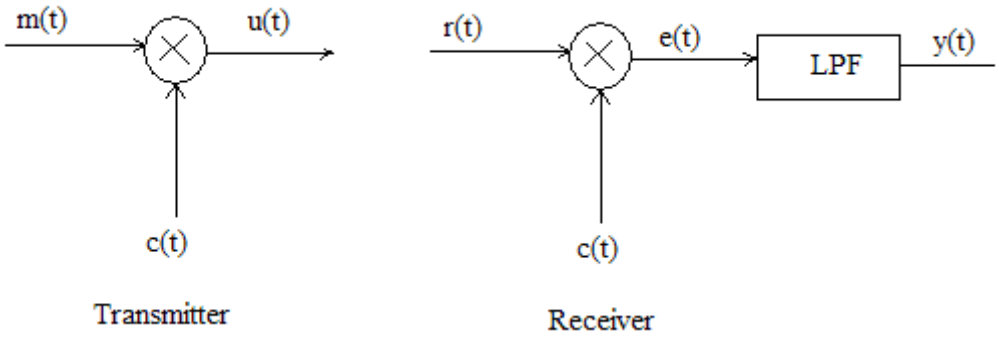

Figure 3. A simple example of real-time processing in communication systems

Students need to design a proper lowpass filter denoted by LPF to extract the baseband signal in real-time. And they should pass the $e(t)$ signal through the obtained LPF for filtering process, which can be done by the convolution operation in MATLAB. Even with this simple example, students can observe all processes in real-time using their input device such as iPod or MP3 players, etc. To check the theory without any transmitting delay, the system in Figure 3 is implemented in a single computer. Through a single computer model of Figure 3, students can check the theory of double sideband AM modulation without being confused by any transmission delay. The spectrums for $m(t)$, $u(t)$, $e(t)$, and $y(t)$ implemented in a single computer, are shown in Figures 4 and 5 for voice and music, respectively. Figures 4 and 5 are obtained by an arbitrarily specified data frame in real-time processing of voice and a song from iPod Shuffle. Note that, in order to have the same results using the two-computers model in Figures 1 and 2, students need to address synchronization, in order to have same data frame. Otherwise, data used in transmission and reception will differ, because of transmission and processing time delays. Like in a purely hardware approach using a spectrum analyzer, the spectrums of $u(t)$, $e(t)$, and $y(t)$ can be shown in real-time as well using the diverse built-in functions of MATLAB such as "fft", "fftshift", "hamming", etc. These spectrums are the ones shown in Figures 4 and 5 as a snapshot of a data frame with iPod shuffle as an input. Again, if a system is not designed correctly, students can hear significant distortion easily. Also, they can recognize losing data if their algorithm is too computationally intensive.

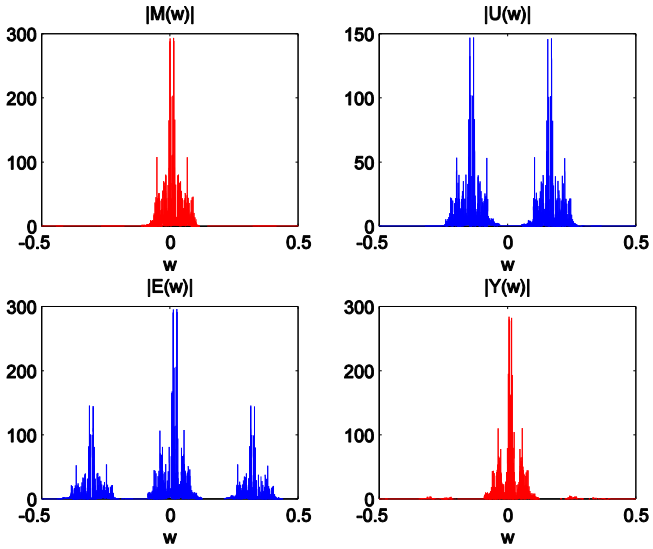

Figure 4. Spectrums of voice shown in real-time for the system given in Figure 3

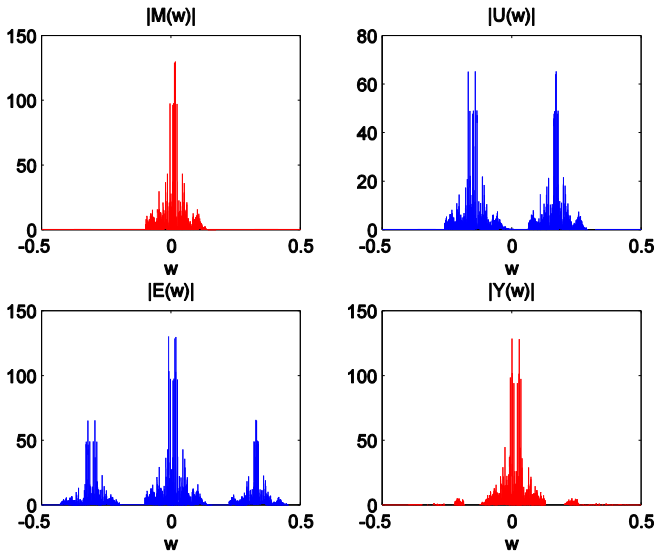

Figure 5. Spectrums of a song shown in real-time for the system given in Figure 3

The simplified real-time lab environment given in this paper can be performed anywhere, such as in the students' home or classroom without any other requiring traditional equipment such as spectrum analyzer, function generator, oscilloscope or without a particular equipment/board for communication systems and/or digital signal processing system in [1]–[4]. Furthermore, note that students can extend the experiment by including additional subsystems on top of the basic modulation/demodulation portion. Also note that students can test their application algorithms in a real-time environment. This lab setup can be incorporated into other classes related to real-time processing and/or class projects. The developed lab was used as one lab at Eastern Washington University for all students in a communication class to introduce real-time processing. One group extended the lab as a class project both for double sideband and single sideband AM, maintaining similar overall basic structure with two computers. To increase the complexity of the project, two signals were combined to create a stereo signal.

5. Conclusions

In this paper, a simplified real-time processing lab for communication systems is developed using a PC sound card and MATLAB. The developed real-time processing lab needs only the Data Acquisition Toolbox of MATLAB and one or two computers that have a sound card. Without any expensive traditional hardware such as communication experimental boards, spectrum analyzer, multimeters, etc., students can verify the various concepts covered in lecture at home or any other convenient location. Since there are many issues to be dealt in real-time processing, it helps students to better understand theory and to have exposure to problem-solving

skills that will typically be required in the workplace. The introduced lab can be extended into any other labs requiring real-time processing.

References

[1] Analog communications instructional module, http://www.labvolt.com/downloads/datasheet/dsa9410.pdf, Lab-Volt.

[2] A. Uluagac and D. Williams, "Building Hardware-based low-cost experimental DSP modules", in the *Proceedings of the American Society for Engineering Education (ASEE) Annual Conference & Exposition*, 2008

[3] G. W.P. York, C. M. Rondeau, and D. F. Fuller, "Teaching Real-time DSP applications (Voice Removal) with the C6711 DSK and MATLAB", in the *Proceedings of the American Society for Engineering Education (ASEE) Annual Conference & Exposition*, 2004.

[4] R. J. Kozick, "Undergraduate Design Projects in a Laboratory for Real-Time Signal Processing and Control", in the *Proceedings of the American Society for Engineering Education (ASEE) Annual Conference & Exposition*, 1996.

[5] Communication Blockset, http://www.mathworks.com/products/commblockset/description2.html , Mathworks.

[6] Data Acquisition Toolbox, http://www.mathworks.com/products/daq/ , Mathworks.

[7] Trial software, http://www.mathworks.com/products/daq/tryit.html , Mathworks.