Massachusetts Institute of Technology
Dept. of Electrical Engineering and Computer Science
Fall Semester, 2006
6.082 Introduction to EECS 2

Lab #1: Matlab and Control of PC Hardware

Goal:

Basic training on Matlab and the control of PC hardware and the oscilloscope. Exercises include using Matlab to create and plot signals, sending those signals to the PC headphones, the USRP board, and oscilloscope. In addition, voice signals will be brought into Matlab from the microphone and then sent to the USRP board and oscilloscope.

Instructions:

1. Complete the activities for Wednesday's lab (see below) and get checked off by one of the TAs before leaving. Be sure to work in pairs with one computer per pair.

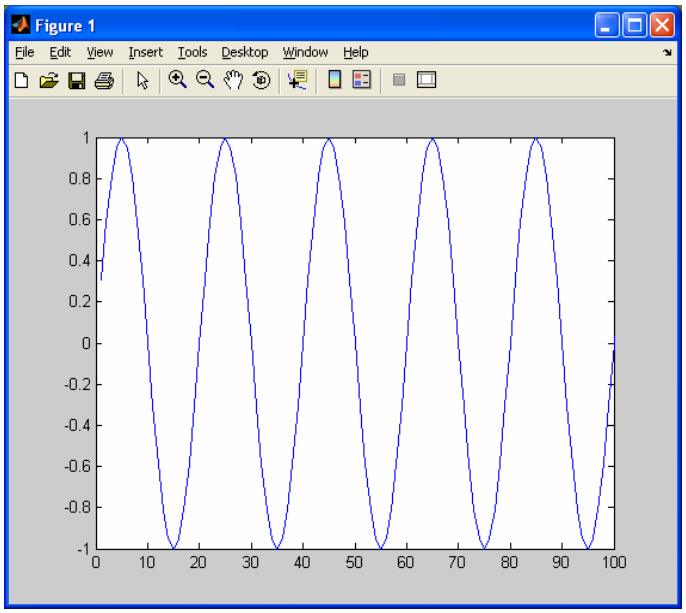
Lab Exercises (2pm – 5pm, Wed., September 6, 2006):

A. Starting Matlab and Performing Basic Plotting Operations

- Choose a lab partner and a PC to work on.
- On the PC, double-click on the Matlab icon in order to start Matlab.
- Within the Matlab execution window, type the following commands:

```
t = 1:10000;
y = sin(2*pi*1/20*t);
plot(t,y);
axis([0 100 -1 1]);
```

  - You should now see a plot of a sine wave on the screen as follows:



- We briefly explain the above commands:
  - **t = 1:10000**;
    - This creates a vector, **t,** whose entries span from 1 to 10000 in increments of 1. In other words, **t** = [1 2 3 ….. 9999 10000]
  - **y = sin(2*pi*1/20*t);**
    - This creates a vector, **y**, which consists of a sine wave evaluated at each value of **t** multiplied by **2*pi*1/20** so that **y** = [sin(2*pi*1/20) sin(2*pi*2/20) .... sin(2*pi*10000/20)]
  - **plot(t,y);**
    - This plots the entries of **y** versus those in **t**.
  - **axis([0 100 -1 1]);**
    - This limits the axis of the plot to **t** = 0 to 100 and **y** = -1 to 1.
- To learn more about a given Matlab command, one can make use of the **help** function within Matlab. As an example, run the following command in Matlab to get more information about the **axis** command:

  help axis

B. Running Matlab Commands From a ".m File"

Although it is easy to run commands directly from the execution window in Matlab, it will get tedious when you want to run those commands repeatedly. Therefore, we now explain how to place such commands within a Matlab .m file and then run the file from Matlab.

- Within Matlab, type the following commands:

```
cd Lab1
edit lab1_script.m
```

- Be sure to answer **yes** when prompted for creation of a new file.

- Within the new edit window, type in the same commands you entered above and then save the file:

```
t = 1:10000;
y = sin(2*pi*1/20*t);
plot(t,y);
axis([0 100 -1 1]);
```

- Within Matlab, execute the above commands by typing:

```
lab1_script
```

- Figure 1 should show the same plot as shown above.

C. Sending a Matlab Signal to the PC Headphones

- Now add the following command to the end of your **lab1_script.m** file, save the file, and then type **lab1_script** in Matlab to execute:

```
t = 1:10000;
y = sin(2*pi*1/20*t);
plot(t,y);
axis([0 100 -1 1]);
soundsc(y)
```

- You should hear a tone from the PC headphone that lasts a little more than one second.. This tone corresponds to the sine wave vector, **y**, that we created above. The frequency of the sine wave was set as **2*pi*1/20**, and the time duration of the sine wave was set by the length of **y**. Note that the length of **y** was set to be 10000 samples as specified by the command **t=1:10000**.

- Let us now change the duration of the sound by increasing the length of **t**. Modify the **lab1_script.m** file as shown below, save it, and then again execute in Matlab:

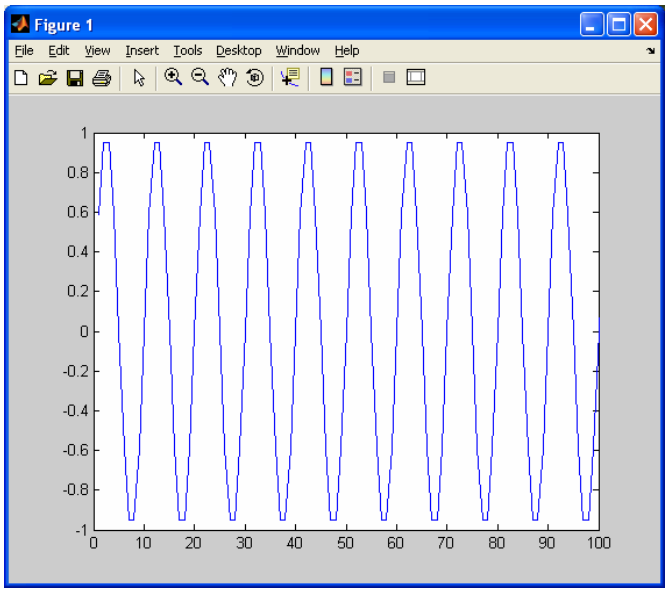
```
t = 1:20000;
y = sin(2*pi*1/20*t);
plot(t,y);
axis([0 100 -1 1]);
soundsc(y)
```

- You should notice the tone lasts twice as long now.

- Let us now change the frequency of the tone by the modification shown below. Be sure to examine the **Figure 1 plot** before and after you run this updated script:

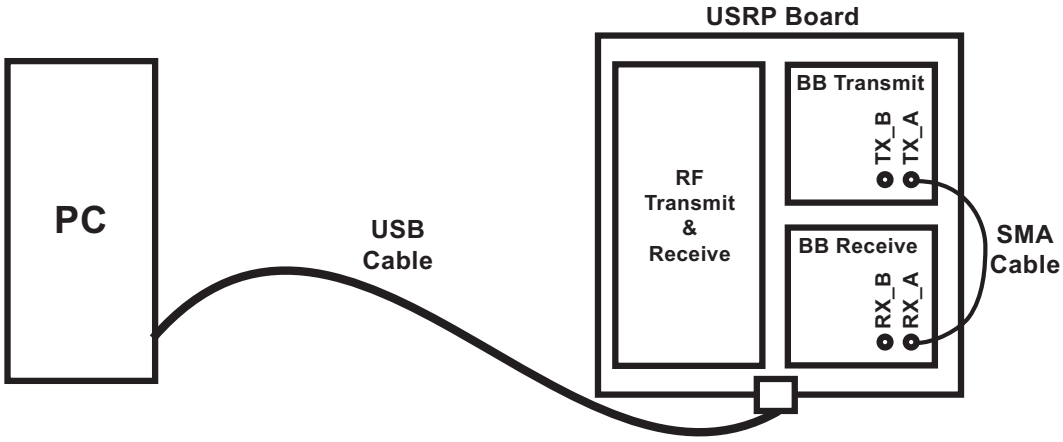
```
t = 1:20000;
y = sin(2*pi*1/10*t);
plot(t,y);
axis([0 100 -1 1]);
soundsc(y)
```

  - The tone should sound a higher pitch, and the plot should show that the sine wave now has a higher frequency. The plot also reveals that the sine wave no longer looks as ideal as before, as shown below. This is an artifact of *sampling* that we will discuss later in this course.



D. Sending and Retrieving Matlab Signals to/from the USRP board

Now that we have created Matlab signals and sent them into the PC headphones, we turn our attention to sending those signals through the USRP board. In particular, we will send the sine wave signal created above through the USB cable and then into the baseband (BB) Transmit USRP board. An SMA cable attached to the transmit output, **TX_A**, then routes that signal into the **RX_A** input of the baseband (BB) receive USRP board. The retrieved signal is then sent back through the USB cable and then returned as the output of a Matlab function shown below. The figure below shows the relevant hardware in this experiment.

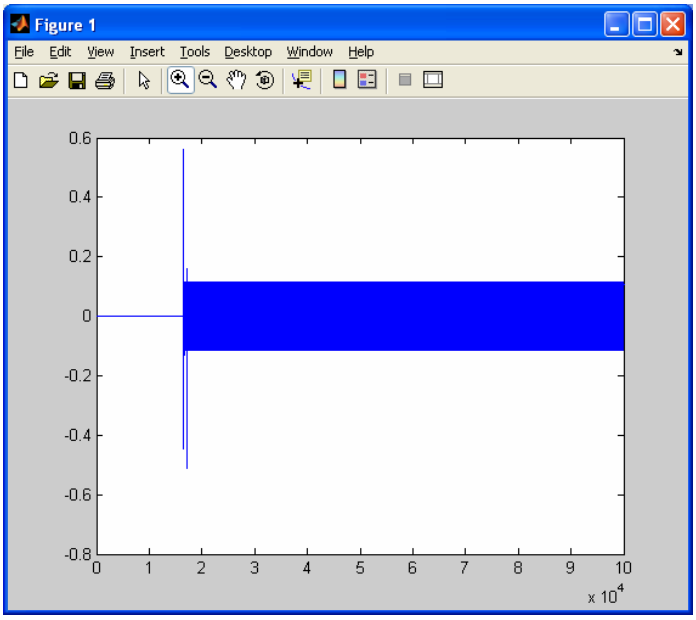


- Now modify and save your **lab1_script.m** file as follows:

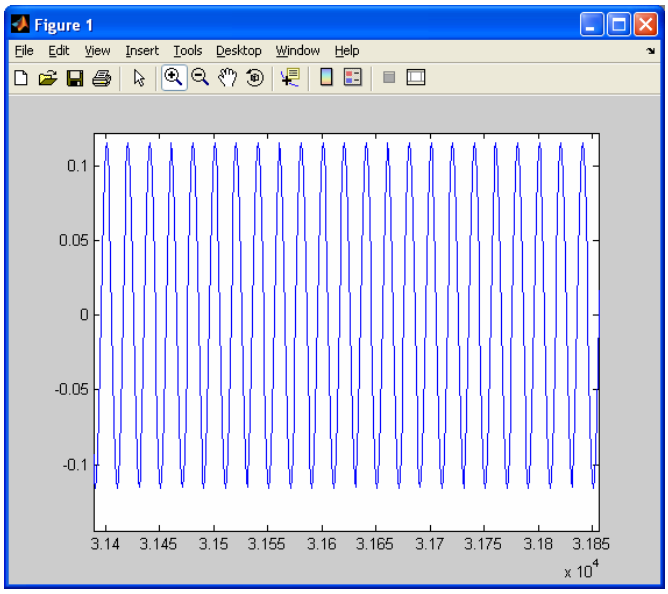
```
t = 1:1e5;
y = sin(2*pi*1/20*t);
% plot(t,y);
% axis([0 100 -1 1]);
% soundsc(y)

system('initialize_usrp')
usrp_bb_trans_basic(y,0*y);
[rx_a,rx_b] = usrp_bb_receive_basic(length(y));
plot(t,rx_a);
usrp_bb_trans_basic('end');
usrp_bb_receive_basic('end');
```

  - Execute the above script. You should obtain a plot that looks similar to the one shown below. Unfortunately, the BB receive board is a bit inconsistent at this point in time, and so you may need to rerun the above script several times to see something similar to what is shown below.



- A few observations are in order
  - Notice that the above signal is zero initially and then suddenly springs to life. This behavior is due to the fact that there is a delay from the time that Matlab sends the transmit signal to when it, in turn, receives it back. The delay is dominated by the roundtrip time that it takes to get the signal through the USB interface of the PC.
  - You can use the hourglass icon (with a **plus** sign within it) inside the **Figure 1 plot window** to zoom in on the signal. In doing so, you should something like what is shown below, which confirms that the sine wave signal was successfully sent through the USB cable and USRP board interfaces.



- In effect, we have demonstrated *communication* through a given *channel* in this example. In this case, the signal communicated is the sine wave within Matlab, and the communication channel consists of the USB cable, USRP circuits, and the SMA cable. More generally, this type of channel is referred to as a *wired* channel. We see that two impacts of the wired channel in this example are that the sine wave signal is delayed and attenuated. In a few labs from now, we will explore sending signals through a *wireless* channel.

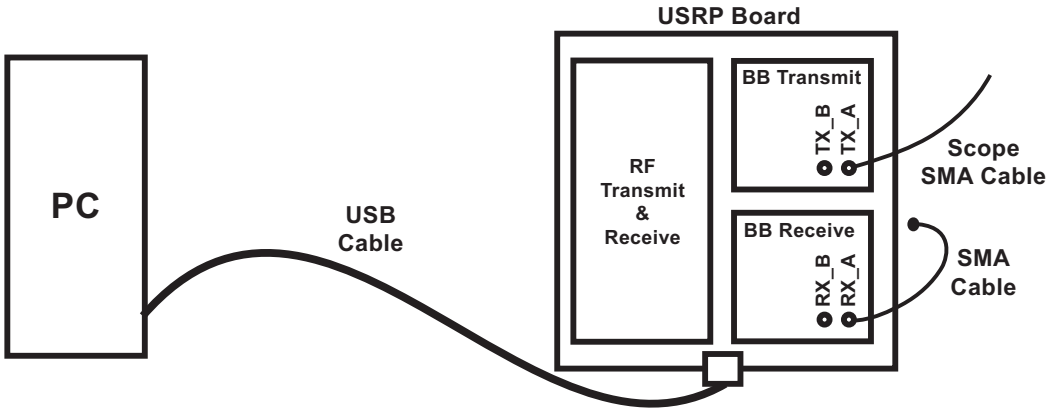
- We should also explain more about the new commands that were used in the above script:

  - **% plot(t,y)**
    - The **%** symbol is used to comment out a given line so that it is not executed
  - **system('initialize_usrp')**
    - This command initializes the USRP board. It actually only needs to be run only once after initial power up of the USRP board, but it doesn't hurt to run it repeatedly as done here.
  - **usrp_bb_trans_basic(y,0*y);**
    - This command sends the signals **y** and **0*y** to the **TX_A** and **TX_B** SMA plugs on the USRP BB transmit board. In this case, **TX_B** is simply a zero-valued signal, so one might wonder why it is specified as **0*y**. The reason is that both signals must have the same length, and the length of **0*y** matches the length of **y**.
  - **[rx_a,rx_b] = usrp_bb_receive_basic(length(y));**
    - This command retrieves the signals coming into the **RX_A** and **RX_B** SMA plugs on the USRP BB receive board, and returns them as signal variables **rx_a** and **rx_b** within Matlab.
  - **usrp_bb_trans_basic('end');**
    usrp_bb_receive_basic('end');
    - These commands clear out memory used by software calls that interface to the USRP board. Although the above example would run without this extra step, we include them for completeness.

E. Probing Signals with an Oscilloscope and Executing For Loops in Matlab

Thus far we have confined our examination of signals to the Matlab framework. In many real world situations, we often want to *probe* signals using dedicated measurement instruments called oscilloscopes. These instruments avoid the extra delay issues we saw with the USB interface in our previous example, and allow for more interactive examination of signals occurring within a given hardware system.

- Unscrew the SMA cable from the **TX_A** SMA plug as shown below. In turn, as also shown, screw on the oscilloscope SMA cable onto the **TX_A** plug.



- Be sure to set up the oscilloscope according to the Oscilloscope document passed out with this lab.
- Now execute the most recent **lab1_script** from the previous section. Examination of the oscilloscope should reveal a waveform that briefly appears and then disappears. The brief appearance is due to the finite length of the sine wave signal that we are sending into the USRP board.
- Note that the **Figure 1 plot** no longer shows the received sine wave waveform as before. This is due to the fact that the SMA cable was removed, so that the wireline channel was broken. Note that you will see a non-zero waveform in the **Figure 1 plot** due to the fact that *noise* is present in the USRP BB receiver board (the amplitude of this noise is much smaller than the sine wave waveform you previously observed). In fact, this noise was present in the previous plots, but was too small to notice compared to the sine wave signals we were observing. We'll talk more about noise later in this class.
- It would be easier to view the signal on the oscilloscope if we extended the length of the sine wave signal we are sending into the USRP board. To so do, now modify and save your **lab1_script.m** file as follows:

```
t = 1:1e5;
y = sin(2*pi*1/20*t);
% plot(t,y);
% axis([0 100 -1 1]);
% soundsc(y)

system('initialize_usrp')

for i = 1:100
    usrp_bb_trans_basic(y,0*y);
% [rx_a,rx_b] = usrp_bb_receive_basic(length(y));
% plot(t,rx_a);
end
```

```
usrp_bb_trans_basic('end');
% usrp_bb_receive_basic('end');
```

- The above change implements what is called a **for** loop which cycles from 1 to 100 in steps of 1. During each **for** loop cycle, the value of variable **i** is updated to reflect the current cycle number (i.e. from 1 to 100). In this case, we don't make use of the variable **i**, but in many future examples we will do so. Note that the **for** loop requires an **end** statement that indicates the scope of commands to be repeated – in particular, all statements between the **for** statement and **end** statement are considered to be within the **for** loop.

- Execute the updated **lab1_script** within Matlab while probing the **TX_A** signal with the oscilloscope. The signal should now remain in view much longer, which enables you to play with the horizontal and vertical controls of the oscilloscope. Using the oscilloscope controls, record the time period of the sine wave signal as well as its peak-to-peak amplitude on the last sheet of this handout. Be sure to include units in your answers.

F. Recording Audio Signals using the Microphone and Matlab

Thus far we have dealt signals that were created within Matlab and then transported to other hardware devices such as the PC headphones, USRP board, and the oscilloscope. We now consider bringing signals *into* Matlab that are generated by a real-world system, namely your voice chords!

- Within the Matlab execution Window, type:

```
edit lab1_mic_script.m
```

  - Be sure to answer **yes** when prompted for creation of a new file.
- Within the new edit window, type in the following commands and then save the file:

```
r = audiorecorder();
recordblocking(r,3);
play(r);
y = getaudiodata(r);
plot(y);
```

  - **Explaining each command:**
    - **r = audiorecorder();**
      - Creates a recording object named **r**. To get more information about the **audiorecorder** function, type **help audiorecorder** in Matlab.
    - **recordblocking(r,3);**

        - Records from the microphone for 3 seconds and places the information within object **r**.
    - **play(r);**
        - Sends the microphone waveform stored in object **r** to the PC headphones.
    - **y = getaudiodata(r);**
        - Grabs the microphone waveform from object **r** and stores in the Matlab variable **y**, which can then be plot.
    - **plot(y);**
        - Plots the recorded signal.

- Within the Matlab execution window, run the **lab1_mic_script** created above.
    - As the script runs, speak into the microphone and listen to the headphones. When the recording portion of the script has completed, you should hear your voice played back through the headphones.
    - Examine the **Figure 1 plot** when the script completes. You are seeing the waveform that was produced by the microphone and which was sent to the headphones for playback.
    - Run the **lab1_mic_script** several more times to see how the plotted waveform changes with different sounds. Be sure to not be too loud as you record in order to keep the noise levels at a reasonable level within lab.

G. Using the Oscilloscope to View the Microphone Waveform

As we conclude Lab 1, our last exercise will be a bit more open-ended in order to give you the chance to explore Matlab, its connection to the USRP board, and use of the oscilloscope. The goal in this section is to send the microphone generated waveform to the **TX_A** SMA plug of the USRP BB transmit board, and then to examine the signal by probing **TX_A** with the oscilloscope. We now provide some hints of how to do this. Once you have completed this exercise, show your results to a TA in lab in order to get checked off for this lab.

Here are your hints:

- Your goal is to see a waveform on the oscilloscope that is similar to what you see in the Matlab plot produced in the previous section. In other words, you are to record into the microphone, and then take that waveform and send it to the USRP board. Proper use of the oscilloscope should allow you to see this waveform by probing the **TX_A** SMA plug.
- There is a catch in the above exercise – if you simply send the waveform once to the USRP, it will appear and then disappear on the scope too fast for your viewing. So, in order to make the waveform visible for a reasonable period of time, you need to loop the sending of the waveform to the USRP board by using a **for** statement in Matlab.
- The other catch is that you need to properly set the controls of the oscilloscope in order to clearly see the waveform. We suggest that you play around with the

oscilloscope controls once you are confident that the waveform is being repeatedly sent to the **TX_A** SMA plug.

- If you like, you could also route the **TX_A** signal into the **RX_A** SMA plug of the USRP BB receive board, and then view the received waveform in Matlab. This exercise is optional.

Upon completion of this exercise, please see a TA and have them sign off your ability to complete this exercise on the next page.

Check-off for Lab 1

_____ _____

Student Name Partner Name

Answer for oscilloscope measurement within Section E (with units indicated):

Sine wave period: _____

Sine wave amplitude (peak-to-peak): _____

Check-off for viewing the microphone waveform with the oscilloscope within Section G:

_____

TA Signature