EECE 2150 - Circuits and Signals: Biomedical Applications

Lab 8

Getting started with Analog to Digital Conversion and Sampling

In this lab we will use an Analog to Digital Converter (also called an A/D, ADC, or DAQ) to get signals from the analog ("real") world (continuous time and amplitude) onto a computer (discrete in time and quantized in amplitude). The particular A/D device we will use is made by National Instruments (NI) and can be controlled from within Matlab.

Part 1. Basic setup of the NI USB-DAQ

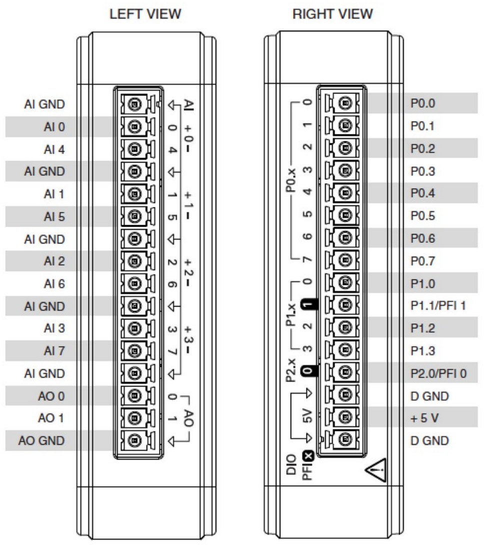

Figure 1. The NI USB-6001"Pinout" (see spec sheet online)

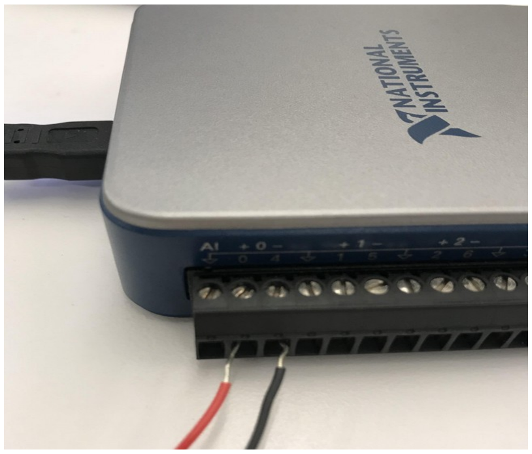

Figure 2. Connecting Wires to the DAQ Board

1.0 We can connect input signals into the NI-DAQ using wires with stripped ends. Insert the wire end into the first Analog Input terminals “AI 0 + and -, also indicated as 0 and 4”. You will need to tighten the connectors “snuggly/gently” with a screwdriver. For now, do not connect the other end of the wires to anything.

1.1 Plug in the DAQ to your computer with the USB cable. You should see a blue light on the top of the board. Open Matlab. We will use a “session based DAQ interface” for this lab.

You can find documentation on the Matlab website which **you can review** here:

http://www.mathworks.com/help/daq/examples/getting-started-with-session-based-interface-using-ni-devices.html#responsive_offcanvas

http://www.mathworks.com/help/daq/acquire-data.html

1.2 In this lab we will go through the basic steps of setting up an acquisition (blue text indicates what you type at the command line, purple text indicates the response from Matlab).

First, you can see what DAQ devices are available by typing:

```
>> daq.getDevices
```

at the command line. If it is connected properly you will see something like the following (after a pause of a few seconds). If not, ***stop*** and make sure everything is hooked up correctly

```
ans =

ni: National Instruments USB-6001 (Device ID: 'Dev1')
   Analog input subsystem supports:
      -10 to +10 Volts range
      Rates from 0.1 to 20000.0 scans/sec
      8 channels ('ai0' - 'ai7')
      'Voltage' measurement type

   Analog output subsystem supports:
      -10 to +10 Volts range
      Rates from 0.1 to 5000.0 scans/sec
      2 channels ('ao0','ao1')
      'Voltage' measurement type

   Digital subsystem supports:
      13 channels ('port0/line0' - 'port2/line0')
      'InputOnly','OutputOnly','Bidirectional' measurement types
   Counter input subsystem supports:
      1 channel ('ctr0')
      'EdgeCount' measurement type
```

1.3 Next, we need to create a "DAQ session" by typing:

```
>> s = daq.createSession('ni')
```

Matlab will return something like the following:

```
s =

Data acquisition session using National Instruments hardware:
   Will run for 1 second (1000 scans) at 1000 scans/second.
   No channels have been added
```

1.4 Next, we need to add an analog data input channel. For this we use the "addAnalogInputChannel" command. The inputs correspond to the DAQ session (s, see 1.3), the device name (Dev1, see 1.2), the input channel (ai0), and the measurement type (Voltage). Note that you device name may be different from Dev1, use the device name returned by MATLAB in 1.2:

```
>>  ch = addAnalogInputChannel(s, 'Dev1', 'ai0', 'Voltage')

ch =

Data acquisition analog input voltage channel 'ai0' on device 'Dev1':
       Coupling: DC
 TerminalConfig: Differential
          Range: -10 to +10 Volts
           Name: ''
             ID: 'ai0'
         Device: [1x1 daq.ni.DeviceInfo]
MeasurementType: 'Voltage'
```

1.6 You can see more DAQ parameters by looking at the subsystems using:

```
>> sub = ch.Device.Subsystems
```

Matlab will return the following:

```
sub =

Analog input subsystem supports:
   -10 to +10 Volts range
   Rates from 0.1 to 20000.0 scans/sec
   8 channels ('ai0' - 'ai7')
   'Voltage' measurement type
Properties, Methods, Events

Analog output subsystem supports:
   -10 to +10 Volts range
   Rates from 0.1 to 5000.0 scans/sec
   2 channels ('ao0','ao1')
   'Voltage' measurement type
Properties, Methods, Events

Digital subsystem supports:
   13 channels ('port0/line0' - 'port2/line0')
   'InputOnly','OutputOnly','Bidirectional' measurement types
Properties, Methods, Events

Counter input subsystem supports:
   1 channel ('ctr0')
   'EdgeCount' measurement type
Properties, Methods, Events
```

PART 2. INPUTTING AND ACQUIRING A TEST SIGNAL.

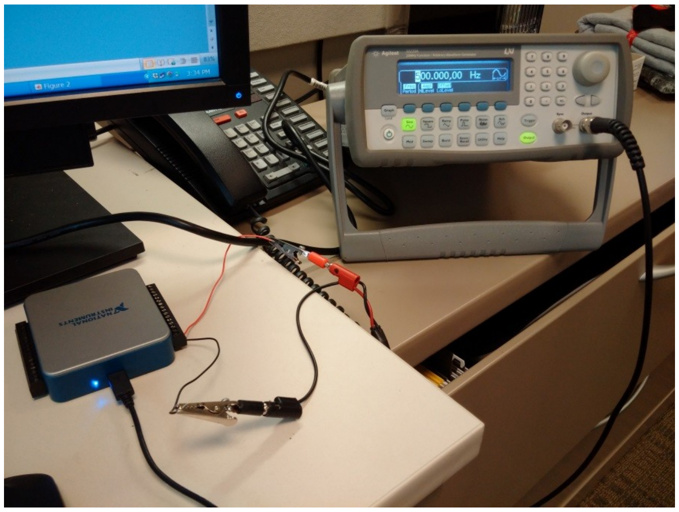

Figure 3. Connecting the Function Generator to the DAQ Board

2.1 Connect the function generator to the USB-DAQ with a BNC to banana connector cable and some alligator clips, similar to figure 3 above.

2.2 Now set up the function generator to generate an output wave as follows:
Sine wave, frequency 500 Hz, Amplitude 2V, DC Offset 0V, Output enabled (on).
Use your BNC "T" to observe the function generator signal on the oscilloscope while it is also going into the A/D converter.
2.3 We can set up the parameters of our acquisition as follows.
Let's assume we want a sampling frequency Fs of $10{,}000\,Hz$ (samples per second), and we want to acquire $0.01\,s$ of data $(100\,samples)$.
We can do this by setting the DAQ input channel acquisition rate:

```
>> s.Rate = 10000
```

Matlab will return:

```
s.Rate =

   Rate: 10000
```

And the number of samples (which Matlab calls "Scans"):

```
>> s.NumberOfScans = 100
```

Matlab will return:

```
s.NumberOfScans =

Data acquisition session using National Instruments hardware:
  Will run for 100 scans (0.01 seconds) at 10000 scans/second.
  Number of channels: 1
    index Type Device Channel   MeasurementType       Range       Name
    ----- ---- ------ ------- ------------------- ---------------- ----
    1     ai   Dev1   ai0     Voltage (Differential) -10 to +10 Volts
```

2.4 Now we can acquire the data using the “startForeground” command as follows:

```
>> [data,time] = s.startForeground();
```

This will read 100 samples into the variable ‘data’ at a rate of $10000\,samples/s$. The variable ‘time’ represents the time axis in seconds. We can check that this worked by plotting the data:

```
>>figure; stairs(time,data, '.-')
>>xlabel('Time (s)');
>>ylabel('Input Signal (V)');
>>title('500 Hz Sinewave, 5V Ampl, 0V DC Offset')
```

Here, xlabel, ylabel and title label the figure and axes. Calling plot with 'r.-' modifier gives us a red line (r), with points marked by dots (.) and connected with a line (-).

Part 3. Limitations of digitization: Quantization error.

Use the following function generator (FG) and analog to digital converter (ADC; Matlab) settings. Keep using the BNC “T” so you can have the output go both the oscilloscope and the A/D converter so that you can observe both the analog and digitized signals.

For each configuration, acquire and plot the data in Matlab (print a copy of your figure to put in your notebook). Remember to generate proper time-axes and labels as above. For each case, ***i) show the plot, ii) explain if the data does or does not properly represent the input sine wave and, iii) explain the cause of the problem, and iv) explain (with help from the instructors, if necessary) how the problem could have been avoided***.

Config1	*FG:*	*Sine wave*
		Amplitude = 100mV (center-to-peak)
		Frequency = 200 Hz
		DC Offset = 0V
	ADC:	*Sampling rate = 1500 Samples / s*

Acq time = 0.01 seconds

Config2	*FG:*	*Sine wave*
		Amplitude = 10mV (center-to-peak)
		Frequency = 200 Hz
		DC Offset = 0 V
	ADC:	*Sampling rate = 10,000 Samples / s*
		Acq time = 0.01 seconds

Voltage Resolution of A/D:

The NI USB6001 is a 14-bit A/D converter that has an input range from -10 V to + 10 V. However, the digitized value can only take on a finite number of values (or in other words, the voltage output is quantized). How many distinct values *N* can a 14-bit register hold? What is the voltage difference between each value $\Delta V = ?$ Does this explain the observations in above? Do you observe anything that cannot be explained by the quantization of the signal in the A/D process?

Part 4. Sampling music or speech.

Use your computer or phone to output an audio signal. Observe this signal on the oscilloscope (try to make the signal as large as possible – 1 to 2 volts p-p is good, somewhat less is ok). Use MATLAB and your ADC to obtain 3 audio samples, one at a 20kHz sampling rate, one at a 5 kHz sampling rate, and one at a 1 kHz sampling rate. Choose the number of samples so that each data set corresponds to a few seconds of sound. You can play back the samples using the MATLAB commands (assuming you name the data from the three scans data20k, data5k, and data1k):

```
>> soundsc(data20k, 20000);

>> soundsc(data5k, 5000);

>> soundsc(data1k, 1000);
```

What do the different samples sound like, compared with the original and with each other. Try to describe them more precisely than just good and bad – what do you hear? Why is this happening? Discuss with your instructors. (You can make a MATLAB script m-file to do all these sequentially for easier comparison)

Part 5. If you have time: A/D Sampling Rate – Deeper Investigation

a) Sampling Interval and Signal Frequency

Clearly, if you sample too slowly you will not be able to record a signal of frequency *f*. To explore this further, set your function generator at a frequency $f = 1000\,Hz$ with a center-to-peak amplitude of $4\,V$. The function generator should be set to high Z load. Collect a 2 second sample of the signal – using **`s.DurationInSeconds=2`** and **`[data,time]=s.startForeground()`**commands at the following sampling frequencies: F_s = 20000, 10000, 4000, 2500, 1800, 1600, 1400 and 1200 samples/s. After collecting each sample, use your computer speakers and the MATLAB `soundsc(data, Fs)` command to play back each sample. (You can make a MATLAB script m-file to do all these sequentially for easier comparisons!) Is there some point where you think you are no longer hearing a signal at 1000 Hz?

If you have time, change the function generator to produce a signal at $f = 2000\,Hz$, sample it at F_s = 20000, 10000, 4000, 3600, 3200, 2800 and 2400 Hz, and play out the sound using `soundsc(data, Fs)`.

Can you generalize to say what the maximum frequency f_{max} is that can be accurately digitized given a sampling frequency F_s? If you sample at a frequency above the maximum frequency, what happens to the signal you hear? What is the maximum frequency you could accurately digitize at $F_s = 20000\,Hz$, $F_s = 5000\,Hz$, and $F_s = 1000\,Hz$? What happens to signals at frequencies above these maximum frequencies? Does this help explain why the sound files you sampled were so distorted at lower sampling frequencies?

DO NOT FORGET TO HAVE THE INSTRUCTOR OR ONE OF THE TAs SIGN OFF ON YOUR NOTEBOOK WHEN YOU ARE DONE.

Department of Electrical Engineering, Northeastern University.
Last updated: 10/11/16, N. McGruer; 9/29/15, DH Brooks, 9/29/15, S W McKnight; 8/10/15, M.Niedre

McKnight; 8/10/15, M.Niedre