BME 194: Applied Circuits Lab 7: Sampling

Kevin Karplus

February 9, 2013

1 Design Goal

This lab is more of a demo than a design lab. The main point is to look at sampling frequency and how it affects the representation of signals stored on computers.

You will have to design a couple of RC filters in order to do the demo: a high-pass filter to block the DC component of the signal generator and shift the voltage up to 2.5V, and a low-pass filter to look at the effects of low-pass filtering on signals.

2 Background

When we record signals on the computer, we cannot store infinitely many data points to represent continuous functions of time. Instead we store data at fixed time intervals (the *sampling period*). Because it is often easier to deal with frequencies rather than times, we refer to the inverse of the sampling period as the *sampling frequency*, f_s.

If our signals are low frequency compared to our sampling frequency, the computer representation of them is quite good, and little, if any, information is lost in storing them at discrete times. There is a theorem that signals that contain no energy above $0.5f_s$ can be accurately reconstructed from the sampled data—the frequency $0.5f_s$ is referred to as the *Nyquist frequency* or the *Nyquist limit*. (The math is beyond the scope of this course, but should be convered in the "Signals and Systems" course.)

In this lab we'll look at what happens to the recordings of signals as they get close to the Nyquist limit.

One interesting effect is *aliasing* in which a frequency higher than the Nyquist frequency appears indistinguisable from a frequency lower than the Nyquist frequency. This effect is commonly observed with stroboscopes (which give you visual samples at fixed time intervals) and Moiré patterns, which sample images at a fixed spatial frequency.

To avoid artifacts in data recording, it is usually necessary to filter the input signal to remove all components that have frequencies above the Nyquist frequency. We'll look at the effect of a very crude RC low-pass filter in this lab. More sophisticated filter design is left for the courses "Signals and Systems" and "Digital Signal Processing".

3 Pre-lab assignment

You will use a signal generator in the lab to generate waveforms of known frequency and look at them using the Arduino DataLogger code. You will need a recent version of the DataLogger code that includes the "Downsampling" window on the "Windows" menu (2013 Feb 5 or newer). If you plan to use your own computer, make sure you have downloaded the latest version of DataLogger.

The first thing to do is to get a sine wave to display nicely on analog input A0. The function generator produces sine waves that are centered at 0V, but the Arduino wants signals between 0V and 5V. Design a high-pass filter that blocks the DC component of the function generator and has a 2.5V DC output when there is no input.

We're going to be looking at very low frequency signals, so the corner frequency of the high-pass filter should be around 30mHz (0.03Hz)

Design a low-pass filter that has a corner frequency around 4Hz.

4 Parts, tools, and equipment needed

Parts for this lab from kit:

- breadboard
- resistors
- capacitors (particularly electrolytic)

Parts students need to provide on their own:

- Arduino
- USB cable

Tools for this lab:

- none

Equipment in lab:

- Signal generator (0.1Hz to 100Hz desired).
- Oscilloscope for checking voltage range before connected to Arduino input.

5 Procedures

Wire up your high-pass filter, using the Arduino board's 0v and 5V lines for power. Set the function generator to have about a 4V peak-to-peak sine wave at 10Hz and look at the output of your high-pass filter on the oscilloscope (DC-coupled). Adjust the function generator output voltage (and, possibly, the design of your high-pass filter) until the scope shows that your output remains between 0 and 5 volts, but gets below 0.5V and above 4.5V. Note that the very large RC time constant for your high-pass filter means that there is a slow power-on transition when you first apply power to the filter, as the electrolytic capacitor slowly charges up. Wait for that to settle before you start trying to adjust the design.

Check that the signal remains in the desired range (low peak between 0V and 0.5V, high peak between 4.5V and 5V) at 0.1Hz, 1Hz, 10Hz, and 100Hz.

Connect the output of the high-pass filter to both A0 and A1 of the Arduino (same signal on both pins). In the DataLogger, set the timed trigger to 20msec, so that you are sampling at 50Hz. Using the "Downsampling" window, set the downsampling for A1 to 5, so that it is sampling at 50Hz/5 = 10Hz.

Set the function generator for a very low frequency (say 0.1Hz) and observe the waveforms with the DataLogger. Make notes in your lab notebook of what you observe. Make sure your notes include the frequency of the sine wave and the sampling frequency of each channel, so that you can reproduce the effect later.

Increase the frequency gradually up to 10Hz. What happens as you change the frequency? What happens at 5Hz? at 10Hz? What if you are just a little off 5Hz (4.9Hz or 5.1Hz)? What happens if you go up to 25Hz?

Find interesting behavior at various frequencies. Describe the patterns you see and record them for inclusion in your report.

What happens to the interesting waveforms if you put your 4Hz low-pass filter between your 0.01Hz high-pass filter and the Arduino? Do the filter do what you expect? Is its corner frequency close to 4Hz?

Again, record some waveforms, keeping careful records in both your lab notebook and the meta-data "Notes:" field of the DataLogger files.

6 Demo and writeup

Show the group tutor or the instructors some of the interesting waveforms.

Your report should have schematics for your entire test setup. The capacitors can be drawn as polarized (with a little plus sign) in CircuitLab by changing their parameters. Make sure your schematics show the polarity the right way around.

Your report should contain several of the interesting waveforms, together with explanations of why they occur and the effect of low-pass filtering on them.

Superimposing the higher sampling frequency plot from A0 and the lower sampling frequency plot from A1 on the same graph helps to explain what is going on. You may need to specify a small xrange (a small time interval) to show what is going on clearly.

7 Design Hints

Remember the ways that we discussed to get "virtual ground" signals half-way between the power rails. (Thévenin equivalents may be useful here.) The virtual ground circuits can be used to produce the high-pass filter that shifts the function generator output up by 2.5V.

The low-pass filter does not do any level shifting, so can be designed as an RC voltage divider with no special tricks.

Remember that electrolytic capacitors have a polarity (a positive and negative side). Make sure that you connect them up the right way around.

Remember to save some of your DataLogger recordings as files, so that you can use gnuplot to make figures for your lab report. The DataLogger gets slow sometimes when it has a lot of data in the display, so it is a good idea to save the data occasionally and clear the feed.