ESE 150 – Lab 02: Digital to Analog Conversion

LAB 02

In this lab we will do the following:
1. Take the samples you collected last week and reconstruct them in Excel
2. Learn how to write Arduino code that outputs voltages between 0 and 5V
3. Learn how to use the Arduino's serial monitor to SEND data to the Arduino
4. Have the Arduino behave as a basic D2A (digital to analog converter) to reconstruct your samples

Background:

Last week you used the Arduino to take an audio signal and turn it into a digital representation; you used an Arduino as an "A2D" converter (ADC, analog to digital converter). You should now have "samples" for the Arduino that you will use in today's lab.

Today we will import the data you captured last week from your A2D converter, into Excel. Next, we'll import the samples into the Arduino and attempt to reconstruct the signal you sampled last week!

Recall the meaning of this picture from class:
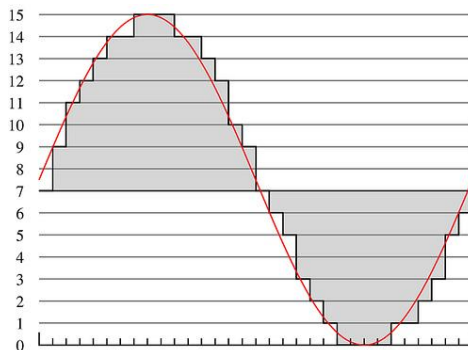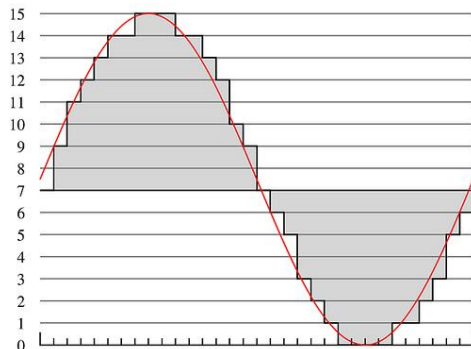
ESE 150 – Lab 02: Digital to Analog Conversion

Lab Procedure:

Prelab: Reconstructing Original Signal in Excel

- You must have the output from your Arduino from Lab 1 before starting this section.
- You should have sampled a 2Vpp 300 Hz signal at 500 Hz, 1000 Hz, and 5000 Hz at 10b of precision.
- In this section, you'll use a spreadsheet (e.g., Excel, Numbers, Google Docs spreadsheet (docs.google.com)) to re-construct your original signal from the samples you collected in Lab 1.

Note: yellow, highlighted items mark question answers, code, data, and graphs that need to appear in your lab report.

1. What was the frequency of the sine wave for which you sampled this data?
2. What does Nyquist's theorem tell us about the sampling rates you recorded compared to the signal rate (i.e. frequency of the sine wave)? (Hint: Which (if any) of the cases are undersampled? What does that tell us about our ability to process the signal?)
3. Open up your 10b sampled data in excel (you should have 3 columns: 500 Hz, 1kHz, 5kHz).
   a. Recall that you took 800 samples, so you should have 800 rows)
4. Use Excel's plotting capability to produce a plot like the below plot _**for each column**_ of your data:
   a. The x-axis should just be the row # (aka the sample #).
   b. The y-axis should be the quantization level: 0->1023 (since you are using the Arduino's _10-bit_ quantization data).
   c. Use a line plot.
   d. Zoom into your plots to show 3 cycles of waveform.
   e. Give each of the 3 plots a title and label the axes with the appropriate units.

5. Next, create & label 3 new plots:

a.　First use your knowledge of the sample rates, and the time between each sample that they imply, to create 3 new columns that contain the time in seconds of each of the 800 samples, for each of the 3 sample rates.

b.　Next use your knowledge of the voltage range of the Arduino (0v-5v) and the quantization levels to create 3 new columns that contain the voltage of each of the 800 samples, for each of the 3 sample rates.

c.　Use these new columns to create 3 new plots.

d.　The x-axis should now be: **TIME.**

e.　The y-axis should now be: **VOLTAGE**

(your lowest voltage should be 0 due to the 1V offset that you used when you sampled the data).

f.　Zoom into your plots to show 3 cycles of waveform.

g.　Give each of the 3 plots a title and label the axes with the appropriate units.

6.　How close does each plot look to a sine wave? Explain the difference between the three plots based on what we've covered so far in the course. [1 paragraph]

7.　For the 5000Hz sample rate data:

a.　Compute a column that quantizes the data to 8b of precision (you may want to think about rounding the quantized data using Excel's ROUND function. Consider what re-quantizing data means in terms of rounding).

b.　Compute a second column that quantizes the data to 2b of precision.

c.　Create a pair of voltage-time plots (as in question 5) for the 8b and 2b quantized data sets you just computed.

8.　How close does each plot look to a sine wave? Explain the difference between the three quantization plots (10b, 8b, 2b) based on what we've covered so far in the course. [1 paragraph]

In your report, turn in the 8 labeled plots.

9.　Watch the Video for the Digital Multi-Meter (DMM): https://youtu.be/Or-M7_bCZbk

When you arrive in lab, compare your answers with your assigned partner. If your answers differ, discuss amongst yourselves and try to resolve your differences.

Early during the lab session, a TA will check you off on prelab. Go ahead and start working on the lab.   It will take some time for the TAs to get around to all the groups.

ESE 150 – Lab 02: Digital to Analog Conversion

Lab – Section 1: Setting up the Arduino as a D2A

- In this section you'll calibrate your Arduino to output voltages between 0 and 5V
- If you need a refresher on using the lab equipment, remember there are instructions and video links on Lab 1.

1. Obtain the following items from the lab bench stock (near the front of Detkin lab):

    (1) Arduino
    (1) Breadboard
    (1) 4.7 kΩ resistor
    (1) 10 uF Capacitor
    Wires to connect things

2. Using your Arduino and breadboard, connect the components above as shown in this schematic:
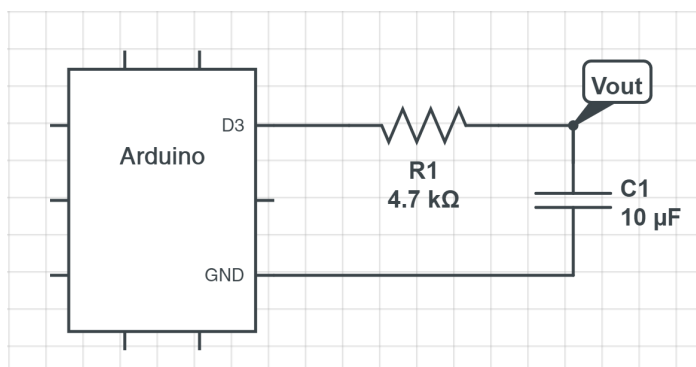
   a.   Ensure that you connect the resistor to digital pin #3 on the Arduino.
   b.   Ensure that you connect the negative terminal (the side with a stripe) of the capacitor to the Arduino's Ground terminal.
   c.   Remember that Lab 1 instructions contained details about how the connection points on the breadboard are connected. Refer back to Lab 1 if you need a refresher.

3. Locate a "BNC -> Minigrabber" cable in the parts box at your workstation:

4. Connect one end to the oscilloscope, the <u>red</u> wire to "**VOUT**" in your circuit, and the black wire to "**GND**".
5. Turn on your oscilloscope.
6. Locate (2) banana to minigrabber wires, one RED, one BLACK:

7. Connect the red wire to the "VOUT" of your circuit, and the other to the "HI" terminal on your **DMM** (Digital Multimeter).
8. Connect the black wire to the "GND" of your circuit, and the other to the "LO" terminal on your **DMM** (Digital Multimeter).
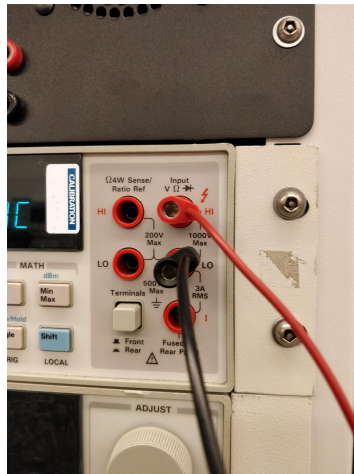
9. Turn on the DMM.

10. Connect your Arduino to your workstation's PC via USB and open up the Arduino IDE software.

11. Copy and paste the following code into your Arduino:

```
int pwmOut = 3 ;     // pin we'll output signal on
int holdTime  = 300 ;  // how long to hold output on PWM pin

void setup() {

  Serial.begin(9600);  // setup serial monitor speed
  pinMode(pwmOut, OUTPUT);  // configure pin for output only

}

void loop() {

  analogWrite(pwmOut, 0); // about zero volts
  delay(holdTime);
  // delay argument is in milliseconds
  analogWrite(pwmOut, 51); // about 1 volt
  delay(holdTime);
  analogWrite(pwmOut, 102); // about 2 volts
  delay(holdTime);
  analogWrite(pwmOut, 255);  // about 5.0 volts
  delay(holdTime);
}
```

12. Upload and run. Observe the resulting waveform on the oscilloscope. What does the waveform look like?

      a. The waveform won't necessarily go all the way to 5V. That is ok.

Look carefully at the code and determine how it works:
1) Arduino's output pins can only put out voltages 0V or 5V, but nothing in between!
    a. It actually puts out a square pulse alternating between 0 and 5V every 2ms.
2) What we can control is how long the output stays 5V during 1 cycle of that pulse.
3) The line of code: analogWrite(3, 64) would hold 5V for about 25% of that cycle (see figure below).
4) The line of code: analogWrite(3, 255) will make it hold 5V for the entire cycle.
5) This is referred to as "Pulse Width Modulation" or PWM, this chart helps visualize:
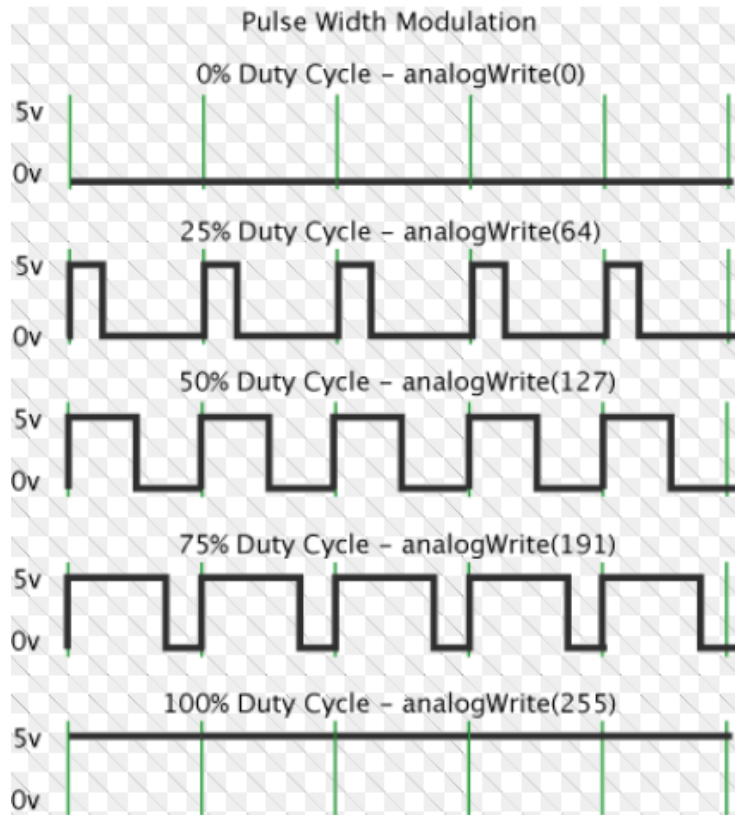
(This chart is provided entirely for explanatory purposes;
This is not something for you to do, and you won't see this during your experiments.)

What is the effect of this code on the resistor-capacitor?

1) When the Arduino puts out 5V for 25% of its duty cycle, on pin 3, current flows through the capacitor and charges up the capacitor to about 1V.

2) When the Arduino puts out 5V for 100% of its duty cycle, the capacitor and charges up the capacitor to about 5V.

3) This happens because the resistor & capacitor have what's called an "RC time constant" you may know this from physics. Basically, it takes time to charge up the capacitor all the way to 5V when the output is high and time to discharge when the output is low. We're taking advantage of this delay to produce voltage between 0V and 5V (e.g. – 0, 1, 2, 3V etc.): Depending on how long the output is high for we can keep the capacitor charged at a specific level, essentially outputting a voltage.

ESE 150 – Lab 02: Digital to Analog Conversion

YOUR FIRST JOB (calibrate Arduino output voltage):

1) Add to the provided Arduino code to produce voltages: 0V, 1V, 2V, 3V, 4V, 5V.
2) The DMM is measuring the voltage across the capacitor, so you will know If your code is working! Adjust the PWM values (0-255) accordingly to calibrate until you get as close as possible to the correct output voltage values.
3) Save your Arduino code.

YOUR SECOND JOB (produce an approximation sine-wave):

1) Make your Arduino output the following voltages in this sequence:
2V, 3V, 4V, 5V, 4V, 3V, 2V, 1V, 0V, 1V, 2V
2) Capture an oscilloscope screenshot of the approximated sine-wave.
   a. Use Excel as you did last week.
3) Save your Arduino code.

ESE 150 – Lab 02: Digital to Analog Conversion

Lab – Section 2: Communicating with the Arduino through the Serial Monitor

- We'd like to convert the samples you took last week, back to a voltage using the Arduino
- We need to find a way to send the sample BACK to the Arduino
- For this we'll use the Serial Monitor to actually send data to the Arduino (instead of just receive it)

1. Create a new sketch and copy and paste the following code in your Arduino IDE:

```
#define MAX_SAMPLES 801

// global variables
int samples [MAX_SAMPLES] ;
boolean samplesReceived = false ;

int outputPin = 3 ;     // PWM digital output pin
int holdTime  = 200 ;   // how long to hold output on PWM pin

void setup() {

  Serial.begin(9600);  // setup serial monitor speed
  pinMode(outputPin, OUTPUT);  // configure pin for output

}

void loop() {

  int i = 0 ;
  while (Serial.available() && i < MAX_SAMPLES ) {
      samples[i++] = Serial.parseInt() ;
      samplesReceived = true ;
  }
}
```
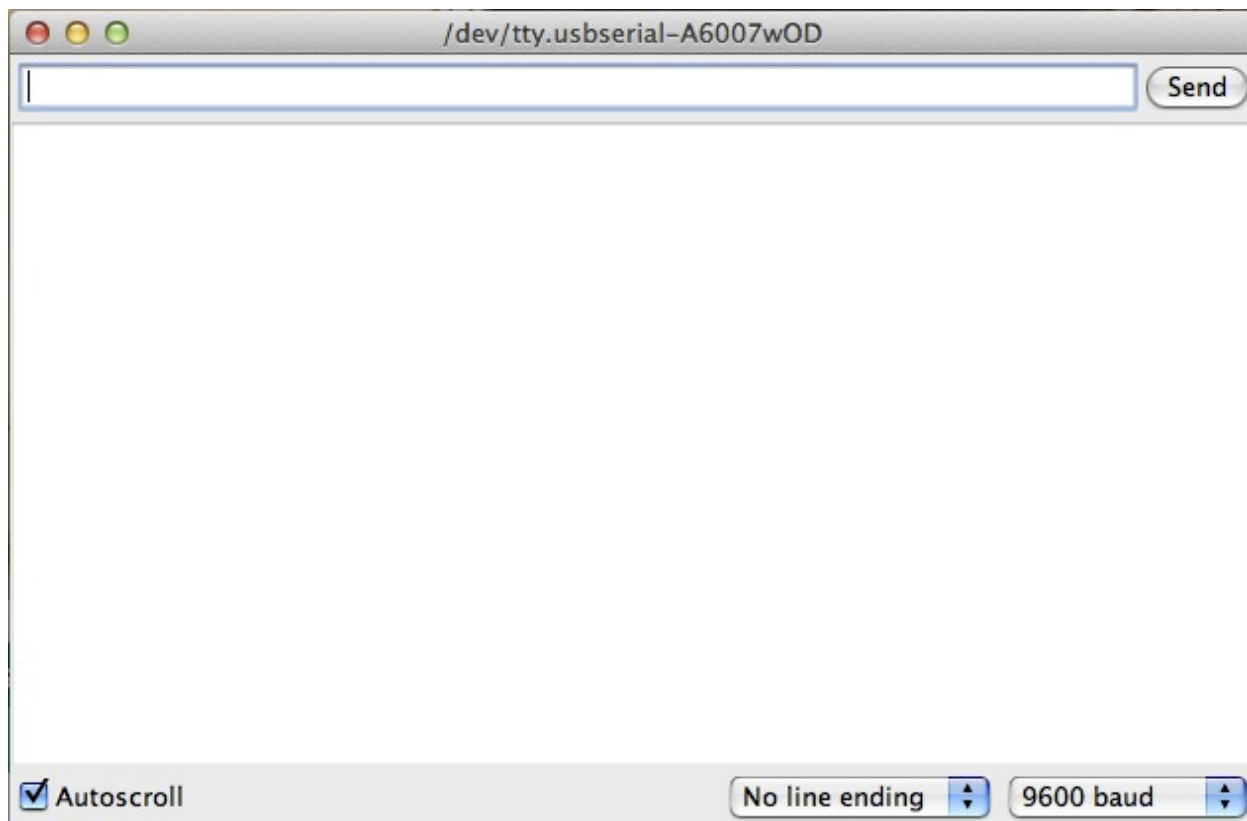
2. Upload the code to your Arduino

As you saw last week, this may generate a message "Low memory available, stability problems may occur"; this is expected and should not indicate a problem.

3. Open the Arduino's Serial Monitor:

4. In the top bar, type in the numbers: 100 200 300 400 1024, then press SEND.
5. Those numbers have now been sent to the Arduino, "parsed" and stored in an array.
6. Add to the Arduino code so that whenever it receives samples it sends back all the samples back to the Serial monitor (use Serial.println()).

> Note: In the past, students have had difficulty getting the correct data written back. We have found that first recording the values in a loop and then printing them in a second loop is the most reliable way to achieve the desired results.

7. Try copying and pasting all 800 samples from your Excel spreadsheet to the serial monitor and clicking SEND.
   a.   Note: In Excel, to select an entire column click on the letter denoting which column it is.
   b.   For this you can use any of your sample data sequences from last week.
8.   Save your modified Arduino code.

ESE 150 – Lab 02: Digital to Analog Conversion

Lab – Section 3: Putting it all together, Arduino as an A2D

- You now have a way of sending your samples to the Arduino
- You also have a way of putting voltages out to an Arduino output pin

1. Determine a conversion factor between quantized data and voltage (Prelab question 5 will be helpful):
    a.   Your samples are between 0 and 1023.
    b.   You can only output voltages between 0 and 5V, and you can only do so by selecting appropriate PWM values between 0 and 255.
    c.   Given what you know about Section 2's code, can you determine an offset and a conversion factor to multiply the samples by to scale them between 0 and 255, simulating an output between 0 and 5V?

2. Create a new sketch that combines Section 1 and Section 2's Arduino code to output your samples as voltages.
3. Your program should receive all 800 samples.
    a.   As you receive values, your program should convert each sample to a PWM value using the conversion factor found in Question 1 and save that in the array.
    b.   For the waveform generation that follows, use the data from the highest sampling frequency you collected.
    c.   This input loop should run only once.
4. It should then output all 800 samples (now converted to PWM values) repeatedly to the output pin.
    a.   This output loop should repeat.
    b.   This loop should not start until after completing the input loop.
5. This will take time. Once you have the output working take a screenshot of the data on your oscilloscope showing 3 cycles of the sine wave. The PWM values you converted your samples to should appear as voltages between 0 and 5v.
6. Before leaving lab, show your generated sine wave to a TA and answer a few questions. This is the Lab Exit Check-off.
7. Make sure code and snapshots are available to both partners before you leave lab.
    a.   As we suggested last week, we recommend you put the data on a Google Drive shared between both partners **before** you leave the lab.
8. Cleanup your lab station, leaving everything as you found it when you arrived.

Postlab: Synthesis

1. What equation could you use to create the sample data for the sine wave mathematically? (a function of sine-wave frequency, sample rate, and quantization).
    a. What is the equation for a continuous sine wave, as a function of frequency and time?
    b. How can you modify that equation to compute discrete values for points in time, as a function of frequency and sample rate? Remember: how often are samples taken with a given sample rate?
2. Using the equation you found in Problem 1, develop a spreadsheet that reproduces the 800 samples for the 300Hz sine wave sampled at a 1000 Hz sample rate with 8b of precision.
    a. You will want to have frequency and sample rate (and later quantization) as separate cells in Excel and reference them in your Excel version of the equations.
       Hint: How does A1 behave differently from A1?
    b. You can use a column in Excel for each sample number and apply your equation to each of those cells in a second column to get a column of datapoints. (If you are using AutoFill to apply the function to each sample while referencing the set parameter cells, this link may be quite helpful! https://stackoverflow.com/questions/2156563/how-to-keep-one-variable-constant-with-other-one-changing-with-row-in-excel)
    c. How can you further modify your equation to quantize the data? Again, you will want this parameter as a separate cell referenced in your equation.
3. With a small change from the first spreadsheet, i.e. using the same equation from Question 2, create a second spreadsheet that produces a 100Hz wave sampled at 1000Hz with 8b of precision.

With this technique, you can create (*synthesize*) sounds directly—no need to generate and sample the source.

HOW TO TURN IN THE LAB

- Upload a PDF document to canvas containing:
  - Prelab (8 plots, 2 answers)
  - Section 1 (revised Arduino code to produce specified voltages, approximate sine-wave screen shot)
  - Section 2 (revised Arduino code)
  - Section 3 (final Arduino code and screenshots)
  - Postlab (answer to Question 1 in PDF)
- Please include adequate labels and text so it is clear where you have included each item requested in your report.
- Upload your postlab spreadsheets (postlab Question 2&3) to the designated canvas lab assignment.
- For your convenience, each item that is required to put in your report/demo is also highlighted within the lab writeup.
- Due by Friday 5pm.