Digital to Analog Conversion Using Pulse Width Modulation

Purpose

Build and test a sawtooth signal generator using pulse width modulation and the Arduino microcontroller.

Equipment Required

Qty.	Description
1	Arduino Uno microcontroller board
1	USB cable (to program and power the Arduino)
2	LEDs
1	DC Power Supply
1	741 Op Amp
1	DMM
1	Oscilloscope
	Resistors and Capacitors as needed

Prelab (complete before lab!)

1. Read the tutorial on pulse width modulation on the Arduino website, `http://arduino.cc/en/Tutorial/PWM`.

2. What is the smallest voltage step available with the **analogWrite()** function?

3. Read the rest of the lab before you proceed.

4. Look at the **ramp** program source code. Sketch the value of **outputLevel** as a function of time. What is the period of this function?

5. Design a second order Sallen Key low pass Bessel filter with a cutoff frequency of 5 kHz. Draw the circuit neatly in your lab book, and label the values of all components. You may use the recipes from the course web site,
`http://solar.physics.montana.edu/kankel/ph262/filters/`.

6. Now look at the gain plots on the course website,
`http://solar.physics.montana.edu/kankel/ph262/filters/calculations/`.
The horizontal axis is frequency divided by the critical frequency (that is, it's plotted as if $f_c = 1$). By what factor (compared to DC) would your filter suppress a 30 kHz signal?

Procedure

1. Connect Pin 9 of the Arduino to the LED anode. Connect the cathode of the LED through an appropriate resistor to the Arduino ground.

2. Load the **ramp** program onto the Arduino (see the Appendix for source code).

3. Describe the behavior of the LED. Is the brightness as a function of time consistent with the **outputLevel** plot you made in the prelab?

4. Use the oscilloscope to view the signal from Pin 9.

   (a) What is the frequency? Is the frequency constant?

   (b) Describe how the duty cycle corresponds to LED brightness.

5. The frequency of the PWM on pins 9 and 10 can be varied by manipulating register TCCR1B.

   (a) Uncomment the appropriate line in **setup()** and run it on the Arduino.

   (b) Has the LED behavior changed significantly?

   (c) Repeat step 4.

6. Now build the filter you designed during pre-lab.

   (a) Use the lab power supply to generate $\pm V_s$ and ground for the filter.

(b) Use the square wave generator to measure the step response of your filter. The output signal should be noticeably rounded off compared to the input when viewed at 100 μs per division. A Bessel should have little or no overshoot. Print a copy of the oscilloscope output and paste into your lab journal.

(c) Connect the output of Pin 9 to the filter input. Connect the filter output to an LED and resistor identical to what you have on Pin 9.

(d) Do the two LEDs behave differently? Describe.

(e) Use the oscilloscope to compare the filtered and unfiltered outputs. Print and paste an example to your log book. Can you see the PWM oscillation in the filtered output? If so, estimate its amplitude.

Conclusion

1. What is the voltage resolution of PWM on the Arduino?

2. What is the default frequency of PWM on the Arduino?

3. Explain the reason for the difference in behavior of the two LEDs (connected to filtered and unfiltered outputs).

4. Was the suppression of PWM oscillations by your filter consistent with the pre-lab estimate?

5. At its most basic, a PWM output consists of a clock signal, a counter, and some logic that turns an output on or off based on the values in the counter. Which of these three elements must have been changed by manipulating the TCCR1B register in the **ramp** program? Explain your reasoning.

Appendix: Arduino Program

```
/*
ramp   (PHSX262 Lab 8, March 2013)

This example uses analogWrite() to output a ramp (sawtooth) signal to pin 9.
This can be used to vary the brightness of an LED.
*/

const int pwmOut = 9; //PWM output will be to this pin.
byte outputLevel = 0; //Variable used for output to the pin.
int dt = 4000; //Delay between steps, in microseconds.

void setup() { //Set output low to begin.
  pinMode(pwmOut, OUTPUT);
  digitalWrite(pwmOut, LOW);
  // TCCR1B = TCCR1B & 0b11111000 | 1;
    // Un-comment the above line to set Timer1 prescale (PWM pins 9 and 10)
    // for fast (31.25 kHz) operation.
}

void loop() {
  analogWrite(pwmOut, outputLevel);
  outputLevel++;
  delayMicroseconds(dt);
}
```