San José State University
Department of Mechanical and Aerospace Engineering

Lab 3 – RC Filters and Basic Timer Functionality February 24, 2017

Report Author: Eric Rosenfeld

Lab Team: Eric Rosenfeld

Lab Date: February 24, 2017

Summary

This report details the characteristics of RC filters and the results obtained from incorporating RC filters into circuits. An input voltage was supplied into the RC filter circuit by the function generator, then output voltage was measured using the oscilloscope. The capabilities and applications of the Arduino Microcontroller are further explored in this lab, and now incorporates using a speaker to become familiar with more inputs and outputs connected to the header pins on the Arduino. Additionally, the timer function on the function generator was utilized to create a tone for a duration of time. Version 4.5 of the EduShields YouKnow board was used for the experiments.

Results/Discussion

Introduction

RC Filter Theory: Low-Pass Filter

In a given higher order circuit, transfer functions are derived to give the ratio of output voltage (V_{OUT}) to input voltage (V_{IN}) as well as the order of the circuit. For a low-pass filter, for example, the transfer function is (Equation 2 from the ME 106 Lab Manual):

$$\frac{V_{OUT}}{V_{IN}} = \frac{1}{1 + j\omega RC}$$

where: V_{OUT} = Output voltage
V_{IN} = Input voltage
j = a complex number $\sqrt{-1}$
ω = Frequency of V_{IN} in $\frac{rad}{s}$
R = Resistor Value
C = Capacitor Value

Because they are inversely related, as the frequency of the input voltage increases, the value of the transfer function $\frac{V_{OUT}}{V_{IN}}$ decreases.

The low-pass filter found in the ME 106 manual was constructed on the breadboard for this exercise. The resistor value was 1.1 kΩ and the capacitor value was 0.1 µF.

San José State University
Department of Mechanical and Aerospace Engineering

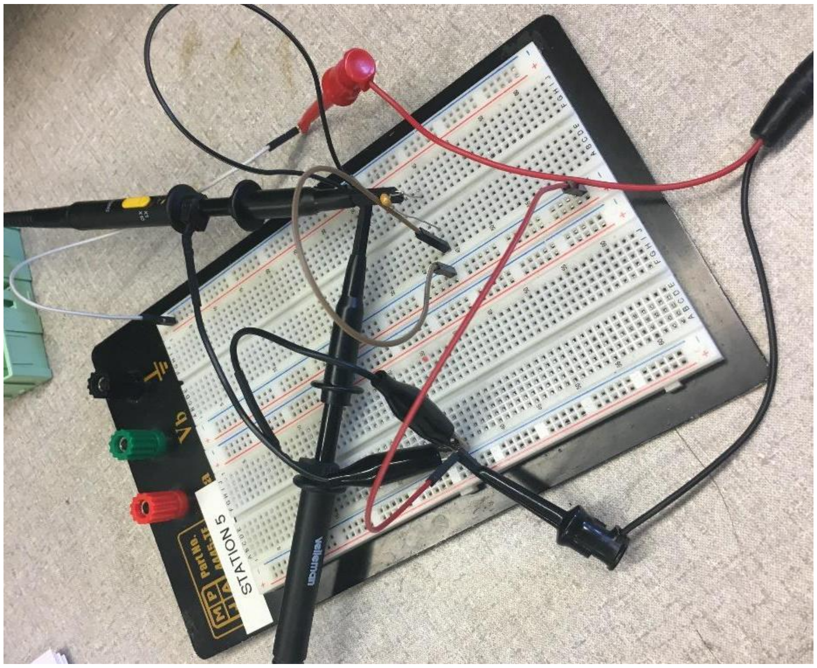

Figure 1. RC Low-Pass Filter circuit constructed on the breadboard. V_{OUT} is measured across the Capacitor.

The function generator was set up to output a sine wave at V_{IN} = 5 V_{P-P} for various frequencies. The amplitude of V_{IN} and V_{OUT} was then measured via the oscilloscope.

Frequency [Hz] "f"	**Frequency** $[\frac{rad}{s}]$ **ω = 2πf**	V_{IN} **[mV]**	V_{OUT} **[mV]**
500 Hz	3141.593 $\frac{rad}{s}$	506.3 mV	468.8 mV
1.6 kHz	10053.1 $\frac{rad}{s}$	506.3 mV	343.8 mV
10 kHz	62831.9 $\frac{rad}{s}$	506.3 mV	84.38 mV

Additionally, the ratio of the output voltage to the input voltage $\frac{V_{OUT}}{V_{IN}}$ that were measured via the oscilloscope were compared with the magnitude of the transfer function from Equation 2 above. Substituting R = 1.1 kΩ and C = 0.1 μF into the transfer function for low-pass filters from above, the magnitude of the transfer function becomes the following equation:

$$\frac{V_{OUT}}{V_{IN}} = \frac{1}{1 + j\omega \times (1.1\ k\Omega) \times (0.1\ \mu F)}$$

Each V_{IN} angular frequency in $\frac{rad}{s}$ is then substituted into the function. Afterwards, the function is evaluated. Lastly, the magnitude of the real and imaginary part of the function is taken.

Frequency $[\frac{rad}{s}]$ **ω = 2πf**	**Ratio** $\frac{V_{OUT}}{V_{IN}}$	**Magnitude of Transfer Function** $\|\frac{V_{OUT}}{V_{IN}}\| = \sqrt{(REAL)^2 + (IMAGINARY)^2}$
3141.593 $\frac{rad}{s}$	0.92593	$\sqrt{(0.9988039104)^2 + (-0.34563839)^2} = 0.9994017763$
10053.1 $\frac{rad}{s}$	0.67904	$\sqrt{(0.4498659657)^2 + (-0.4974802294)^2} = 0.6707204826$
62831.9 $\frac{rad}{s}$	0.166660	$\sqrt{(0.0205048471)^2 + (-0.1417194353)^2} = 0.1431951364$

The percent differences are: 7.63212%, 1.23274%, and 15.1457% respectively. **(Question 1) Although the percent difference corresponding to the 1.6 kHz was at a low 1.23274%, the other two percent differences are high probably due to the fluctuations in the output voltage seen on the oscilloscope during lab. The larger the frequencies became, the more that the output voltage was reduced. This circuit in which the resistor and capacitor are configured as such (with V_{OUT} being measured across the capacitor) is called a low-pass filter because lower frequencies are "passed" to the output with little attenuation [cut-off] and higher frequencies are significantly attenuated (i.e., not "passed")** (Alciatore & Histand, 2011, p. 128)**.**

The low-pass filter circuit was still in use for the following task: the phase lag of the low-pass circuit at 1.6 kHz was to be measured on the oscilloscope. The phase lag is the measure of the amount of time that the output voltage lags behind the input voltage, and it is measured in degrees (Alciatore & Histand, 2011, p. 129). First, the theoretical value of the phase lag is calculated for 1.6 kHz, using the Imaginary and Real parts of the Transfer Function:

$$\phi = (\frac{IMAGINARY}{REAL})$$

$$= \left(\frac{0.4974802294}{0.4498659657}\right) = 47.877°$$

Next, the observed value of the phase lag is calculated for 1.6 kHz. The procedure listed in the lab manual was followed in order to align the input voltage waveform and the output voltage waveform, as shown in the image below.

For V_{IN}, Period [T] = 625 µs
For V_{OUT}, Period [T] = 625 µs

$$t_1 = 400\ \mu s$$
$$t_2 = 484\ \mu s$$
$$\Delta t = t_2 - t_1 = 84\ \mu s$$

$$\phi = \frac{(360)\ \times\ (\Delta t_d)}{T}$$

where: Φ = Phase lag, or phase angle
Δt_d = time displacement between the input and output signal [sec]
T = Period of the signals [sec]

$$= \frac{(360)\times(84\ \mu s)}{625\ \mu s} = 48.384\ °$$

(Alciatore & Histand, 2011, p. 129). **(Question 2) One period on the waveform represents 360°** (Furman, 2017, p. 3). **The theoretical value of 47.877° agrees with the observed value of 48.384°, with a 1.05339% difference. This means that the values for the Transfer Function are correct, and the time displacement is correct.**

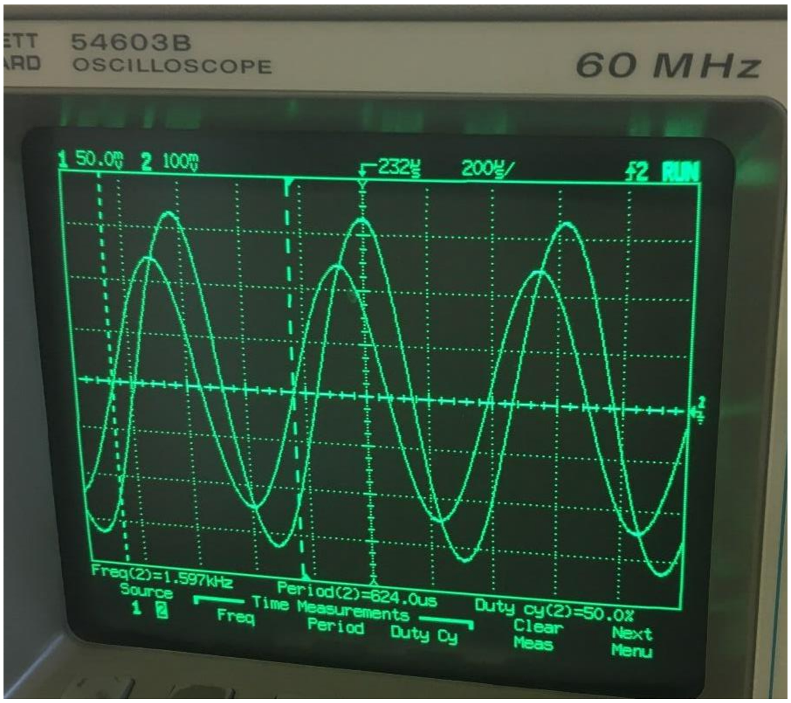

Figure 2. The input voltage waveform and the output voltage waveform are depicted after aligning them so that it would be easier to measure the phase lag.

RC Filter Theory: High-Pass Filter

As stated earlier, transfer functions are derived to give the ratio of output voltage (V_{OUT}) to input voltage (V_{IN}) as well as the order of the circuit. Now, the transfer function for a high-pass filter is (Equation 4 from the ME 106 Lab Manual):

$$\frac{V_{OUT}}{V_{IN}} = \frac{j\omega RC}{1 + j\omega RC}$$

where: V_{OUT} = Output voltage
V_{IN} = Input voltage
j = a complex number $\sqrt{-1}$
ω = Frequency of V_{IN} in $\frac{rad}{s}$

R = Resistor Value
C = Capacitor Value

Now, the high-pass filter found in the ME 106 manual was constructed on the breadboard for this exercise. The capacitor value and the resistor value were once again 0.1 μF and1.1 kΩ, respectively.

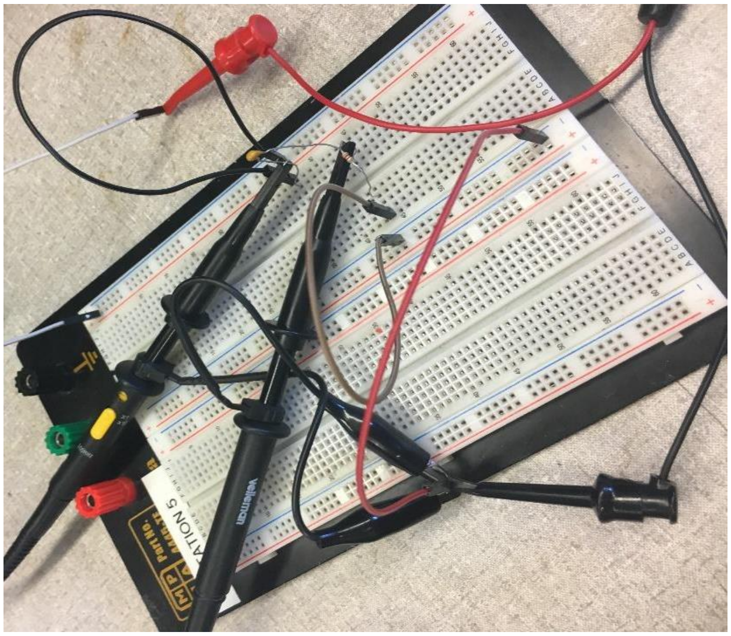

Figure 3. RC High-Pass Filter circuit constructed on the breadboard. Notice that the Resistor and Capacitor have switched places. Now, V_{OUT} is measured across the Resistor.

Again, the function generator was set up to output a sine wave at V_{IN} = 5 V_{P-P} for various frequencies. The amplitude of V_{IN} and V_{OUT} was then measured via the oscilloscope.

Frequency [Hz] "f"	**Frequency $[\frac{rad}{s}]$ ω = 2πf**	**V_{IN} [mV]**	**V_{OUT} [mV]**
500 Hz	3141.593 $\frac{rad}{s}$	506.3 mV	153.1 mV
1.6 kHz	10053.1 $\frac{rad}{s}$	493.8 mV	340.6 mV
10 kHz	62831.9 $\frac{rad}{s}$	493.8 mV	493.8 mV

Additionally, the ratio of the output voltage to the input voltage $\frac{V_{OUT}}{V_{IN}}$ that were measured via the oscilloscope were compared with the magnitude of the transfer function from Equation 4 above. Substituting C = 0.1 μF and R = 1.1 kΩ into the transfer function for high-pass filters from above, the magnitude of the transfer function becomes the following equation:

$$\frac{V_{OUT}}{V_{IN}} = \frac{j\omega \times (1.1\ k\Omega) \times (0.1\ \mu F)}{1 + j\omega \times (1.1\ k\Omega) \times (0.1\ \mu F)}$$

Each V_{IN} angular frequency in $\frac{rad}{s}$ is then substituted into the function. Afterwards, the function is evaluated. Lastly, the magnitude of the real and imaginary part of the function is taken.

Frequency $[\frac{rad}{s}]$ **ω = 2πf**	**Ratio** $\frac{V_{OUT}}{V_{IN}}$	**Magnitude of Transfer Function** $\|\frac{V_{OUT}}{V_{IN}}\| = \sqrt{(REAL)^2 + (IMAGINARY)^2}$
3141.593 $\frac{rad}{s}$	0.30620	$\sqrt{(0.1066820325)^2 + (0.3087085621)^2} = 0.3266221553$
10053.1 $\frac{rad}{s}$	0.68975	$\sqrt{(0.5501340343)^2 + (0.4974802294)^2} = 0.7417102091$
62831.9 $\frac{rad}{s}$	1.0	$\sqrt{(0.9794951529)^2 + (0.1417194353)^2} = 0.9896944745$

The percent differences are: 6.45431%, 7.25975%, and 1.03589% respectively. **(Question 1) The percent differences are now reasonably low, meaning the fluctuations in the output voltage seen on the oscilloscope were less, and the was overall more accuracy. In contrast with the low-pass filter, in which case the larger the frequencies became the more that the output voltage was reduced, this time, the lower the frequency the more that the output voltage was reduced. This circuit in which the capacitor and resistor are configured as such (with V_{OUT} being measured across the resistor) is called a high-pass filter because higher frequencies are now "passed" to the output with little attenuation and lower frequencies are now "not-passed" and are more attenuated [cut-off]** (Alciatore & Histand, 2011, p. 128)**. The ratio of $\frac{V_{OUT}}{V_{IN}}$ for 10 kHz is 1.0, which most likely means that the high-pass filter has reached its limit – it cannot attenuate any frequencies higher than 10 kHz.**

The oscilloscope was then used to measure the phase lag of the high-pass filter circuit at 1.6 kHz. The same results as with the low-pass filter were obtained. **(Question 2) The theoretical value of the phase lag was 47.877° and the observed value was 48.384°, leading to a percent difference of 1.05339%. Again, this means that the values for the Transfer Function are correct, and the time displacement is correct.**

Using the high-pass filter circuit, a DC offset was to be added from the function generator output. **(Question 3) With a +2.00 V DC Offset value for 10 kHz, the peak-to-peak voltage of V_{IN} was 487.5 mV and the peak-to-peak voltage of V_{OUT} was also 487.5 mV. Adding a DC offset voltage does not affect the output voltage V_{OUT} of the high-pass filter because the capacitor blocks the DC component of the signal.**

Using the Arduino to Output a Frequency

Basic Introduction to IC Timer-Counters

(Question 4) Using a 16-bit counter (Timer1 on the ATmega328), a chosen prescaler value of 8, and a standard 16 MHz clock tick, it will take the counter 32768 microseconds to reach its maximum value.

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

$$2^{16} = (65536) \times (0.5\,\mu s\ counter\ tick) = 32768\,\mu s$$

Driving a Speaker with the Function Generator

An auxiliary speaker was used for this exercise. **(Question 5) Using the Digital Multimeter (DMM), the DC impedance of the auxiliary speaker was measured to be 3.628 Ω.** The function generator was then set up with a 1 V peak-to-peak (V_{P-P}), 0 V offset, sine wave signal. Also, it was set for a 50 Ω output impedance because the speaker is a low-impedance load. The function generator was attached to the speaker, and the oscilloscope was attached across the them. **Using the knob on the function generator to increase the frequency, the tone could no longer be heard at a frequency of 16 kHz (outside of human hearing range)**.

Next, the function generator was set up to sweep frequencies from 200-15,000 Hz using the procedure listed in the lab manual. **(Question 6A) After the sweep button on the generator is pressed, a low police siren sound, a low *"whooo"*, is emitted from the speaker.** The waveform was then scaled vertically on the scope screen to get a better view of the behavior of the waveform. To do so, Auto-scale was pressed, and the horizontal scale was changed from 1.00 ms/div to 2.00 ms/div using the horizontal Time/Div knob. **(Question 6B) On the oscilloscope, frequency increased continuously until the sweep reached the end of the range (15,000 Hz), at which point the waveform began the cycle again. Also, the amplitude of the waveform spikes near the beginning of the sweep cycle probably because a sudden increase in frequency causes a sudden increase in voltage. This continues in an endless cycle, and images of each instance are displayed below.**

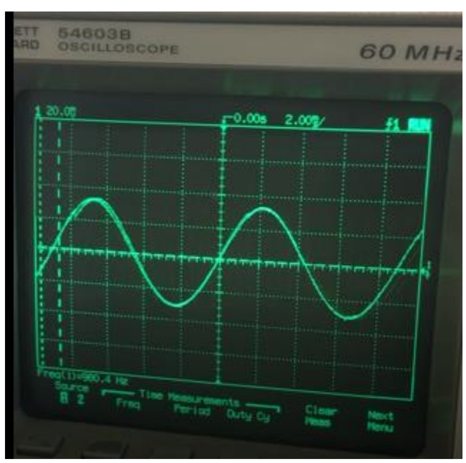

Figure 4. During the beginning of the sweep cycle, the sinusoidal waveform observed on the oscilloscope started with a low frequency and low amplitude.

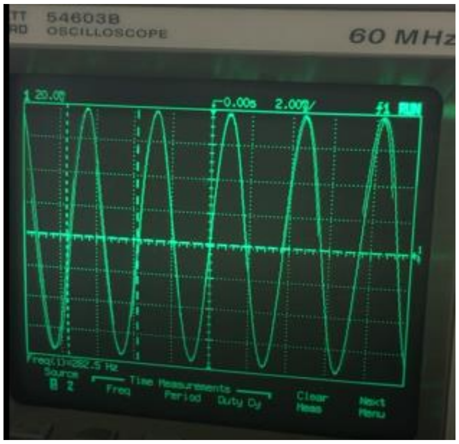

Figure 5. Also near the beginning of the sweep cycle, the frequency of the sinusoidal waveform observed on the oscilloscope increased as it swept across the range. Amplitude also spiked.

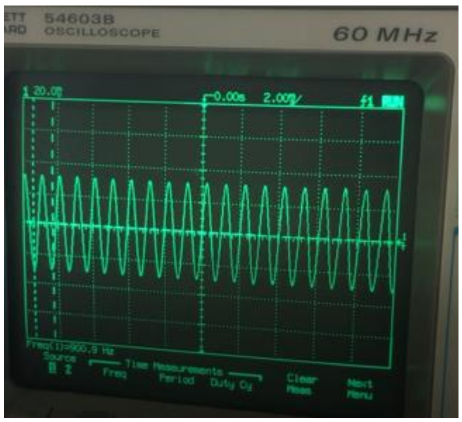

Figure 6. Frequency continued to increase until the end of the sweep (15,000 Hz) was reached. Amplitude had eventually reduced as well.

(Question 6C) The waveform on the function generator was switched from a sine-wave to a square-wave. The tone then sounded different: a louder, higher-pitched tone that sounded similar to an evacuation alarm emitted from the speakers. But the same behavior was observed as with the sinusoidal wave: frequency increased continuously until the sweep reached the end of the range (15,000 Hz), at which point the waveform began the cycle again. Also, the amplitude of the waveform spikes near the beginning of the sweep cycle probably because a sudden increase in frequency causes a sudden increase in voltage. This continues in an endless cycle, and images of each instance are displayed below.

Because the square wave is not smooth like the sine wave, the input signal will also not be smooth, so the tone will not be as smooth. This is similar to a capacitor charging and discharging, which is represented by an exponential graph. The sine wave may be preferred for maintaining a constant low-sounding noise/tone, while the square wave might be preferred for applications that require sudden noises, such as a fire or evacuation alarm.

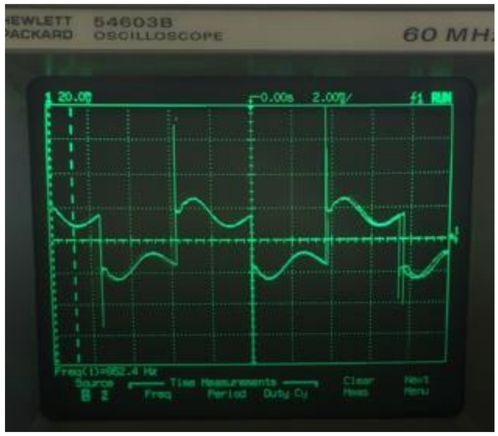

Figure 7. During the beginning of the sweep cycle, the square waveform observed on the oscilloscope started with a low frequency and low amplitude.

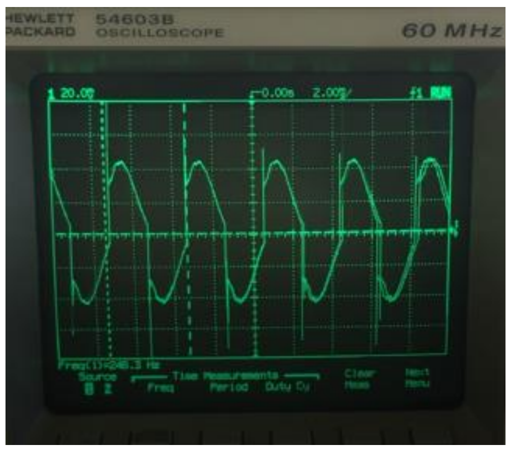

Figure 8. Also near the beginning of the sweep cycle, the frequency of the square waveform observed on the oscilloscope increased as it swept across the range. Amplitude also spiked.

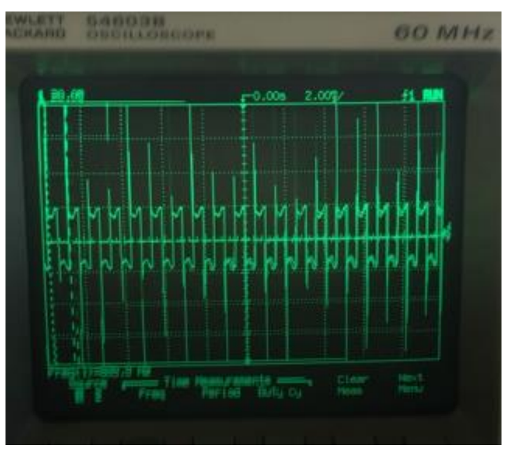

Figure 9. Frequency continued to increase until the end of the sweep (15,000 Hz) was reached. Amplitude had eventually reduced as well.

Now, instead of using the function generator, the Arduino was used to output audio to a real, auxiliary speaker. The circuit shown in the lab manual was constructed on the breadboard. **(Question 7) With the 3.628 Ω speaker and a 176.36 Ω resistor connected, the current limited was calculated using Ohm's Law and series resistors:**

$$V = I \times R$$

Where V = 5 V from the USB cable
And R = 3.628 Ω + 176.36 Ω = 179.988Ω

$$I = \frac{V}{R} = \frac{5\,V}{179.888\,\Omega} = 0.02778\,A = \ 27.78\,mA$$

(Question 8) Now, an Arduino program was made using the Arduino IDE that generated a sweep tone from a minimum of 100 Hz to a maximum of 15,000 Hz in 5 seconds. Refer to Appendix A for the pitches.h source code and to Appendix B for the sketch pertaining to this exercise.

(Question 9) Now, the Arduino program was modified to time the sweep sequences and print the result on the Serial Monitor. Refer to Appendix C for the sketch pertaining to this exercise.

(Question 10) The breadboard was put aside for the last exercise. The function generator was set up to produce an input square wave with 5 V_{P-P} amplitude, 2.5 V offset, 50% duty cycle, Hi-Z termination, for any frequency between 1 kHz and 5 kHz. The frequency chosen was 10 kHz, and the generator was connected to digital pin 11 on the Arduino. An Arduino sketch was created that used the `pulseIn()` function to time the length of the high portion of the signal pulses from the function generator and printed it out on the Serial Monitor. Refer to Appendix D for the sketch pertaining to this exercise.

Figure 10. Serial Monitor displayed the duration of the high portions of the signal pulses from the function generator.

Conclusions and Recommendations

Lab 3 demonstrated the characteristics and uses of RC filters: low-pass and high-pass filters in a circuit. For low-pass filters, the higher the frequency became, the more that the output voltage was reduced. Low-pass filters only “pass” lower frequencies to the output with little attenuation [cut-off] and higher frequencies are significantly attenuated (i.e., not “passed”). The opposite is true for high-pass filters: the lower frequencies resulted in reduced output voltage. Higher frequencies are now “passed” to the output with little attenuation and lower frequencies are now “not-passed” and are more attenuated [cut-off]. Additionally, using the function generator as an input signal for the Arduino’s Digital pin 11 created a tone for the speaker. The sweep button on the generator will be useful when a range of frequencies is to be swept across.

For the future, it may be useful to use a RC filter to filter out high frequencies or low frequencies, respectively, depending on the situation. Additionally, using the technique covered in Question 10 may become useful because some sensors may not report their measurements as analog or digital values, and instead as pulse lengths and duty cycles.

References

Furman, B. (2014). *ME 106 Lab Manual*. San Jose, CA: San Jose State University.

Alciatore, D. G. & Histand, M. B. (2011). *Introduction to Mechatronics and Measurement Systems* (4th Ed.). New York, NY: McGraw-Hill.

Appendix A: Pitches.h Source Code

```
/*************************************************
 * Public Constants
 *************************************************/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
```

```
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```

Appendix B: Source Code for Question 8

```
/*
 *  Hello. This program was developed by Patrick Barrera and Jimmy He for a
laboratory project in one of
 *  the upper division college courses at San Jose State University:
Mechanical Engineering 106:
 *  Fundamentals of Mechatronics. This is Lab 3 "RC Filters and Basic Timer
Functionality", Question 8.
 *  The program generates a sweep tone on the speaker from a minimum of 100
Hz to a maximum of 15,000 Hz
 *  in 5 seconds.
 */


#include "pitches.h"



// I) Variables
/* Pin Assignments: Declare and Initialize Input, Output, and Variables
 */
int hertz = 100;



// II) setup() is analogous to Constructors
/* Configure Pins. Set up Input and Output with the setup() function.
 *    pinMode(12, INPUT); function initializes pin 12 (Button0) to INPUT
 *    pinMode(SPEAKER, OUTPUT); function initializes the SPEAKER (pin 5) to
OUTPUT
 *    pinMode(SW0, INPUT_PULLUP); function initializes switch 0 (pin 12) to
INPUT_PULLUP.
 *       INPUT_PULLUP initializes switch 0 to default at 0 Volts.
 *    MyServo.attach(10); Attaches the servo on pin 10 to the servo object
 */
void setup()
{
  pinMode(11, OUTPUT);
}



// III) loop() is analogous to main()
/* Loop Forever
 */
void loop()
{
  for(hertz = 10; hertz <= 15000; hertz+=50)
  {
    //int noteDuration = 1000 / noteDurations[hertz];
    tone(11, hertz);
    //digitalWrite(11, HIGH);
    delay(15);
  }
```

```
  delay(15);

  for(hertz = 15000; hertz >= 10; hertz-=50)
  {
    tone(11, hertz);
    //digitalWrite(11, HIGH);
    delay(15);
  }

  digitalWrite(11, LOW);

}
```

Appendix C: Source Code for Question 9

```
/*
 *  Hello. This program was developed by Patrick Barrera and Jimmy He for a laboratory project in one of
 *  the upper division college courses at San Jose State University: Mechanical Engineering 106:
 *  Fundamentals of Mechatronics. This is Lab 3 "RC Filters and Basic Timer Functionality", Question 9.
 *  The program generates a sweep tone on the speaker from a minimum of 100 Hz to a maximum of 15,000 Hz
 *  in 5 seconds, times the sweep sequence, and prints it to the Serial Monitor.
 */



#include "pitches.h"



// I) Variables
/* Pin Assignments: Declare and Initialize Input, Output, and Variables
 */
int hertz = 100;
unsigned long time;



// II) setup() is analogous to Constructors
/* Configure Pins. Set up Input and Output with the setup() function.
 *    pinMode(12, INPUT); function initializes pin 12 (Button0) to INPUT
 *    pinMode(SPEAKER, OUTPUT); function initializes the SPEAKER (pin 5) to OUTPUT
 *    pinMode(SW0, INPUT_PULLUP); function initializes switch 0 (pin 12) to INPUT_PULLUP.
 *       INPUT_PULLUP initializes switch 0 to default at 0 Volts.
 *    MyServo.attach(10); Attaches the servo on pin 10 to the servo object
 */
void setup()
{
  pinMode(11, OUTPUT);

  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}


// III) loop() is analogous to main()
/* Loop Forever
 */
void loop()
{
  time = millis();

  for(hertz = 10; hertz <= 15000; hertz+=50)
```

```
    {
      //int noteDuration = 1000 / noteDurations[hertz];
      tone(11, hertz);
      //digitalWrite(11, HIGH);
      Serial.println(time);
      delay(15);
    }

    delay(15);

    time = 0;

    for(hertz = 15000; hertz >= 10; hertz-=50)
    {
      tone(11, hertz);
      //digitalWrite(11, HIGH);
      delay(15);
    }

    digitalWrite(11, LOW);

}
```

Appendix D: Source Code for Question 10

```
/*
 *  Hello. This program was developed by Patrick Barrera and Jimmy He for a
laboratory project in one of
 *  the upper division college courses at San Jose State University:
Mechanical Engineering 106:
 *  Fundamentals of Mechatronics. This is Lab 3 "RC Filters and Basic Timer
Functionality", Question 10.
 *  This program uses the pulseIn() function to time the length of the high
portion of the signal pulses
 *  from the function generator and prints it out on the Serial Monitor. The
Arduino takes in an input
 *  signal from the function generator via Digital pin 11 (and ground) on the
Arduino.
 */



#include "pitches.h"



// I) Variables
/* Pin Assignments: Declare and Initialize Input, Output, and Variables
 */
int hertz = 100;
unsigned long duration;



// II) setup() is analogous to Constructors
/* Configure Pins. Set up Input and Output with the setup() function.
 *    pinMode(12, INPUT); function initializes pin 12 (Button0) to INPUT
 *    pinMode(SPEAKER, OUTPUT); function initializes the SPEAKER (pin 5) to
OUTPUT
 *    pinMode(SW0, INPUT_PULLUP); function initializes switch 0 (pin 12) to
INPUT_PULLUP.
 *       INPUT_PULLUP initializes switch 0 to default at 0 Volts.
 *    MyServo.attach(10); Attaches the servo on pin 10 to the servo object
 */
void setup()
{
  // take in input signal from function generator
  pinMode(11, INPUT);

  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}



// III) loop() is analogous to main()
/* Loop Forever
 */
```

```
void loop()
{
  duration = pulseIn(11, HIGH);
  Serial.println(duration);
}
```