E101 Final Project Report

Team: ACES

Team Members: Audrey Gruian, Caiya Coggshall, Erin Wang, and Shreya Jampana

II. Background (Brief description of project tasks. Short summary of design calculations for the Butterworth and Modified Comb filters used from HW8, do NOT rederive)

4th order Butterworth filter:

The transfer function for the 4th order Butterworth filter was derived to be:

$$H(s) = \frac{\omega_c^4}{(s^2 - 2\omega_c cos(\frac{5\pi}{8})s + \omega_c^2)(s^2 - 2\omega_c cos(\frac{7\pi}{8})s + \omega_c^2)} \; where \; \omega_c = 2\pi f_p = 20\pi$$

To derive this, we used the transfer function for a low pass butterworth filter, which was given to us as

$$H(s)H(-s) = \frac{1}{1+(\frac{s}{jwc})^{2N}} .$$

We found an equation for the 8 poles for our 4th order filter, and after plotting them on a circle in the s-plane, chose 4 poles on the left half semicircle (corresponding to the poles of H(s)) to construct a stable, causal filter. By using these four poles as the denominator, and converting them to sinusoids, we arrived at the following transfer function:

$$H(s) = \frac{c}{(s^2 - 2\omega_c cos(\frac{5\pi}{8})s + \omega_c^2)(s^2 - 2\omega_c cos(\frac{7\pi}{8})s + \omega_c^2)}.$$

To find the value of the constant c, we used the fact that the pass band had to have a magnitude of 1, giving us a value of $c = w_c^4$.

Seeing as the 4th order butterworth filter had to be implemented with a two stage "Sallen-Key" circuit, we chose the standard resistors in the circuit to design the filter for a passband edge frequency of 10Hz. Using the given capacitor values and matching our transfer function to that of the "Sallen-Key" circuit, we calculated our standard resistor values to be R1 = 47 kΩ, R2 = 270 kΩ for the first circuit and R1 = 27 kΩ, R2 = 100 kΩ.

Modified Comb Filter:

The transfer function for the Modified Comb filter was found to be:

$$H_{mod}[n] = 0.0382(1 + 2.6180z^{-1} + 4.2361z^{-2} + 5.2361z^{-3} + 5.2361z^{-4} + 4.2361z^{-5} + 2.6180z^{-6} + z^{-7})$$

To derive this, we used the transfer function for a comb filter, which was given to us as:

$$H_{mod}(z) = b_0(1 - z^{-M})$$

where M is the number of zeros. We simplified the transfer function for 10 zeros. We were told to remove three zeros

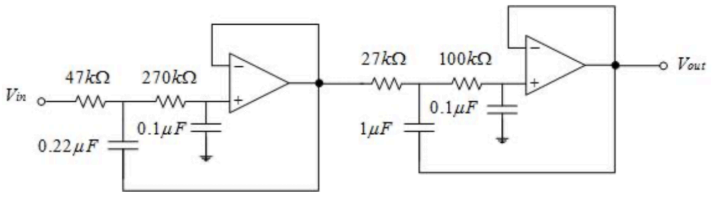
which were $z = e^{j0}, e^{\pm j\frac{\pi}{5}}$. This creates a passband at $\hat{w} = \pm\frac{\pi}{5}$. To obtain the modified comb filter we divided out the three zeros by multiplying our transfer function by

$$\frac{z^8}{(z-0)\,(z-\frac{\pi}{5})(z+\frac{\pi}{5})}.$$

After a lot of computation, we arrived at the final modified comb filter, written out above.

Circuit Design:

We made a two stage sallen-key circuit using the following provided diagram:



We used op amp A and B on the quad op amp LMC6484 to make the following circuit:

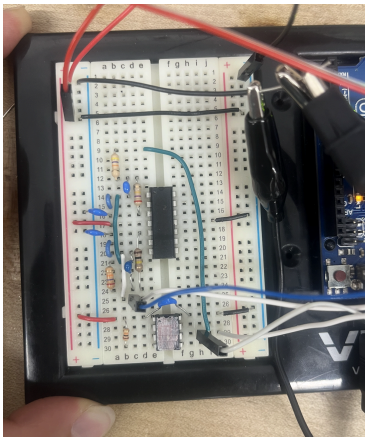


Figure 1 (left): Breadboard layout of low pass analog filter

Originally we received a PIC chip that produced a 10KHz signal, and our circuit was railing out due to an ungrounded component but we were able to troubleshoot using an input pulse train on the oscilloscopes wave generator. We created the circuit as seen in Figure X, where the blue connection is to the Arduino's A1 port, the white connection is to the A0 port, and the red/black are 5V/GND.

III. Experimental Results (Experimental time and frequency plots)

Note: Windowing is discussed in Analysis of Results.

Analog Filter

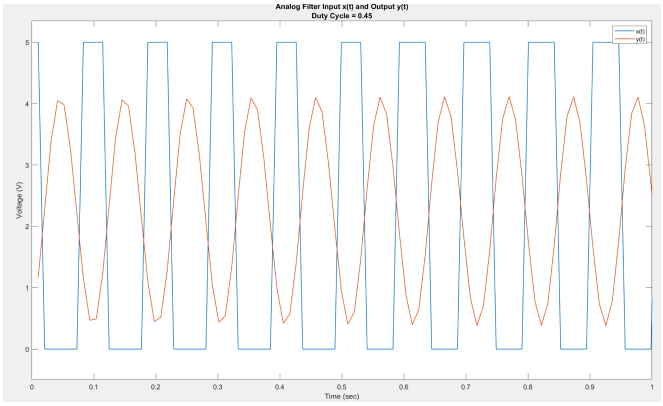


Figure 2: Voltage vs. time graph of analog filter input (x[n]) and output (y[n]) with duty cycle 0.45. Input signal oscillates between 0 and 5 V, while output resembles a sinusoid of amplitude 4 V. The duty cycle was calculated by measuring the percentage of the time the pulse train was "on," which was 46.8 ms for a period of 104 ms, leading to a duty cycle of 46.8/104 = 0.45.

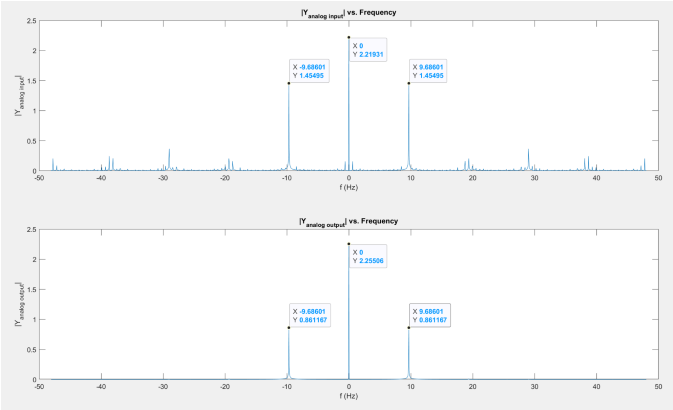


Figure 3: Analog input (top) and output (output) frequency plots after Fourier Transform. Signals visible in the input graph have been attenuated, and noise has been removed. Input |Y| = 2.21931 at f = 0 Hz, and |Y| = 1.45495 at f = 10 Hz and -10Hz. Output |Y| = 2.25506 at f = 0 Hz, and |Y| = 0.861167 at f = 10 Hz and -10Hz.

Digital Filter

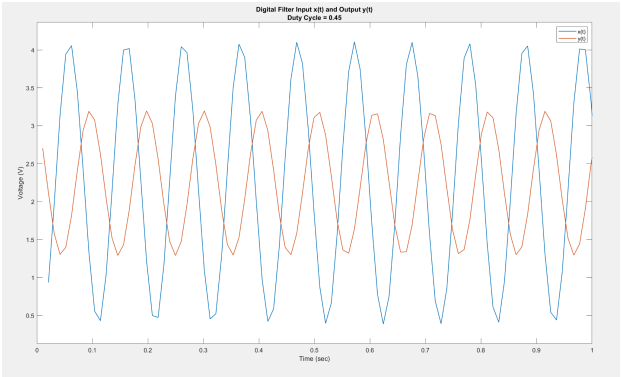


Figure 4: Voltage vs. time graph of digital filter input and output with duty cycle 0.45. Input signal oscillates between approximately 0 and 4 V, while output resembles a sinusoid of amplitude 2 V, centered around 2 V.

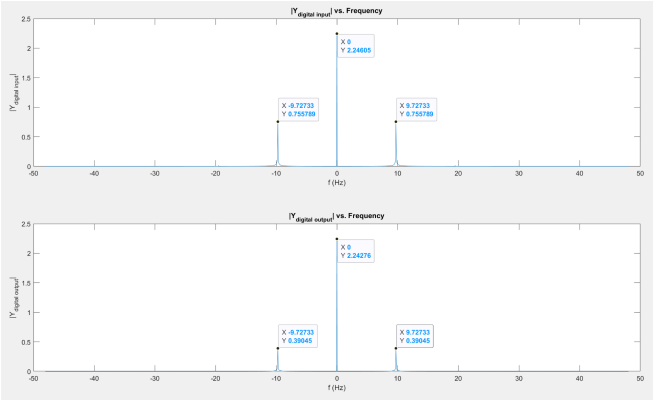


Figure 5: Digital input (top) and output (output) frequency plots after Fourier Transform. Signals visible in the input graph have been attenuated, and some noise has been removed. Input |Y| = 2.24605 at f = 0 Hz, and |Y| = 0.755789 at f = 10 Hz and -10Hz. Output |Y| = 2.24276 at f = 0 Hz, and |Y| = 0.39045 at f = 10 Hz and -10Hz.

IV. Analysis of Results (Details of signal processing methods used, including frequency response function of both the analog filter and digital filter or relevant calculations)

Hanning Window Implementation

When taking the Fourier Transform of the data, we assume that our data is perfectly periodic when this may not actually be the case. This can cause some irregularities in sampling, where one period starts at a different point than where the previous period ended, causing discontinuity at the data's edges. To fix this, we decided to implement a Hanning window, which would ensure that the beginning and end of the data approaches 0. When implementing a Hanning window, we need to account for the energy reduction from the window by multiplying by the length of the data set and dividing by the sum of the window.

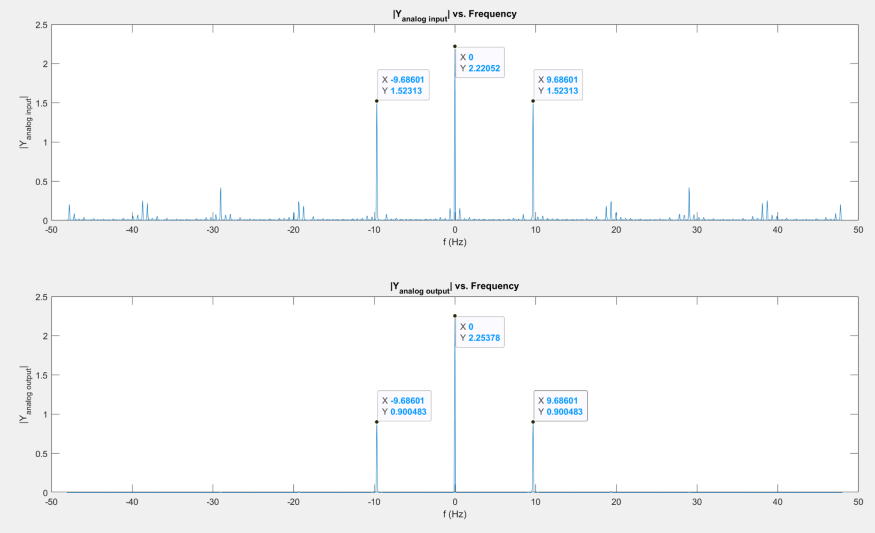


Figure 6: Hanning window on analog filter input and output. Input |Y| = 2.22052 at f = 0 Hz, and |Y| = 1.52313 at f = 10 Hz and -10Hz. Output |Y| = 2.25378 at f = 0 Hz, and |Y| = 0.900483 at f = 10 Hz and -10Hz.

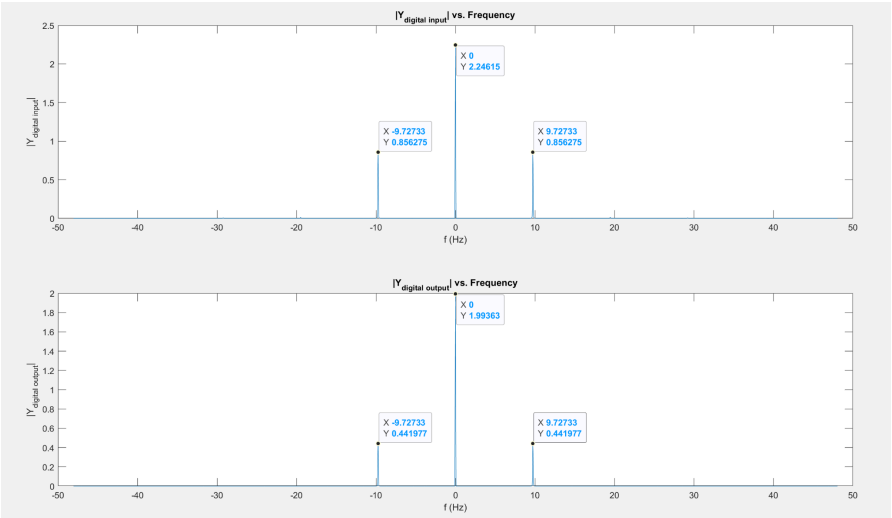


Figure 7: Digital Hanning window implementation for input and output. Input |Y| = 2.24615 at f = 0 Hz, and |Y| = 0.856275 at f = 10 Hz and -10Hz. Output |Y| = 1.99363 at f = 0 Hz, and |Y| = 0.441977 at f = 10 Hz and -10Hz.

Comments on Hanning Window results:

Adding in the Hanning Window did change the Fourier series coefficients slightly and most notably for the analog input signal/the impulse train. This makes sense because the pulse train jumps from 0 to 5V and thus the discontinuity of the sampled edges are more likely to cause errors in the Fourier transform compared to the analog output or digital output sinusoidal signals which are likely to have a smaller discontinuity at the sampling edges.

Measured Fourier series coefficients of analog filter inputs and outputs

Frequency (Hz)	No Window	Hanning Window
Impulse Train/Analog Input		
-9.68601	1.45945	1.52313
0	2.21931	2.22052
9.68601	1.45945	1.52313
Analog Output		
-9.68601	0.866167	0.900483
0	2.25506	2.25378
9.68601	0.866167	0.900483
Ratio		
-9.68601	0.593	0.591
0	1.016	1.015
9.68601	0.593	0.591

Measured Fourier series coefficients of digital filter inputs and outputs

Note: We separated Analog Output and Digital Input Fourier Series coefficients because we used different sets of data to analyze the Analog Input/Output and the Digital Input/Output which leads to slightly different FS coefficients.

Frequency (Hz)	No Window	Hanning Window
Digital Input		
-9.72733	0.755789	0.856275
0	2.24605	2.24615
9.72733	0.755789	0.856275
Digital Output		
-9.72733	0.39045	0.44197

0	2.24276	1.99363
9.72733	0.39045	0.44197
Ratio		
-9.72733	0.271	0.516
0	0.967	0.971
9.72733	0.271	0.516

FRFs and Estimation of FS Coefficients for Analog and Digital Filter

Analog Filter FRF

4th order butterworth filter:

$$H(s) = \frac{\omega_c^4}{(s^2 - 2\omega_c \cos(\frac{3\pi}{8})s + \omega_c^2)(s^2 - 2\omega_c \cos(\frac{7\pi}{8})s + \omega_c^2)}$$

used: $-2\cos(\frac{7\pi}{8}) = (R_1 + R_2)' \cdot C_2' \cdot \omega_c$

$-2\cos(\frac{5\pi}{8}) = (R_1 + R_2)'' \cdot C_2'' \cdot \omega_c$

where $R_1' = 46.6\,k\Omega$, $R_2' = 247.6\,k\Omega$

$R_1'' = 26.6\,k\Omega$, $R_2'' = 100\,k\Omega$

plugged into matlab to get bode plot and extract expected ratio:

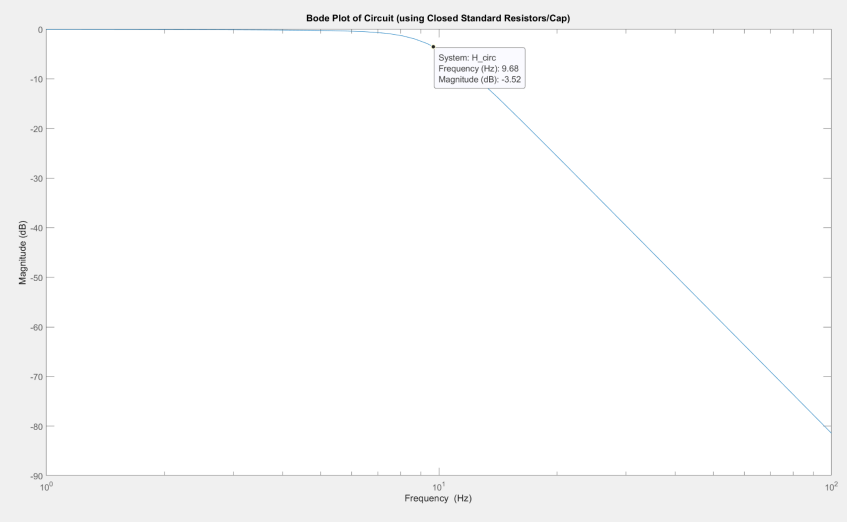


Figure 8: Bode plot of circuit using closed standard resistors/capacitors, with conversion -3.52 dB = 20log(Gain), and gain @ ±9.68 Hz = 0.67

Digital Filter FRF

$$H_{mod,comb}(z) = 0.0382\left(1 + 2.618z^{-1} + 4.236z^{-2} + 5.236z^{-3} + 5.236z^{-4} + 4.236z^{-5} + 2.618z^{-6} + z^{-7}\right)$$

replace z with $e^{j\hat{\omega}}$:

$$H_{mod,comb}(e^{j\hat{\omega}}) = 0.0382\left(1 + 2.618e^{-j\hat{\omega}} + 4.236e^{-2j\hat{\omega}} + 5.236e^{-3j\hat{\omega}} + 5.236e^{-4j\hat{\omega}} + 4.236e^{-5j\hat{\omega}} + 2.618e^{-6j\hat{\omega}} + e^{-7j\hat{\omega}}\right)$$

changing $\hat{\omega}$ to ω using $\hat{\omega} = \omega T$, so $\omega = \frac{\hat{\omega}}{T} = 100\hat{\omega}$

plugged into matlab to get bode plot and extract expected ratio:

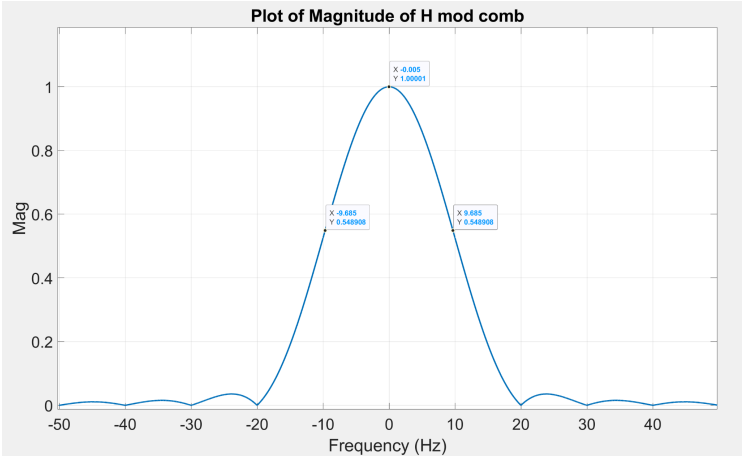


Figure 9: Magnitude of $H_{\text{mod comb}}$

Expected ratio at 0 Hz: 1.0001

Expected ratio at 9.685 Hz and -9.685 Hz: 0.548908

V. Conclusions (Interpretations of results and comparison of measured and expected filter performance)

A single Sallen-Key stage implements a second-order active low pass filter. If you had a passive filter you would create a gain that is unable to be amplified. You can use multiple stages of an op amp to better use higher-order filters. Since we had a 4th order Butterworth filter, this was most appropriate.

Aliasing happens when high frequency components fold into the passband as also noted in the sampling theorem. Low-pass Sallen-Key filters are usually used as anti-aliasing filters to remove frequencies above $\frac{f_s}{2}$.

Estimated versus measured (Windowed) ratios for Analog Filter

Frequency (Hz)	Estimated	Measured
-9.68601	0.67	0.592
0	1.000	1.016
9.68601	0.67	0.592

Possible sources of error:

Error in an analog filter may be due to several sources. Because resistors and capacitors have manufacturing tolerances, we can expect any output of our circuit to vary at least slightly from what we expect. Oscilloscope accuracy may also alter the results of the filter from expected values. Additional factors like temperature, age, and fluctuations in power supply could have impacted our results.

Estimated versus measured (Windowed) ratios for Digital Filter

Frequency (Hz)	Estimated	Measured
-9.68283	0.548908	0.517
0	1.0001	0.9985
9.68283	0.548908	0.517

Possible sources of error:

Digital filters have well-known sources of error. The filters are subject to noise, such as from circuit components, code, or white noise. Roundoff error may occur when measurements are rounded before or during calculations, or due to the finite word length constraints of the filter. However, our measured Fourier series ratio matched quite closely with our estimate.

Appendix A. Matlab Code

Analog Filter Code

```
%% Analog Filter

% Read in data fron AnalogFilter txt file
analogdata = readtable('AnalogFilterTest3');

% Create data columns with conversion factor
analogdata.Var3 = (analogdata.Var1)/204.6;      % Input to analog filter x(t)
analogdata.Var4 = (analogdata.Var2)/204.6;      % Output of analog filter y(t)

% Create sample vector
n = (1:length(analogdata.Var1));

% Create time vector using sample rate of 0.0104 seconds per sample
Ts = 0.0104;
Fs = 1/Ts;
time = (Ts)*n;

% Plot x(t) and y(t) against t
figure
plot(time,analogdata.Var3, LineWidth=1)
hold on
plot(time, analogdata.Var4, LineWidth=1)

legend("x(t)", "y(t)")
xlabel("Time (sec)")
ylabel("Voltage (V)")
title("Analog Filter Input x(t) and Output y(t)" + newline + "Duty Cycle = 0.45")
xlim([0 1])
ylim("padded")
```

Code for analog input and output, duty cycle 0.45

Fourier Transform Code for Analog Filter

```
%% Fourier Series Analog Filter
[Yanalogin, fanalogin] = fdomain(analogdata.Var3, Fs);

figure
subplot(2,1,1)
plot(fanalog,abs(Yanalogin))
xlabel('f (Hz)')
ylabel('|Y_{analog input}|')
title('|Y_{analog input}| vs. Frequency')


[Yanalogout, fanalogout] = fdomain(analogdata.Var4, Fs);

subplot(2,1,2)
plot(fanalog,abs(Yanalogout))
xlabel('f (Hz)')
ylabel('|Y_{analog output}|')
title('|Y_{analog output}| vs. Frequency')
```

Code for fourier transform of analog filter input and output

Digital Filter Code

```
%% Digital Filter
% Read in data from DigitalFilter txt file
digitaldata = readtable('DigitalFilterTest1');

% Create data columns with conversion factor
digitaldata.Var3 = (digitaldata.Var1)/204.6;      % Input to digital filter x(t)
digitaldata.Var4 = ((digitaldata.Var2)/204.6) + 1.5;      % Output of digital filter y(t)

% Create sample vector
n1 = (1:length(digitaldata.Var1));
time1 = (0.0104)*n1;

% Plot x[n] and y[n] against n
figure
plot(time1,digitaldata.Var3, LineWidth=1)
hold on
plot(time1, digitaldata.Var4, LineWidth=1)

legend("x(t)", "y(t)")
xlabel("Time (sec)")
ylabel("Voltage (V)")
title("Digital Filter Input x(t) and Output y(t)" + newline + "Duty Cycle = 0.45")
xlim([0 1])
ylim("padded")
```

Code for voltage vs. time for digital input and output with duty cycle 0.45

Fourier Transform of Digital Filter

```
%% Fourier Series Digital Filter
lengthdig = length(digitaldata.Var3)-1;
windowdig = hann(lengthdig);
correctionfactordig = (lengthdig/sum(windowdig));


Var3hanndig = windowdig.*digitaldata.Var3(2:end);
[Ydigitalin, fdigitalin] = fdomain(digitaldata.Var3(2:end), Fs);
Ydigitalinfix = correctionfactordig*Ydigitalin;

figure
subplot(2,1,1)
plot(fdigitalin,abs(Ydigitalin))
xlabel('f (Hz)')
ylabel('|Y_{digital input}|')
title('|Y_{digital input}| vs. Frequency')



Var4hanndig = windowdig.*digitaldata.Var4(2:end);
[Ydigitalout, fdigitalout] = fdomain(digitaldata.Var4(2:end), Fs);
Ydigitaloutfix = correctionfactordig*Ydigitalout;

subplot(2,1,2)
plot(fdigitalout,abs(Ydigitalout))
xlabel('f (Hz)')
ylabel('|Y_{digital output}|')
title('|Y_{digital output}| vs. Frequency')
```

Code for fourier transform of digital filter input and output

Hanning Window Code

```
%% Fourier Series Analog Filter
lengthanalog = length(analogdata.Var3);
windowanalog = hann(lengthanalog);
correctionfactoranalog = (lengthanalog/sum(windowanalog));

Var3hannanalog = windowanalog.*analogdata.Var3;
[Yanalogin, fanalogin] = fdomain(Var3hannanalog, Fs);
Yanaloginfix = correctionfactoranalog*Yanalogin;

figure
subplot(2,1,1)
plot(fanalog,abs(Yanaloginfix))
xlabel('f (Hz)')
ylabel('|Y_{analog input}|')
title('|Y_{analog input}| vs. Frequency')

Var4hannanalog = windowanalog.*analogdata.Var4;
[Yanalogout, fanalogout] = fdomain(Var4hannanalog, Fs);
Yanalogoutfix = correctionfactoranalog*Yanalogout;

subplot(2,1,2)
plot(fanalog,abs(Yanalogoutfix))
xlabel('f (Hz)')
ylabel('|Y_{analog output}|')
title('|Y_{analog output}| vs. Frequency')
```

Code for Hanning window fourier transform of analog filter input and output

```
%% Fourier Series Digital Filter
lengthdig = length(digitaldata.Var3)-1;
windowdig = hann(lengthdig);
correctionfactordig = (lengthdig/sum(windowdig));


Var3hanndig = windowdig.*digitaldata.Var3(2:end);
[Ydigitalin, fdigitalin] = fdomain(Var3hanndig, Fs);
Ydigitalinfix = correctionfactordig*Ydigitalin;

figure
subplot(2,1,1)
plot(fdigitalin,abs(Ydigitalinfix))
xlabel('f (Hz)')
ylabel('|Y_{digital input}|')
title('|Y_{digital input}| vs. Frequency')



Var4hanndig = windowdig.*digitaldata.Var4(2:end);
[Ydigitalout, fdigitalout] = fdomain(Var4hanndig, Fs);
Ydigitaloutfix = correctionfactordig*Ydigitalout;

subplot(2,1,2)
plot(fdigitalout,abs(Ydigitaloutfix))
xlabel('f (Hz)')
ylabel('|Y_{digital output}|')
title('|Y_{digital output}| vs. Frequency')
```

Code for Hanning window fourier transform of digital filter input and output

Appendix B. Arduino Code

AnalogInput2.ino

AnalogInput2.ino

```
/*
Reads 2 analog input pins (range 0-1023) and
prints the results to the serial monitor.
We will use this code to read the input and output.

The circuit:
* analog signal connected to analog pin A0.
* analog signal connected to analog pin A1.
*/

// These constants won't change. They're used to give names
// to the pins used:
const int inputPin0 = A0; // Analog input pin 0
const int inputPin1 = A1; // Analog input pin 1

void setup() {
Serial.begin(9600);
}

void loop() {
// analogRead values from 0 to 1023 (10 bit ADC)
int sensorVal0 = analogRead(inputPin0);
int sensorVal1 = analogRead(inputPin1);
// print the results to the serial monitor:
Serial.print(sensorVal0);
Serial.print(" ");
Serial.println(sensorVal1);
delay(10); // sample time ms
}
```

DigitalFilter.ino

Digital_Filter.ino

```
/*
Digital FIR Filter
Reads analog input pin (range 0-1023) and
E101 2024 Final Project Analog and Digital Filters
prints the filtered output to the serial monitor
The circuit:
* analog signal connected to analog pin A1
The filter used in this template is a three point averaging filter.
You will need to adapt the program for a modified comb filter
*/
const int inputPin = A1; // Analog input pin
// filter coefficients
const float b0=0.0382;
const float b1=0.1;
const float b2=0.1618;
const float b3=0.2;
const float b4=0.2;
const float b5=0.1618;
const float b6=0.1;
const float b7=0.0382;

// initial values for internal signals
int xstore1 = 0;
int xstore2 = 0;
int xstore3 = 0;
int xstore4 = 0;
int xstore5 = 0;
int xstore6 = 0;
int xstore7 = 0;
int yn = 0;
void setup() {
Serial.begin(9600);
}

void loop() {
// analogRead value from pin A1 (0-1023)
int xn = analogRead(inputPin);

// difference equation for filter
yn= b0*xn + b1*xstore1 + b2*xstore2 + b3*xstore3 + b4*xstore4 + b5*xstore5 + b6*xstore6 + b7*xstore7;

// print to output
Serial.print(xn);
Serial.print(" ");
Serial.println(yn);

// update stored values
xstore7 = xstore6;
xstore6 = xstore5;
xstore5 = xstore4;
xstore4 = xstore3;
xstore3 = xstore2;
xstore2 = xstore1;
xstore1 = xn;
delay(10); // sample time ms
```