Name: MANSI UNIYAL
Roll No.: 19EE10039
Experiment 3
Title - Digital low pass FIR filter implementation on Arduino Hardware

---

Objective:
To Program ATMEGA328p in Arduino Uno to filter the input signal received in the ADC port of the microcontroller and gives a filtered output which we can observe in the serial plotter as a virtual output.

Requirements:
1. Arduino Uno Board
2. USB A to USB B cable
3. Arduino IDE

Methodology:
Build the required digital filter parameters in MATLAB and use the parameters in assembly code to program the ATMEGA328p microcontroller in Arduino.

Frequencies used in the design:-
FSampling = 2000Hz
Fcutoff = 50Hz

MATLAB code:-
```
%% Filter Specifications
M=32;
fs=2000;
fc=100;
wc=2*fc/fs;
h = fir1(M,wc);
h_fixed=fi(h, 1, 8, 7);
%% Plot filter gain vs frequency response
fvtool(h);
title('Designed filter');
fvtool(h_fixed);
title('Fixed point filter');
%% Input Signal
L=500; f1=50; f2=500;
sig=zeros(L,1);
for i=1:L
sig(i)=sin(2*pi*f1*i/fs)+sin(2*pi*f2*i/fs);
end
```

```
sig_fixed=fi(sig, 1, 8, 7);
%% Do convolution to get output signal
y=conv(sig,h);
y_fixed=conv(sig_fixed,h_fixed);
%% Plot input & output signal
figure(1)
subplot(2,2,1)
plot(sig)
title('Input Signal');
subplot(2,2,2)
plot(sig_fixed)
title('Fixed point Input Signal');
subplot(2,2,3)
plot(y)
title('Low Pass Filtered Signal');
subplot(2,2,4)
plot(y_fixed)
title('Fixed point Low Pass Filtered Signal');
%% Write fixed-point filter coefficients & input signal in hex format file1=fopen('Filter
Co-efficients from MATLAB.txt', 'w'); for i=1:1:length(h_fixed)
hh=h_fixed(i)*2^8;
if i<length(h_fixed)
fprintf(file1, '%d, ', hh);
else
fprintf(file1, '%d', hh);
end
end
fclose(file1);
file2=fopen('Input Signal Data from MATLAB.txt', 'w'); for i=1:length(sig_fixed)
si=sig_fixed(i)*2^8;
if i<length(sig_fixed)
fprintf(file2, '%d, ', si);
else
fprintf(file2, '%d', si);
end
end
fclose(file2);
```

This code generates two files that contain input signal data(50Hz) with noise(500Hz) and the filter parameters. Now we can write the C++ code in Arduino IDE for the ATMEGA328p microcontroller to perform the digital filtering on the provided input signal mixed with high-frequency noise.

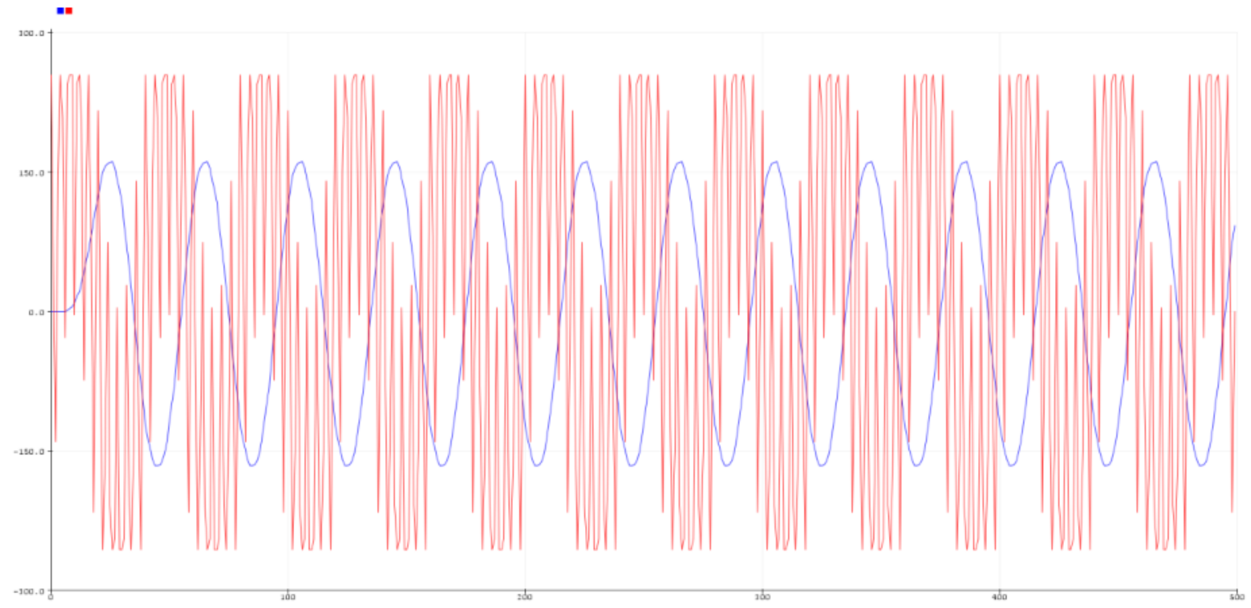
Code:

Arduino IDE code:-

```
int m = 500, n = 30;
int x[500] = {254, 80, -140, 150, 254, 208, -28, 244, 254, 254, -4, 244, 254, 208, -74,
150, 254, 80, -216, 0, 216, -80, -256, -150, 74, -208, -256, -244, 4, -256, -256, -244, 28,
-208, -256, -150, 140, -80, -256, 0, 254, 80, -140, 150, 254, 208, -28, 244, 254, 254, -4,
244, 254, 208, -74, 150, 254, 80, -216, 0, 216, -80, -256, -150, 74, -208, -256, -244, 4,
-256, -256, -244, 28, -208, -256, -150, 140, -80, -256, 0, 254, 80, -140, 150, 254, 208,
-28, 244, 254, 254, -4, 244, 254, 208, -74,
150, 254, 80, -216, 0, 216, -80, -256, -150, 74, -208, -256, -244, 4, -256, -256, -244, 28,
-208, -256, -150, 140, -80, -256, 0, 254, 80, -140, 150, 254, 208, -28, 244, 254, 254, -4,
244, 254, 208, -74, 150, 254, 80, -216, 0, 216, -80, -256, -150, 74, -208, -256, -244, 4,
-256, -256, -244, 28, -208, -256, -150, 140, -80, -256, 0, 254, 80, -140, 150, 254, 208,
-28, 244, 254, 254, -4, 244, 254, 208, -74, 150, 254, 80, -216, 0, 216, -80, -256, -150,
74, -208, -256, -244, 4, -256, -256, -244, 28, -208, -256, -150, 140, -80, -256, 0, 254,
80, -140, 150, 254, 208, -28, 244, 254, 254, -4, 244, 254, 208, -74, 150, 254, 80, -216,
0, 216, -80, -256, -150, 74, -208, -256, -244, 4, -256, -256, -244, 28, -208, -256, -150,
140, -80, -256, 0, 254, 80, -140, 150, 254, 208, -28, 244, 254, 254, -4, 244, 254, 208,
-74, 150, 254, 80, -216, 0, 216, -80, -256, -150, 74, -208, -256, -244, 4, -256, -256,
-244, 28, -208, -256, -150, 140, -80, -256, 0, 254, 80, -140, 150, 254, 208, -28, 244,
254, 254, -4, 244, 254, 208, -74, 150, 254, 80, -216, 0, 216, -80, -256, -150, 74, -208,
-256, -244, 4, -256, -256, -244, 28, -208, -256, -150, 140, -80, -256, 0, 254, 80, -140,
150, 254, 208, -28, 244, 254, 254, -4, 244, 254, 208, -74, 150, 254, 80, -216, 0, 216,
-80, -256, -150, 74, -208, -256, -244, 4, -256, -256, -244, 28, -208, -256, -150, 140, -80,
-256, 0, 254, 80, -140, 150, 254, 208, -28, 244, 254, 254, -4, 244, 254, 208, -74, 150,
254, 80, -216, 0, 216, -80, -256, -150, 74, -208, -256, -244, 4, -256, -256, -244, 28,
-208, -256, -150, 140, -80, -256, 0, 254, 80, -140, 150, 254, 208, -28, 244, 254, 254, -4,
244, 254, 208, -74, 150, 254, 80, -216, 0, 216, -80, -256, -150, 74, -208, -256, -244, 4,
-256, -256, -244, 28, -208, -256, -150, 140, -80, -256, 0, 254, 80, -140, 150, 254, 208,
-28, 244, 254, 254, -4, 244, 254, 208, -74, 150, 254, 80, -216, 0, 216, -80, -256, -150,
74, -208, -256, -244, 4, -256, -256, -244, 28, -208, -256, -150, 140, -80, -256, 0, 254,
80, -140, 150, 254, 208, -28, 244, 254, 254, -4, 244, 254, 208, -74, 150, 254, 80, -216,
0};
int h[33] = {0, 0, 0, 0, 0, 0, 0, 2, 4, 6, 10, 14, 16, 20, 24, 26, 26, 26, 24, 20, 16, 14, 10, 6,
4, 2, 0, 0, 0, 0, 0, 0, 0}; //lowpass
long int y = 0;
void setup() {
// initialize serial communication with 9600 bits per second:
Serial.begin(115200);
//convolution
for (int i=0; i<m+n; i++){
y = 0;
for (int j=0; j<n; j++){
if ((i-j)>=0 && (i-j) < m)
y = y + (h[j]*x[i-j])/256;
}
Serial.print(y);
Serial.print(", ");
if (i < m)
Serial.println(x[i]);
delayMicroseconds(500); //0.5ms delay
}
}
void loop() {
// put your main code here, to run repeatedly:
}
```

Results:

The simulated output as observed in the lab is produced. The Arduino output can be observed in the serial plotter, which shows a filtered output signal of 50Hz frequency with very little noise in the output.



In the above diagram, the blue curve is the input signal and the red curve is the 20Hz filtered output signal. We can see that the input signal has both low and high frequencies while the output signal only has the low-frequency component. Also, we can observe a little delay in the input and output signal as well as a little clipping in the peak values of the input signal.
The observation confirms that the low pass filter implemented on the Arduino board is working correctly as only waves of low frequencies are allowed to pass through and appear in the output of the filter.

Discussions:

As we can observe, there is a little clipping in the input signal near the peak values. As we use

8-bit registers to store values, whenever the values of the input signal reach above 255 or below -256 the clipping occurs.
Also from observation, there is a little delay between input and output signal. Whenever a signal is passed through a filter it causes a delay in the filtered output which is termed a group delay. If the delay is constant for all frequencies then we can compensate for the delay with proper programming. Reducing the number of filter parameters reduces accuracy but it reduces the delay too.