# PROJECT REPORT

# OSCILLOSCOPE

## Aim:

Converting analog data values to discrete levels using ADC and applying a discrete filter on the sampled data, recreating the sound file again and checking the differences occurring in normal sound file vs filtered sound file.

## Testing with slide pot:

Before directly reading the audio file directly, we will test it with slide pot. First we download and install the driver and software of cool term serial monitor and make sure it works fine. Now test it by sliding the slide pot and check the values on the serial monitor.The output voltage values should vary between 0-3.3 volt.

## Reading Audio File:

Now get the .wav form of audio file which we want to read using wavesurfer software. Play the audio file and read it using audio jack connected to microcontroller. Store the values we get in a text file or any other format.

## Applying Filters:

We applied low pass, band pass and high pass filters on the sampled Data and analysed how the filtered data is varying in all the above cases by plotting the Frequency Response and Time Responses.Below is the code for generating filter and filtering the data using it and plotting the responses.

```
import numpy as np
import matplotlib.pyplot as plt
from math import *
f=open("Data.txt","r")
num_lines = sum(1 for line in f)
f.close()
g=open("Data.txt","r")
x=[]
for i in range(num_lines):
        a=(g.readline()[12:17]).split(',')[0]
        x.append(int(a))
g.close()

x = np.multiply(3.3/65536.0,x)
low_pass = [-0.000009896943711427464,0.00860145798960607,0.01698272049398045,0.02572381386847099,0.02846170315083110 6,0.020357838156729157,0.000739439
4200085718,-0.02448793215689427 2,-0.04345927582898785,
-0.04247887995856428,-0.01232867227491338,0.04614340031388637,0.12020502327835196,0.18870979201030966,0.2298410045175105,0.2298410045175105,0.18870979
201030966,0.12020502327835196,0.04614340031388637,
-0.01232867227491338,-0.04247887995856428,-0.04345927582898785,-0.02448793215689427 2,0.000739439420008571 8,0.020357838156729157,0.028461703150831106,0
```

```
.02572381386847099,0.01698272049398045,0.00860145798960607
,-0.00000989694371142746 4]

high_pass = [0.032114175662653136,-0.03596791862643143,0.05785576576281618,-0.07357322264645809,0.08929507930471871,-0.10470727594996831,0.11679294207
279844,-0.12491240241016631,0.1282961570925122,
-0.12424664375704891,0.11419417848500098,-0.09769719702446578,0.07490876538668502,-0.04677560838087435,0.01562207196030683 2,0.01562207196030683 2,-0.04
677560838087435,0.07490876538668502,-0.09769719702446578,
0.11419417848500098,-0.12424664375704891,0.1282961570925122,-0.12491240241016631,0.11679294207279844,-0.10470727594996831,0.08929507930471871,-0.07357
322264645809,0.05785576576281618,-0.03596791862643143,
0.032114175662653136]

band_pass = [-0.0016175661852355294,0.008223100877676252,0.001983582557256769,-0.0038418068308688443,-0.00015704726697886016,-0.014001067762420273,0.0
037970333555006 27,0.05235438769716051,
-0.02180052482778838,-0.10354494869444536,0.05898954729036707,0.14863608605298126,-0.10825963219804506,-0.1673911357344765,0.15106889202514392,0.15106
889202514392,-0.1673911357344765,-0.10825963219804506,
0.14863608605298126,0.05898954729036707,-0.10354494869444536,-0.02180052482778838,0.05235438769716051,0.003797033355500627,-0.014001067762420273,-0.00
015704726697886016,-0.0038418068308688443,0.001983582557256769
,0.008223100877676252,-0.0016175661852355294]

#  Plotting output signal in time domain after applying filters
y_low=np.convolve(x,low_pass) y_high=np.convolve(x,high_pass)
y_band=np.convolve(x,band_pass)

fig1 = plt.figure(1,figsize=(20,8))
A = 'Output signals from Various Filter'
fig1.canvas.set_window_title(A) plt.subplot(411)

plt.title("Input Signal") plt.plot(range(len(x)),x,'r')
plt.subplot(412) plt.title("Low Pass Output")
plt.ylabel('Voltage in (V)')
plt.plot(range(len(y_low)),y_low,'b') plt.subplot(413)



plt.title("High Pass Output")
plt.plot(range(len(y_high)),y_high,'g') plt.subplot(414)

plt.title("Band Pass Output") plt.xlabel('Time')
plt.plot(range(len(y_band)),y_band,'r') plt.tight_layout()


plt.show()

W1 = np.linspace(-pi,pi,num=len(x))
W2 = np.linspace(-pi,pi,num=len(y_low))
W3 = np.linspace(-pi,pi,num=len(y_band))
W4 = np.linspace(-pi,pi,num=len(y_high))
#  Finding magnitude response def
magnitude(h,w):
    H_r,H_i,H = 0.0,0.0,0.0 for i in
    range(len(h)):
        H_r += h[i]*cos(-1*w*i) H_i +=
        h[i]*sin(w*i)
    H = sqrt(H_r**2+H_i**2) return
    H
def mag_response(h,W):
    H_t=[]
    for j in W : H_t.append(magnitude(h,j))

    H_t = np.asarray(H_t) return
    H_t
#  Plotting Magnitude response of filters
H_low_pass=mag_response(low_pass,W1)
H_high_pass=mag_response(high_pass,W1)
H_band_pass=mag_response(band_pass,W1) fig2 =
plt.figure(2,figsize=(18,8))
B = 'Magnitude Response of Various Filter'
fig2.canvas.set_window_title(B) plt.subplot(311)

plt.title("High Pass Filter")
plt.plot(W1,H_high_pass,'r')
plt.subplot(312) plt.title("Low Pass
Filter")
plt.ylabel('Frequency response(Magnitude Response)')
plt.plot(W1,H_low_pass,'b')
plt.subplot(313)
plt.title("Band Pass Output")
plt.xlabel('Frequency(w)')
plt.plot(W1,H_band_pass,'g')
plt.tight_layout() plt.show()



#  Plotting magnitude response of output signal
H_x=mag_response(x,W1)
H_low=mag_response(y_low,W2)
H_band=mag_response(y_band,W3)
H_high=mag_response(y_high,W4)
plt.figure(3,figsize=(18,8))
```

```
plt.xlabel('Frequency(w)')
plt.ylabel('Frequency response(Magnitude Response)')
plt.subplot(411)
plt.title("Input Signal")
plt.plot(W1,np.multiply(20,np.log10(H_x)),'r')
plt.subplot(412)
plt.title("Low Pass Output")
plt.plot(W2,np.multiply(20,np.log10(H_low)),'b')
plt.subplot(413)
plt.title("High Pass Output")
plt.plot(W4,np.multiply(20,np.log10(H_high)),'g')
plt.subplot(414)
plt.title("Band Pass Output")
plt.plot(W3,np.multiply(20,np.log10(H_band)),'r')
plt.tight_layout()
plt.show()
```

## Some Improvements done by us:

We playback the Original and Filtered data by using pyAudio library of python and analysed how the generated sound will change when we apply appropriate filter to it (our aim here was to remove the electrical household noises which generally have frequency around 50Hz)

## Conclusion:

In this way we can sample an analog file and do audio processing  to remove noise and other unwanted frequencies. It can be extended further so as to plot the real time frequency and time response using software processing which can be used in Industrial application (like an app can be created which does live video recording with filters in real time).