MÜHENDİSLİK FAKÜLTESİ
FACULTY OF ENGINEERING
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ
DEPARTMENT OF ELECTRICAL AND
ELECTRONICS ENGINEERING

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T: +90 312 210 23 02
F: +90 312 210 23 04
ee@metu.edu.tr
www.eee.metu.edu.tr

EXPERIMENT 6. SYSTEM IDENTIFICATION WITH ADAPTIVE PROCESSING, DESIGN AND IMPLEMENTATION OF LMS FILTER

I. Introduction

In this experiment, adaptive filters are considered. Adaptive filters are used in many fields which require tracking the input statistics. Some examples of applications are system identification, equalization, linear predictive coding, spectrum estimation, signal detection, noise canceling, echo suppression, beamforming, etc. In this experiment, FIR adaptive filter structure is used together with the least mean square (LMS) adaptation algorithm for system identification. Evaluation of adaptive filters is done by considering factors such as rate of convergence, misadjustment, tracking performance, robustness, computational complexity, structure and numerical properties.

II. Preliminary Work

1) Read the following information on optimum filtering.

6.1. Adaptive Filtering: Least Mean Square, LMS, Algorithm

When there is sufficient statistical information, optimum solution for a variety of problems can be found by solving the Wiener-Hopf equations. Hence one can easily find the filter coefficients by

$$\mathbf{h}_{opt} = \mathbf{R}_x^{-1}\mathbf{r}_{dx}$$

An alternative for finding the optimum filter coefficients and optimum solution or output is to use an iterative algorithm that starts from an initial point and move towards the optimum in steps. One advantage of the adaptive approach is its ability to track the slowly changing statistics of the input signal. While there are a variety of adaptation algorithms, LMS algorithm is considered in this experiment due to its simplicity. Furthermore, the adaptive filter is assumed to have a transversal (or FIR) form. The structure of an adaptive filter is given below.

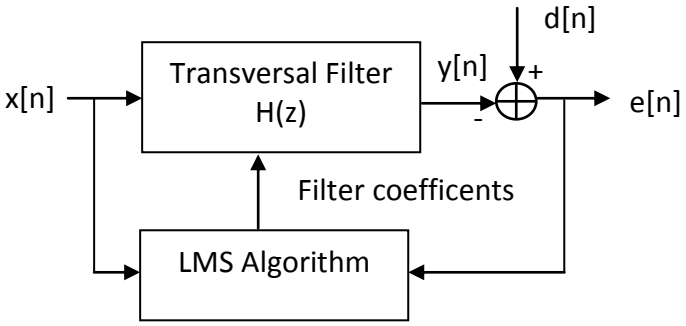

Fig. 1. Adaptive filter structure.

The input signal is filtered by H(z) and the adaptive filter output, y[n], is obtained. At each time-step, n, the error, e[n]=d[n]-y[n] is computed where d[n] is the desired response. LMS algorithm takes e[n] and the input, x[n], to compute the adaptive filter weights at the next iteration.

6.2. Derivation of the LMS algorithm

It is assumed that there are M filter coefficients which need to be set by the LMS algorithm. LMS algorithm tries to minimize the error between the desired response, d[n] and the adaptive filter output, y[n]. The cost function that is minimized is the sample error function. Hence LMS algorithm is a deterministic type of an adaptive filter. The cost function is given as,

$$J[n]=e^2[n]=(d[n]-y[n])^2=\left(d[n]-\sum_{k=0}^{M-1}h[k]x[n-k]\right)^2 \quad (1)$$

J[n] is a quadratic function of the adaptive weights, h[k], and it has a single minimum. At each iteration, J[n] is reduced by moving h[k] proportional to $-\frac{\partial J[n]}{\partial h_k}$. Hence the weight update equation is given as,

$$h_k[n+1]=h_k[n]-\hat{\mu}\frac{\partial J[n]}{\partial h_k} \quad (2)$$

At each time instant n, the kth coefficient's current value is changed and assigned to the n+1 time value. The derivative expression in (2) can be written as,

$$\frac{\partial J[n]}{\partial h_k}=\frac{\partial e^2[n]}{\partial h_k}=2e[n]\frac{\partial e[n]}{\partial h_k}=-2e[n]x[n-k] \quad (3)$$

Inserting (3) in (2), we obtain,

$$h_k[n+1]=h_k[n]+\mu e[n]x[n-k],\quad k=0,1,\ldots,M-1 \quad (4)$$

where $\mu=2\hat{\mu}$.

6.3. System Identification

Adaptive filters can be used to identify the impulse response of an unknown system and track its characteristics in time if it changes. This is possible if the input and output signals of the unknown system are available. Fig. 2 shows the use of adaptive filter for finding the impulse response of the system.

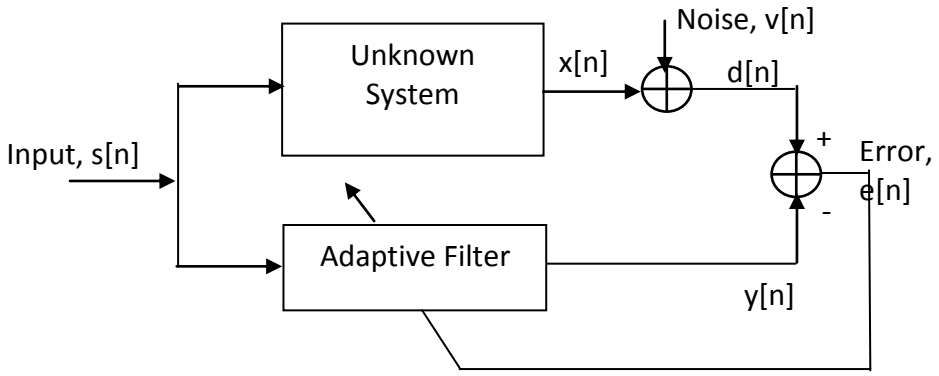

Fig. 2. System identification with an adaptive filter.

As it is seen in Fig. 2, unknown system output is x[n]. x[n] is noise corrupted by v[n] and the observable desired signal d[n] is obtained. When there is no noise, i.e., v[n]=0, d[n]=x[n]. In this case, adaptive filter converges more securely. The difference between d[n], and the output of the adaptive filter, y[n], is the error, e[n], which is used to update the adaptive filter coefficients. The adaptation causes y[n] to approach d[n] in time. A small step size ensures a stable response but convergence may be slow. In this case, *misadjustment* is expected to be small compared to a larger step size. A larger step size may lead faster convergence but *misadjustment* is larger and the response may be unstable leading to progressively larger output samples.

6.4. Interference and Noise Cancellation

Adaptive filters can be used for interference and noise cancellation. The advantage of adaptive filters is the adaptation to the time varying signal characteristics as opposed to other techniques. The structure for noise cancellation share the same idea as in system identification. Fig. 3 shows the structure of the noise cancellation system.

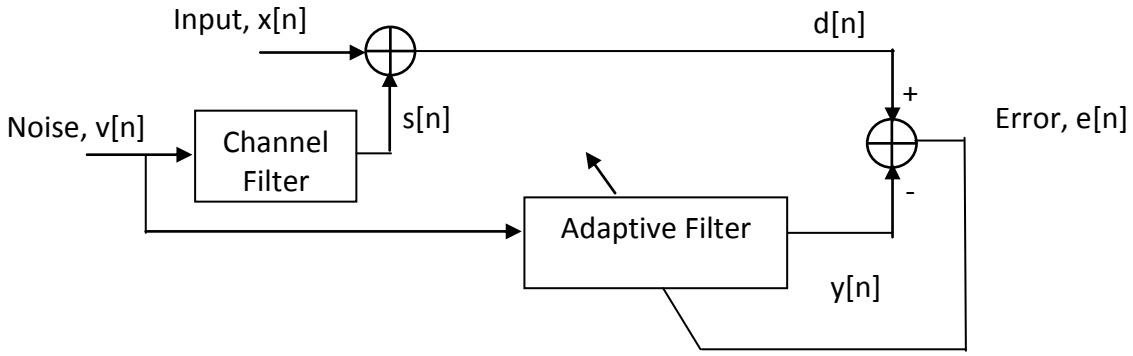

Fig. 3. Interference and noise cancellation system structure.

As it is seen in Fig. 3, desired signal x[n] is corrupted by the noise, s[n], which is the output of channel filter. The target is to clean d[n] from the noise by subtracting the adaptive filter output, y[n], from d[n]. The function of the adaptive filter is to converge to the channel filter coefficients such that it replicates the interference signal. As the adaptive filter converges, e[n] gets closer to x[n].

PART 1

6.5. MATLAB Programming Tasks

6.5.1. System Identification in MATLAB

In this part, the structure in Fig. 2 is implemented in MATLAB.

a) Generate a white Gaussian signal, s[n] with unit variance. Select a FIR filter with N coefficients. Filter the signal and obtain the output signal, x[n]. Add white Gaussian noise to the output signal and obtain d[n]. The noise standard deviation of the white Gaussian noise v[n] is a parameter.

b) Design an adaptive filter whose **parameters** are the **filter length**, **step size**, **input signal**, and the **error signal**. The output of the adaptive filter is y[n]. Input signal is s[n], error signal is e[n]=d[n]-y[n]. Use LMS algorithm for the adaptation of the filter coefficients.

c) The **program outputs** are the **adaptive filter coefficients (or weights)**, and **the least squares error**, i.e.

$$LSE = \frac{1}{K}\sum_{k=0}^{K-1} e^2[k]$$

d) Determine the converge speed for the selected step size by plotting $e^2[n]$ versus n.

e) Change the adaptive filter length to observe its effects. Explain your observations.

f) Change the FIR filter length, N and explain its effects.

g) Change the step size and explain its effect on the convergence speed as well as misadjustment which is the final error after convergence.

h) Change the standard deviation of noise, v[n], and explain your observations. Note that noise free case, v[n]=0, should return the best performance.

6.5.2. Noise Cancellation in MATLAB

In this part, the structure in Fig. 3 is implemented in MATLAB for noise removal.

a) Generate a white Gaussian signal, v[n] with unit variance. Select a FIR channel filter with N coefficients. Filter v[n] and obtain the output signal, s[n]. Add s[n] with the input signal, x[n] to obtain noise corrupted signal, d[n]. Note that ultimate target is to clear d[n] from noise to obtain x[n]. The noise standard deviation for v[n] is a parameter.

b) Design an adaptive filter whose **parameters** are the **filter length**, **step size**, **input signal**, and the **error signal**. The output of the adaptive filter is y[n]. Input signal is v[n], error signal is e[n]=d[n]-y[n]. Use LMS algorithm for the adaptation of the filter coefficients.

c) The program outputs are the adaptive filter coefficients (or weights), and the least squares error, i.e.

$$LSE = \frac{1}{K}\sum_{k=0}^{K-1} e^2[k]$$

d) Determine the converge speed for the selected step size by plotting $e^2[n]$ versus n.

e) Change the adaptive filter length to observe its effects. Explain your observations.

f) Change the FIR filter length, N and explain its effects.

g) Change the step size and explain its effect on the convergence speed as well as misadjustment which is the final error after convergence.

PART 2

6.6. MyRIO Programming Tasks

In this part, the programing tasks in MATLAB are realized in real-time in MyRIO. Hence both system identification and noise cancellation are implemented. While system identification is realized on MyRIO CPU, noise cancellation uses the analog input and output ports of MyRIO and hence the realization involves both CPU and FPGA.

6.6.1. System Identification in MyRIO

Fig. 4 shows the project structure for the System Identificaiton Part. For this part, you will only use CPU programming.

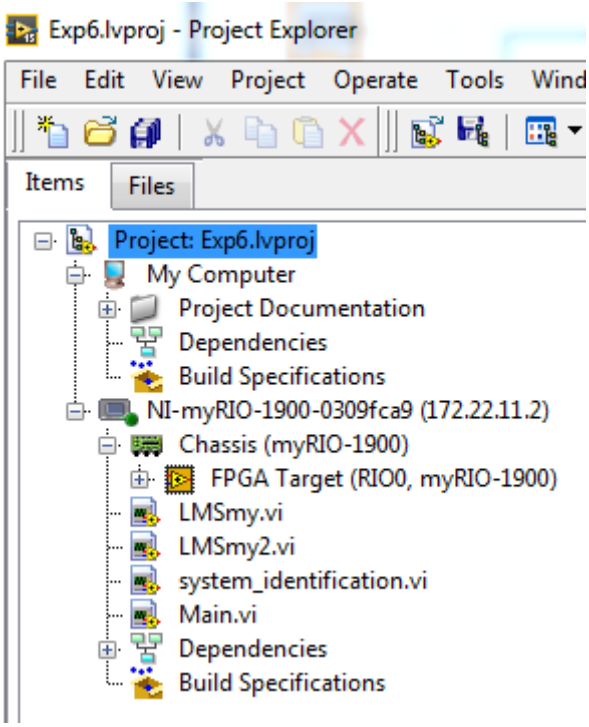

Fig. 4. Project Structure for the System Identification.

As it is seen in Fig. 4, there are three VI's under MyRIO. The two VI's are for the adaptive filter implemenetation using the LMS algorithm, namely **LMSmy.vi** and **LMSmy2.vi**, respectively. Actually, these two are exactly the same. They are named differently to avoid memory corruption when two adaptive filters are called simultaneously to observe the convergence speeds. The other VI is for the system identification.

a) Design an adaptive filter structure in LabVIEW, i.e. **LMSmy.vi**. The VI structure for the adaptive filter using the LMS algorithm is given in Fig. 5. Note that you should be able call this VI from another VI as a subVI and hence you should assign input and outputs for this VI appropriately.

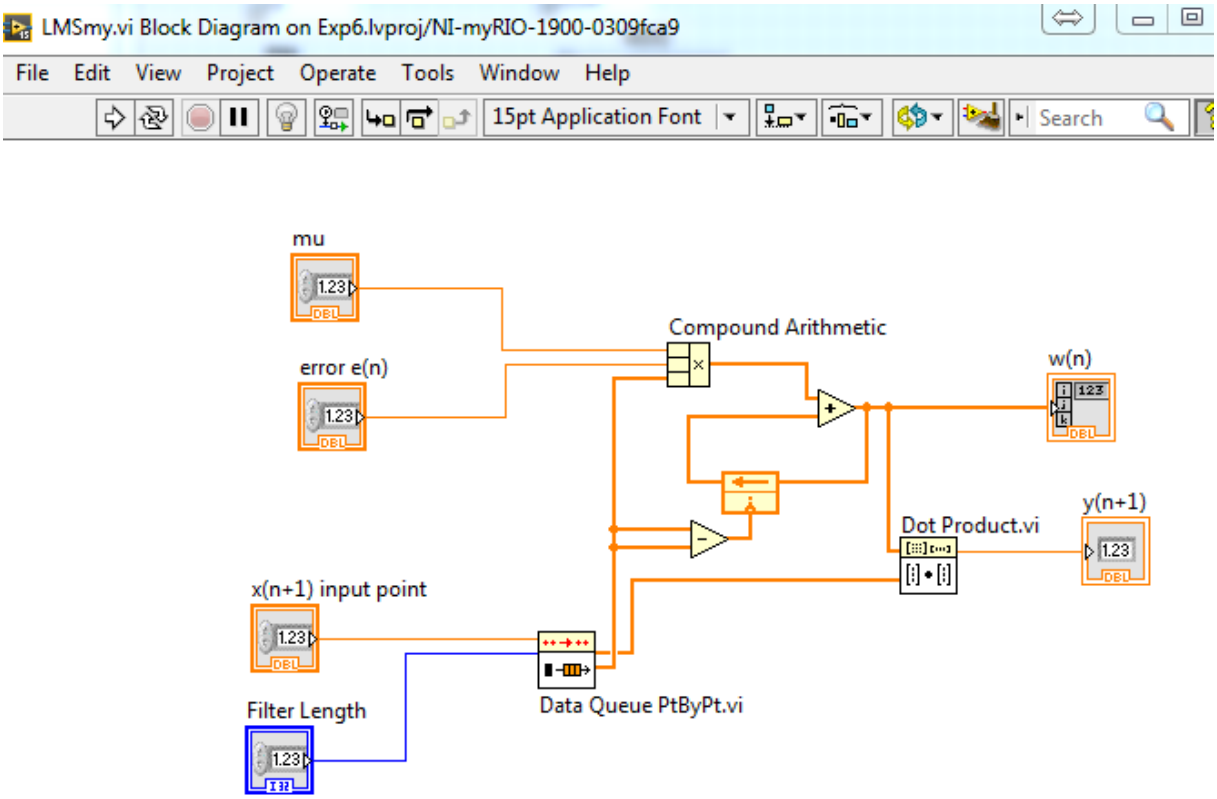

Fig. 5. Adaptive filter structure in LabVIEW.

b) Create another VI, **LMSmy2.vi** which is exactly the same as **LMSmy.vi.** Note that this VI also should be called from another VI as a subVI and hence you should assign input and outputs for this VI appropriately.

c) Create a new VI named as **system_identification.vi**. Generate a random input signal, filter it through a filter whose filter coefficients are adjusted from the front panel. Then build the adaptive filter structure as shown in Fig. 2 for obtaining the filter coefficients by the help of the adaptive filter. Note that filter output is corrupted by white noise and the result is the d[n] signal. The structure for this purpose is given in Fig. 6.

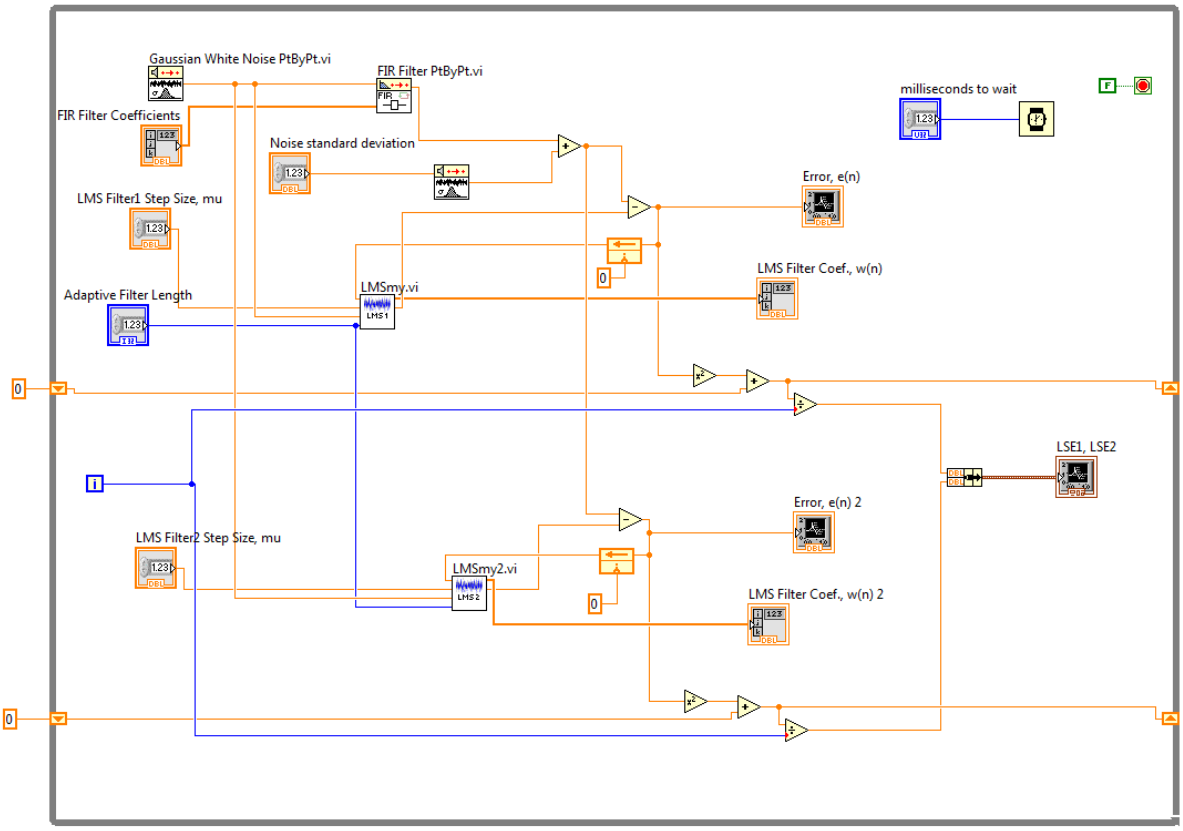

Fig. 6. The system identification structure in LabVIEW.

Note that the adaptive filter VI's, **LMSmy.vi** and **LMSmy2.vi** are seen with a symbol in Fig. 6. The same structure is repeated below the previous in order to observe the convergence speed and misadjustment for two adaptive filters which have different step sizes. The front panel for system identification is shown in Fig. 7.

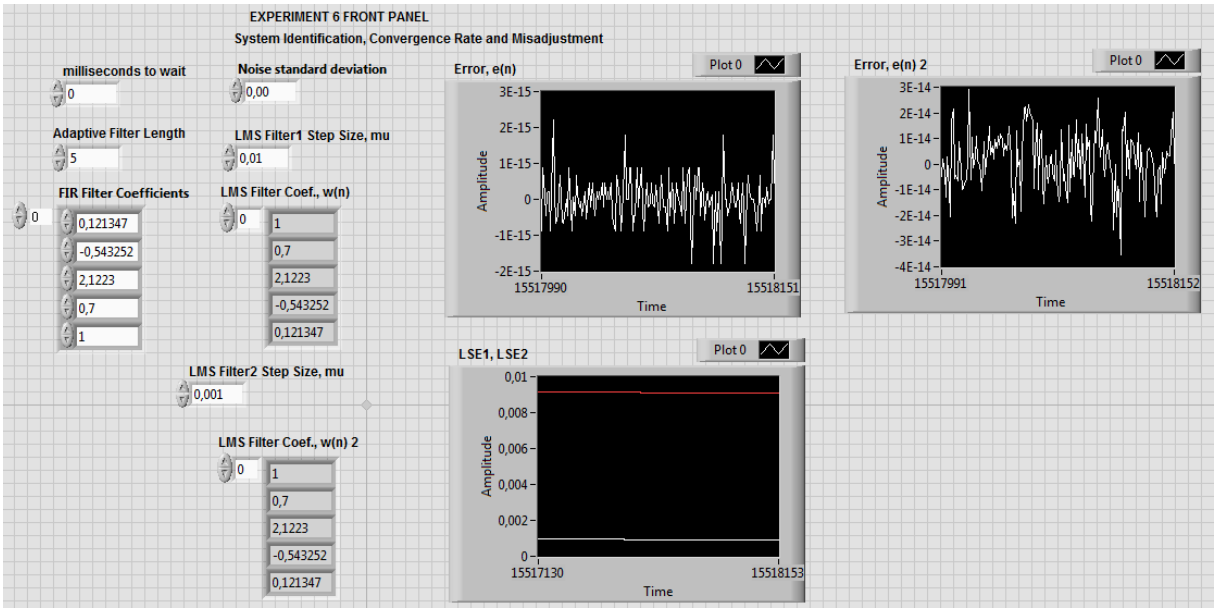

Fig.7. Front panel for system identification in Experiment 6.

d) Set the noise standard deviation to zero and run the VI. Check if the FIR filter coefficients are found accurately by the adaptive filter. Note that the coefficients should be the same only their order is different. Set the step sizes for adaptive filters to 0.01 and 0.001 in order to observe the difference between convergence speeds as well as the misadjustments. Explain which filter has faster convergence and which filter has the smallest misadjustment.

e) Set the noise standard deviation to 1 and **repeat d**. Explain your results and observations.

f) Increase the number of coefficients for the first adaptive filter. What is the effect of this? **Explain**.

g) Decrease the number of coefficients for the first adaptive filter. What is the effect of this? **Explain**.

6.6.2. Noise Cancellation in MyRIO

In this part, noise cancellation structure in Fig. 3 is implemented in MyRIO. The complete project structure for Experiment 6 is shown in Fig. 8.

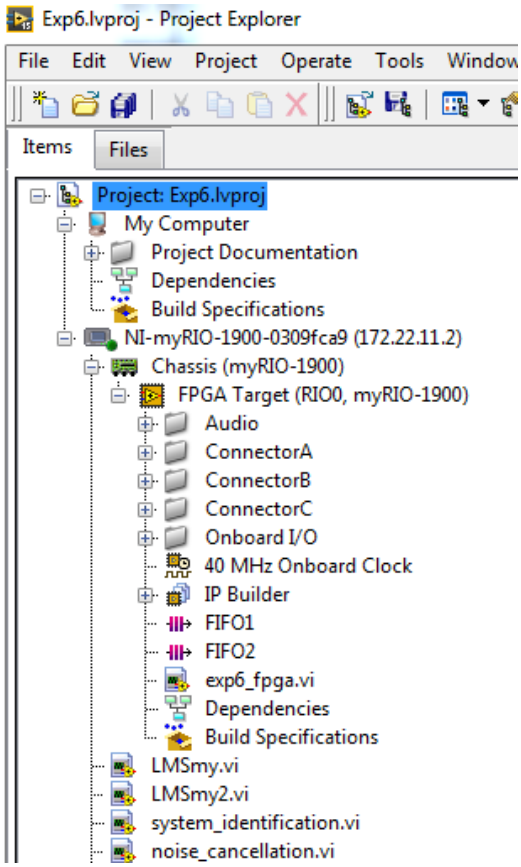

Fig. 8. Complete Project structure for Experiment 6.

The front panel for noise cancellation is given in Fig. 9. You need to have a computer headset as shown in Fig. 10 in order to give an input and listen to the sound from the audio ports.

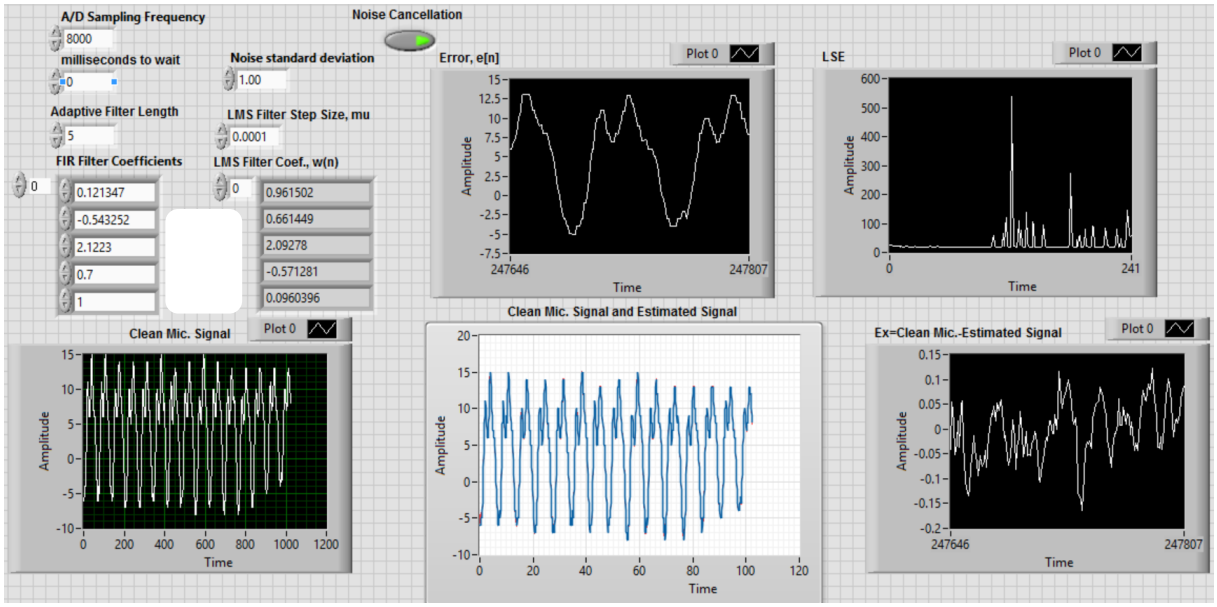

Fig. 9. Front panel for noise cancellation in Experiment 6.

Fig. 10. MyRIO audio input, output ports and a compatible headset.

a) Generate FIFO1 and FIFO2. **FIFO1** is **I16**, **Target to Host-DMA** type and **FIFO2** is **I16**, **Host to Target-DMA** type.

b) Prepare the FPGA VI for audio input-output as shown in Fig. 11.

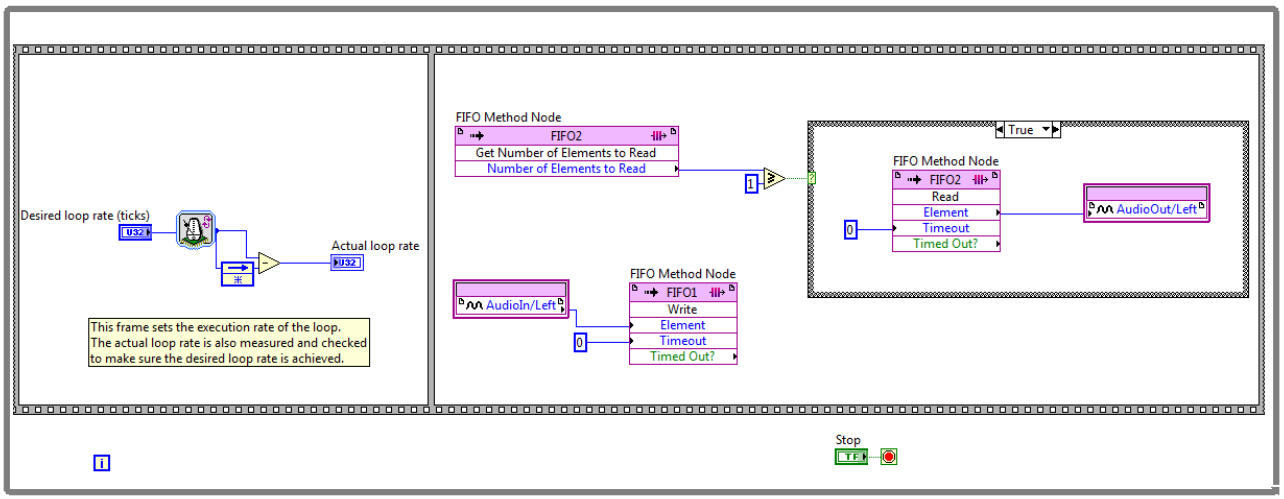

Fig. 11. FPGA VI for Experiment 6.

c) Configure the FIFO1 and FIFO2 in **noise_cancellation.vi** as shown in Fig. 12.

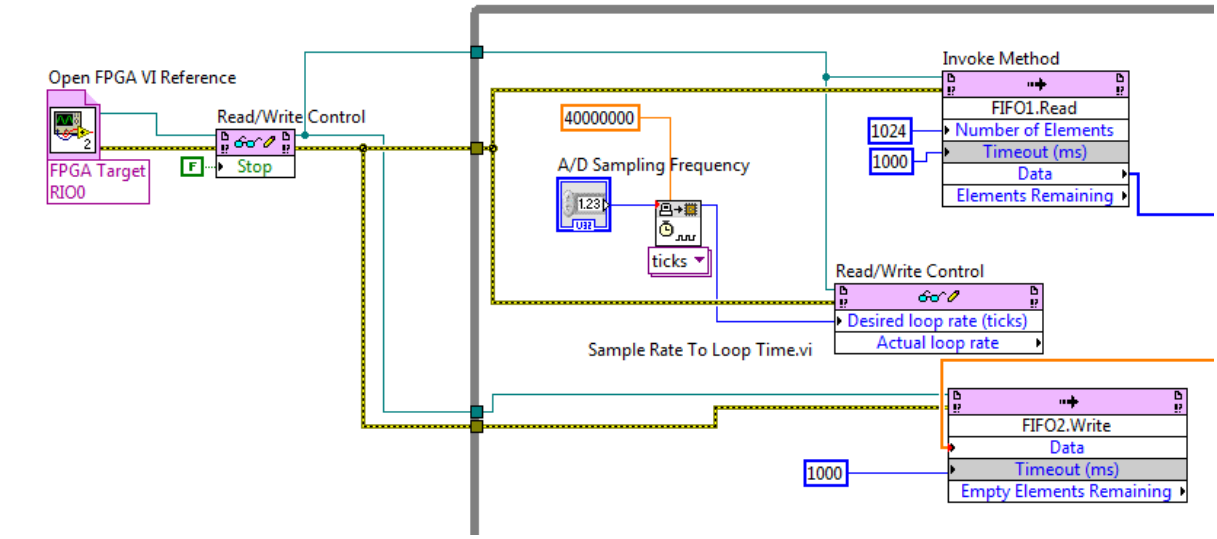

Fig. 12. The configuration of FIFO1 and FIFO2 for audio input-output.

d) The sample based adaptive filter structure may need to operate in blockwise manner since the audio samples are acquired in blocks of size 1024 samples. This configuration can be done as shown in Fig. 13.

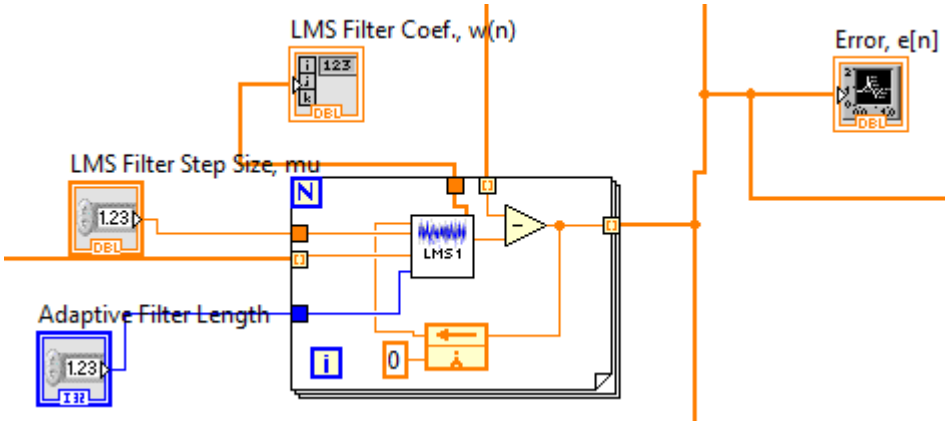

Fig. 13. The configuration of the sample-based adaptive filter structure to work in blockwise manner.

Complete the VI by adding necessary blocks as shown in Fig. 3. x[n] in Fig. 3 represents the sampled audio signal coming from FPGA by FIFO1. Add s[n] by using FIR filter whose coefficients are entered from the Front Panel. As it is seen in Fig. 9, Noise Cancellation button lights when it is pressed and adaptive filter error signal, e[n], is fed to the audio output. When the button is not pressed, noise corrupted microphone signal, d[n]=x[n]+s[n], is fed to the audio output. Note that e[n] is the estimated signal, $\hat{x}[n]$ after convergence and it is reasonable to have a large LSE when someone is speaking from the microphone. In this experiment, you should be able to hear the difference when there is noise cancellation.

e) Adjust the parameters of the program as shown in Fig. 9 and run your program. Check the convergence of the adaptive filter coefficients. While you speak to the microphone, press the Noise Cancellation button on and off to identify the function of the noise cancellation.

f) Change the adaptive filter length to see the effect on the quality of the noise cancellation.

g) Change the A/D sampling rate to **2000 Hz**. **Repeat e**. What are the differences between case **e and g**? **Explain the reasons**.

h) Set the A/D sampling rate to **8000 Hz** again. Increase the step size of the adaptive filter until divergence. Note the value of this step size. **Explain the reason** for divergence.