Microprocessors 2DP4

Final (Interface) Project

Instructor: Dr. Doyle

Moshiur Howlader – howlam - 1316948

Table of Contents

1) Introduction & Background

The ability to acquire medical data, whether it be the patient’s heart pulse, weight, blood pressure, and medical imaging, all require embedded systems [1,2]. These system uses a transducer to intake a physical phenomenon, and convert it to a way a computer can understand. This data then can be stored in a data for the doctor to diagnose and asses the patient. More and more, doctors rely these systems to help keep the patient’s health in check. See the figure below for an abstraction of a data acquiring system used to store patient’s data in a health care setting:

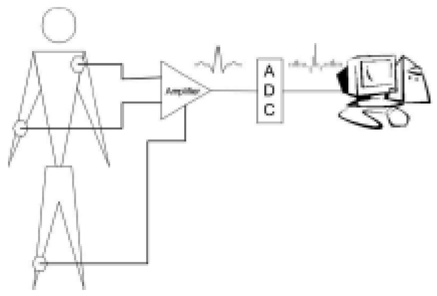


Figure 1: Data Acquisition System in a healthcare setting

The circuitry and the microprocessor system used can be built by the engineer in such a way that it can acquire any form of physical information! The fundamental steps to designing any (medical) data acquiring device can be broken down to figure 2. To illustrate this design process, we will be designing and testing a data acquisition system to keep track of the noise signal being inputted into the circuit. We can graphically plot this noise signal on the computer using ADC (Analog Data to Digital Data Conversion) and Serial Communication. The figure below (figure 2) illustrates the overall process taken in acquiring the noise data:

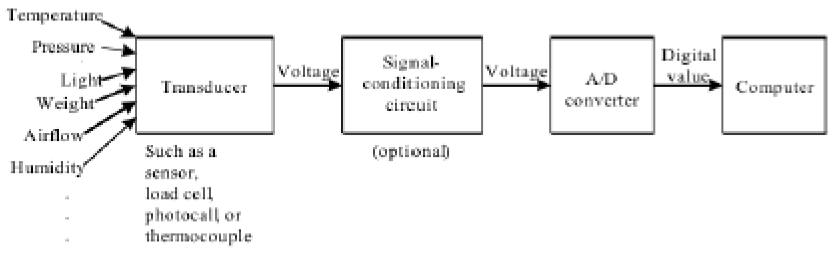


Figure 2: ADC Conversion Procedure (from Dr. Doyle Project Specification)

2) Design Methodology

This section will illustrate the steps to designing and building this data acquisition system. As shown in figure 2, the four main blocks are systematically built and tested to create a working system. Through each step, manual testing is done to ensure the working quality of the system. The figure below is the circuit used for conditioning the signal:

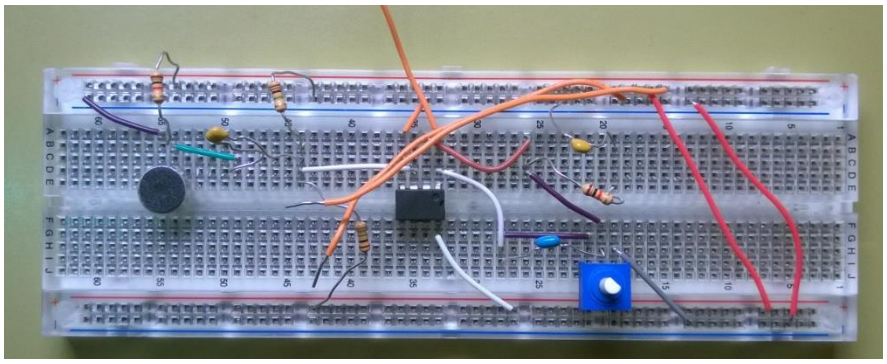


Figure 3: The conditioning circuit

This circuit has numerous components used to filter and condition the sound signal, such as the transducer (the electric microphone), the capacitors, op-amp, and level shifters. The following section will explain in detail how each of these components within the circuit is to condition the circuit.

a) Quantify Signal Properties

Here is the pSpice simulation of the circuit:

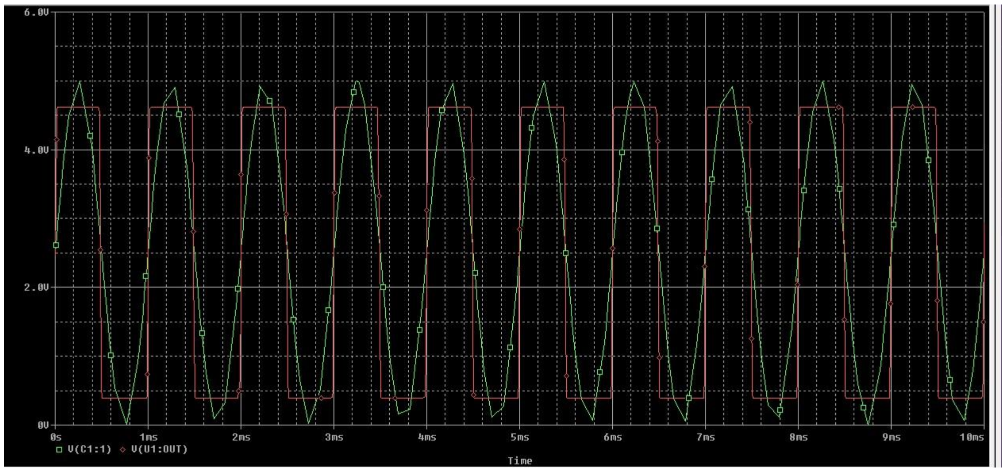


Figure 4: Simulation with numeric value

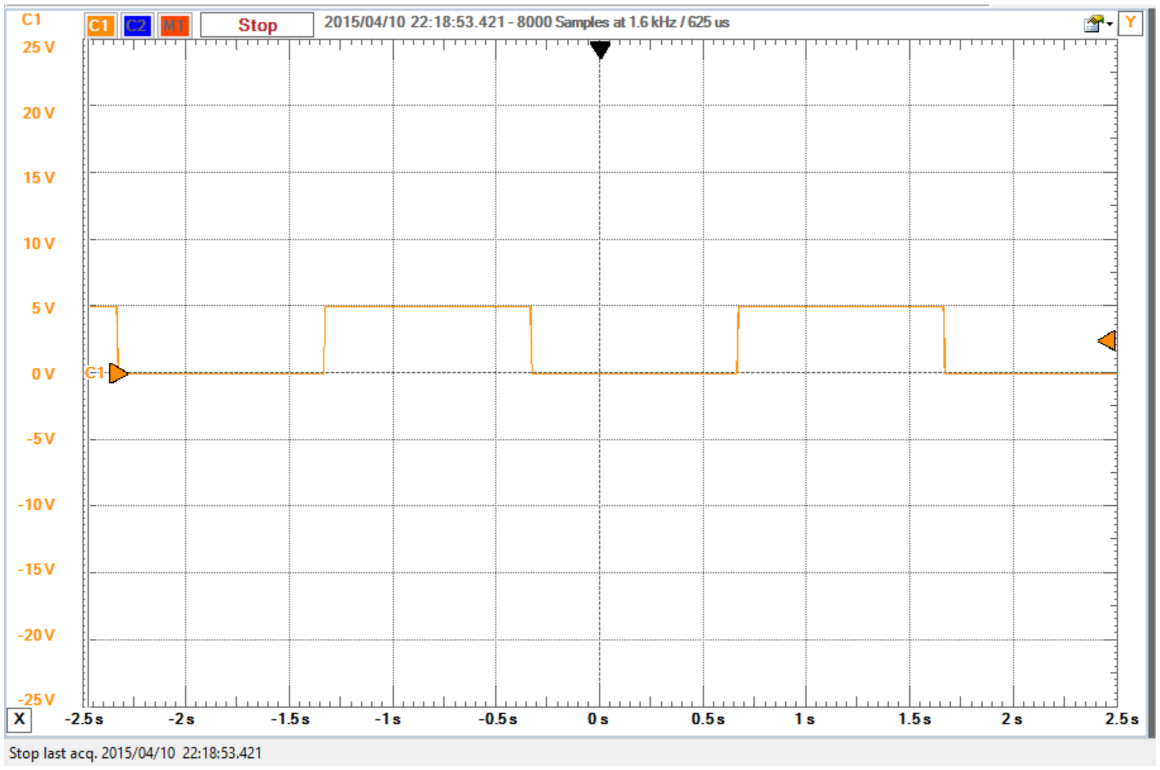


Figure 5: 1 Second Clock Ouput

b) Transducer

Using Matlab, the plots of the transducer is shown below:

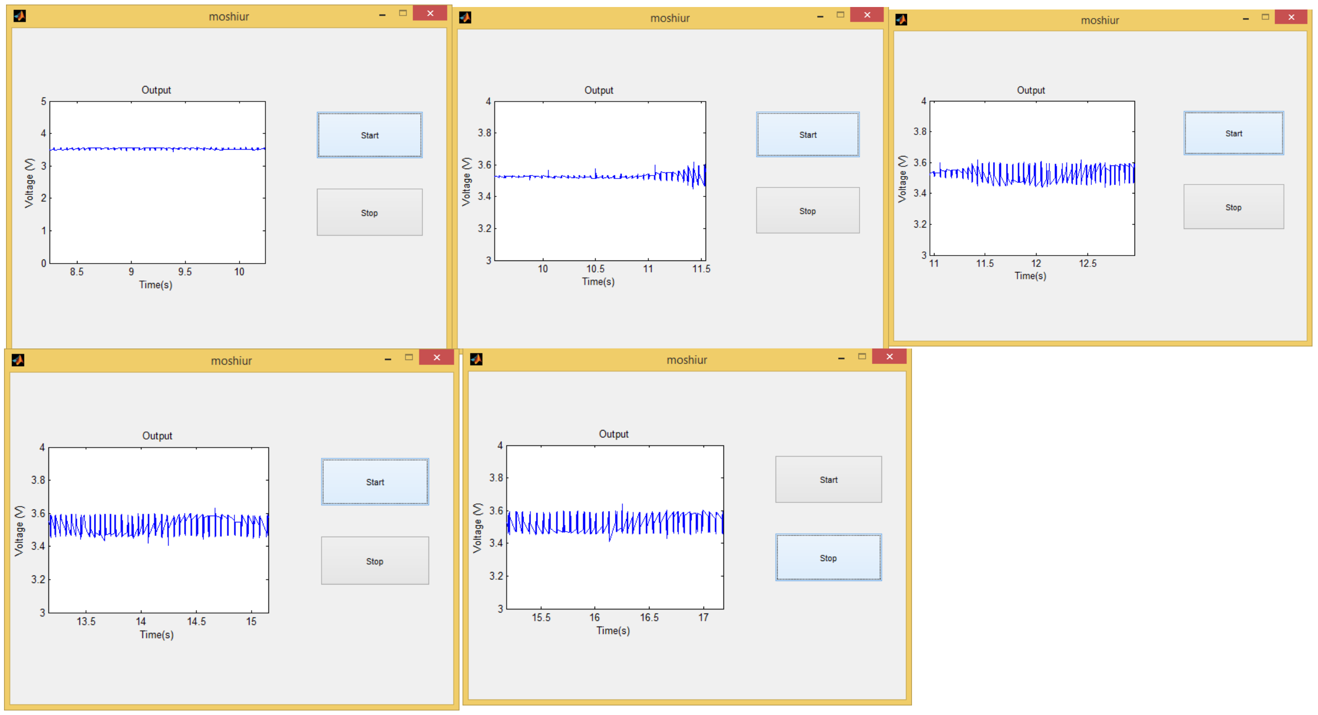
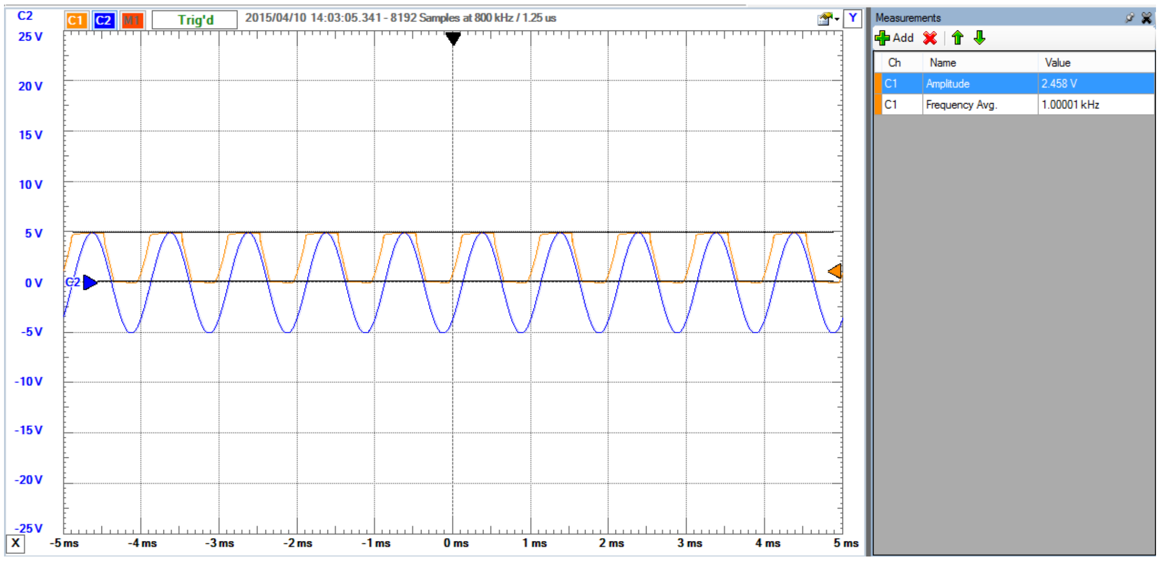


Figure 6: Matlab plots over a 17 second interval

c) Precondition/Amplification

This is an oscilloscope input/output plot of the circuit. The input is being filtered from 10 V amplitude into half. Also the voltage is being straightened at the tops/the peaks.

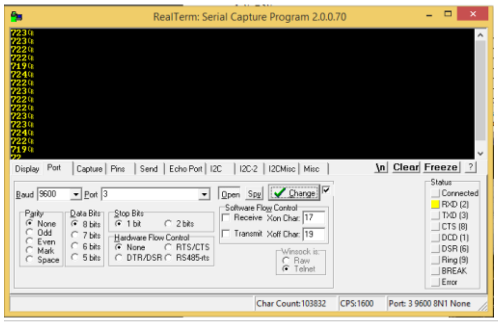


Ch	Name	Value
C1	Amplitude	2.458 V
C1	Frequency Avg.	1.00001 kHz

Figure 7: Testing the Precondition Capability of the Circuit via Oscilloscope

d) ADC

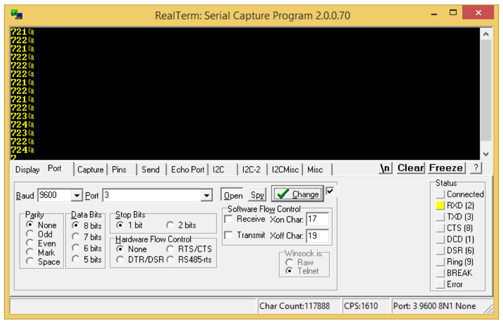
Real term output of the analog to digital output.


RealTerm: Serial Capture Program 2.0.0.70

```
723
723
722
722
719
724
722
723
722
722
723
723
724
722
719
72
```

Baud 9600, Port 3, Parity: None, Data Bits: 8 bits, Stop Bits: 1 bit, Hardware Flow Control: None

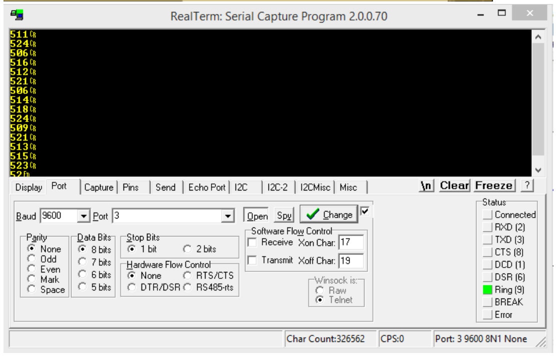
Char Count:103832 | CPS:1600 | Port: 3 9600 8N1 None


RealTerm: Serial Capture Program 2.0.0.70

```
721
722
721
722
722
722
721
722
721
722
723
724
723
722
724
7
```

Baud 9600, Port 3, Parity: None, Data Bits: 8 bits, Stop Bits: 1 bit, Hardware Flow Control: None

Char Count:117888 | CPS:1610 | Port: 3 9600 8N1 None

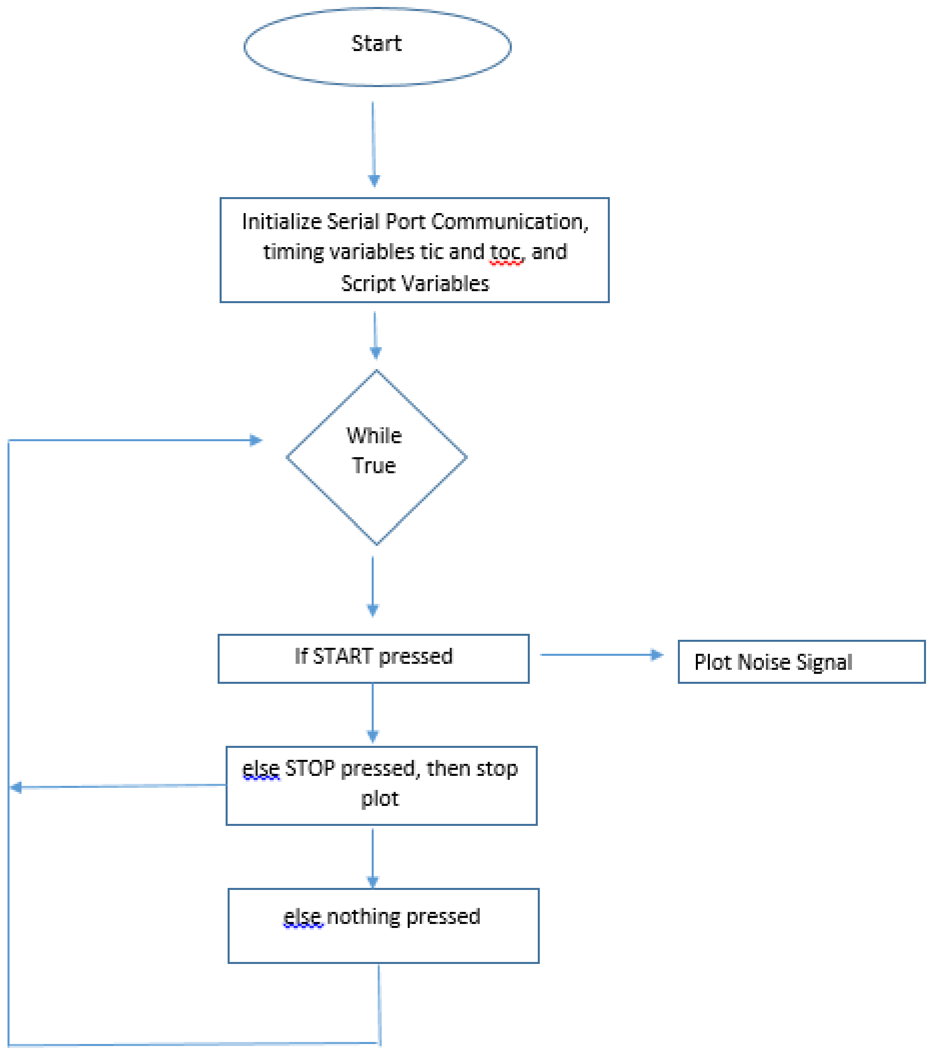

RealTerm: Serial Capture Program 2.0.0.70

```
511
524
506
516
512
521
506
514
518
524
509
521
513
515
523
52
```

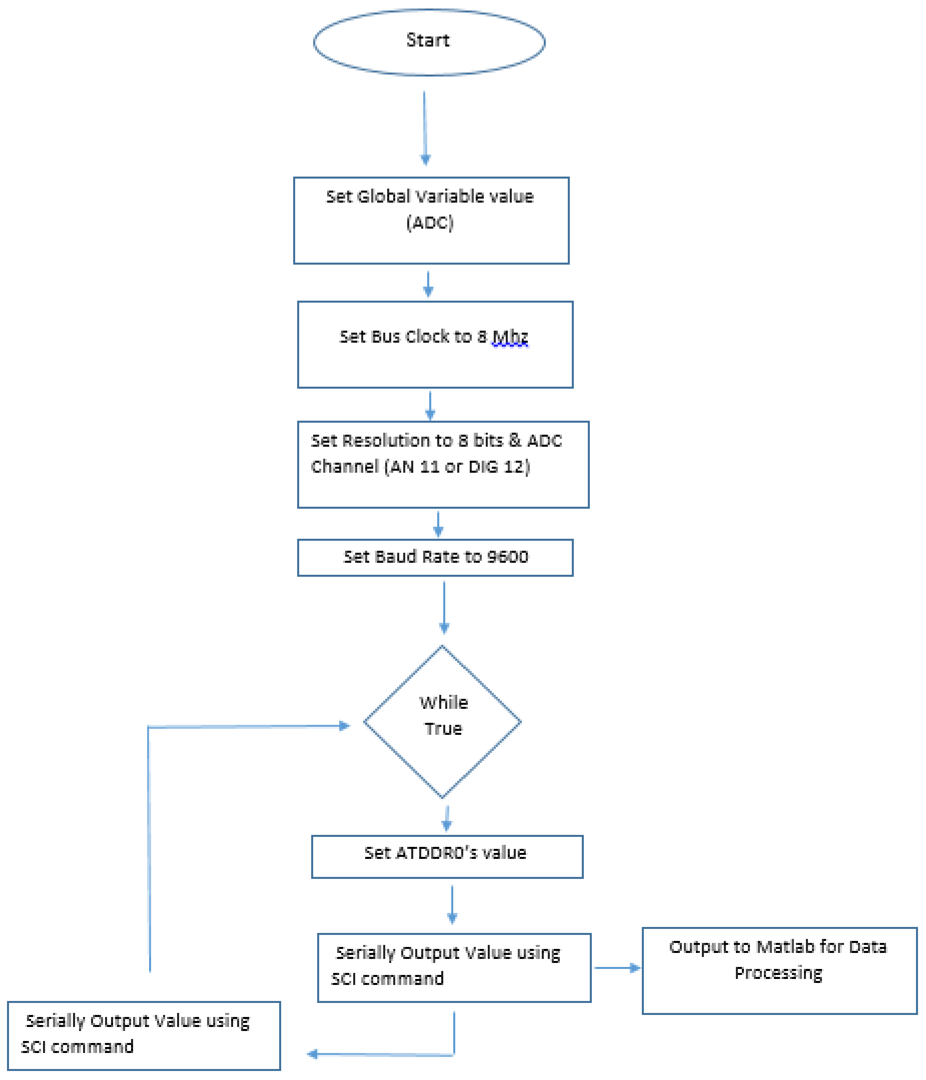
Baud 9600, Port 3, Parity: None, Data Bits: 8 bits, Stop Bits: 1 bit, Hardware Flow Control: None

Char Count:326562 | CPS:0 | Port: 3 9600 8N1 None

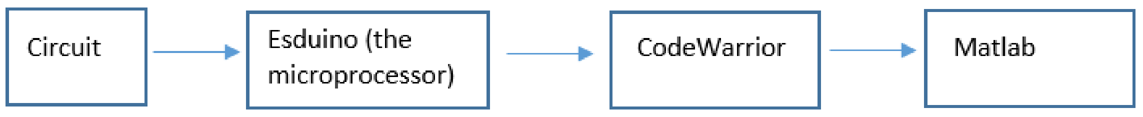
e) Data Processing (Algorithm Flowchart)


Start
Initialize Serial Port Communication, timing variables tic and toc, and Script Variables
While True
If START pressed
Plot Noise Signal
else STOP pressed, then stop plot
else nothing pressed

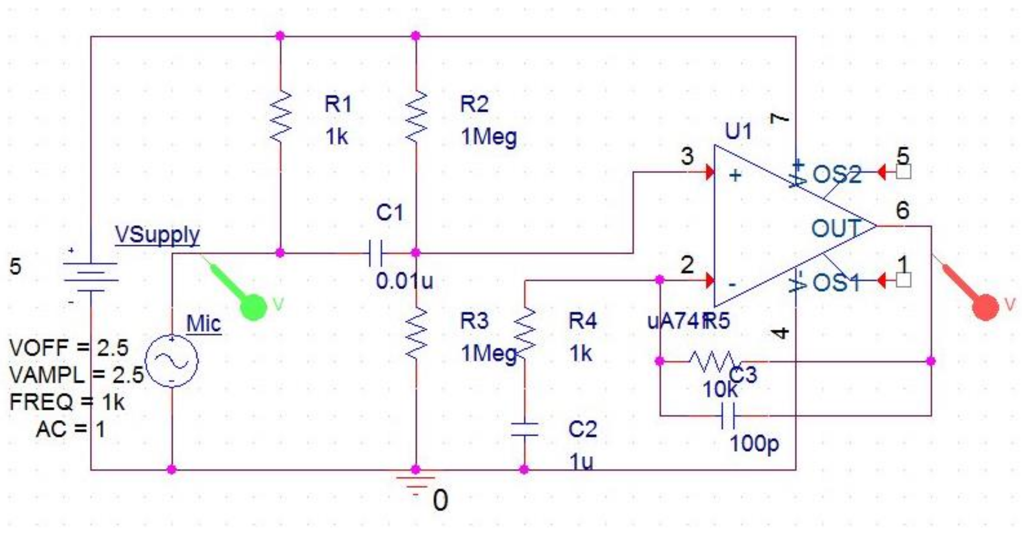
f) Control/Communicate (Algorithm Flowchart)


Start
Set Global Variable value (ADC)
Set Bus Clock to 8 Mhz
Set Resolution to 8 bits & ADC Channel (AN 11 or DIG 12)
Set Baud Rate to 9600
While True
Set ATDDR0's value
Serially Output Value using SCI command
Output to Matlab for Data Processing
Serially Output Value using SCI command

g) Full System Block Diagram



h) Full System Circuit Schematic
Below is the circuit schematic of the circuit used for this project?



3) Results

After testing the data, and the experimental results of each part of the Methodology, the conclusion can be made that the system function, not as well as it should. There is a wave output being plotted in the graph in real time. The circuitry of the system does work, just not as well as it should (the hardware). Other than the hardware, the software and the interface works fine.

4) Discussion

1. The maximum quantization error is equal to the ADC resolution (Δ). The calculation is shown below:

$$resolution = \Delta = V_{FS}/2^{m}, \text{ where m=10}$$
$$resolution = \Delta = 5/1024$$
$$resolution = \Delta = 0.004883$$

2. The maximum serial communication rate that can be implement is based on the assigned bus speed. In this case, the bus speed is 10 MHz. To calculate the maximum serial communication, refer to the formula below (taken from Dr. Doyle's Week A's lecture):

$$\frac{(Bus\ speed) \times (10^6)}{(16) \times (input\ baud\ rate)}$$
$$\frac{(10) \times (10^6)}{(16) \times (9600)}$$
$$65.1042\ Hz$$

3. The element which contributes to the speed limitation to the entire system would be the analog to digital conversion. Tests were done (using oscilloscope and a timer) to verify that this component is the limitation by timing. For example, the transducer and signal-

conditioning portion of the system was nearly instantaneous in its timing. The Matlab portion of the system (the computer plotting part) was also real-time/instantaneous. The A/D converter portion of the system took some time (few seconds) to convert and serially output to the computer.
4. According to Nyquist Rate, the maximum frequency of the analog signal you can effectively reproduce should be twice the highest frequency available in the sampling. If this condition is not satisfied, and the input signal exceeds this special frequency, aliasing will occur. When aliasing occurs, the critical features of the signal such as amplitude and the frequency will be lost or distorted. The poor plotting of Matlab can be attributed to the poor sampling rate (by not having the sampling frequency greatly exceed the Nyquist Rate, although import features were preserved).

5) Conclusion

The simple noise signal acquisition system worked well to plot the sound signal and to illustrate the embedded system design. The plots of the sound graphs were reasonable (reacting to sound, and vibration). This embedded system design can be extended to help make any form of medical data acquisition system. For example, we may take this exact same project, can simply use a different transducer applicable for the human body, such that we can record important biological information, such as patient's heartbeat. The same system (as well as the code) may as well be used to plot the heartbeat in real time! With more budget, more sophisticated systems can be built. There is a lot of future and economy in medical embedded systems. Really, embedded system is the future of medicine.

6) Appendix (note the code can also be found in the zip package)

CodeWarrior Script:

```
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */
#include "SCI.H"

//Moshiur Howlader
//1316948
//howlam

void delayby1ms(int x);



unsigned short val; // set as a global variable

void main(void) {

//setting Bus clock to 4MHz
/*PLLSEL=1;//choose PLL
OSCE=0;
VCOFRQ=00; //at this point fREF = 1MHz
```

```
SYNDIV=3 //VCLOCK=2*1MHz*(SYNDIV+1)= 8MHz
//At this point VCLOCK=8 MHz

LOCK = 1; //then use formula below for PLLCLK
POSTDIV = 0;//PLLCLK=VCLOCK/(POSTDIV+1)=8MHz/1=8MHz */


//Bus Clock = PLLCLK/2 = 4MHz

//CODE for Bus Clock

 CPMUCLKS = 0x80;  //PLLSEL = 1
 CPMUOSC  = 0x00;  //OSCE = 0
 CPMUSYNR = 0x1F;  //VCOFRQ = 0, SYNDIV = 31

 // Now fREF = 1 MHz and fVCO = 2* fREF * (31+1) = 64MHz
 CPMUFLG  = 0x00; //sets lock=0


 // PLLCLK = fVCO/4= 16MHz
 // Thus the Bus Clock = 8MHz = PLLCLK/2

 // Setup and enable ADC channel 3
ATDCTL1 = 0x2F; // set for 10-bit resolution, 8-bit --> 0x00
ATDCTL3 = 0x88; // right justified, one sample per sequence
ATDCTL4 = 0x02; // prescaler = 2; ATD clock = 6.25MHz / (2 * (2 + 1))
== 1.04MHz
ATDCTL5 = 0x2B; // continuous conversion on channel 0 because
equivalent for channel 11 0x2B, 0x20, 0x2C
                // ATDCTL5 is used to configured the pin for the
transducer circuit output
SCI_Init(9600); //sets baude rate


for(;;)
{

 val=ATDDR0; // read result from data result register 0
 SCI_OutUDec(val);   //copies value sends it to SCI
 SCI_OutChar(CR); //new cell



 delayby1ms(3.33);   //sample frequency: 300Hz. (1/300)*1000=3.333

}
}



void delayby1ms(int k)
{
 int ix;
 /* enable timer and fast timer flag clear */
 TSCR1 = 0x90;
 /* disable timer interrupt, set prescaler to 1*/
 TSCR2 = 0x00;
 /* enable OC0 */ //(not necessary)
 TIOS |= 0x01;
 TC0 = TCNT + 8000;
 for(ix = 0; ix < k; ix++) {
 while(!(TFLG1_C0F));
 TC0 += 8000;
 }
 /* disable OC0 */ //(not necessary)
 TIOS &= ~0x01;
}
```

Matlab Script:

```
function varargout = moshiur(varargin)
% MOSHIUR MATLAB code for moshiur.fig
%      MOSHIUR, by itself, creates a new MOSHIUR or raises the existing
%      singleton*.
%
%      H = MOSHIUR returns the handle to a new MOSHIUR or the handle to
%      the existing singleton*.
%
%      MOSHIUR('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in MOSHIUR.M with the given input
arguments.
%
%      MOSHIUR('Property','Value',...) creates a new MOSHIUR or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before moshiur_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to moshiur_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help moshiur

% Last Modified by GUIDE v2.5 08-Apr-2015 14:14:17

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @moshiur_OpeningFcn, ...
                   'gui_OutputFcn',  @moshiur_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
```

```
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before moshiur is made visible.
function moshiur_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to moshiur (see VARARGIN)

% Choose default command line output for moshiur
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes moshiur wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = moshiur_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in togglebutton1.
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of togglebutton1

delete(instrfindall);

    data = serial('COM3'); %COM6 port on laptop
    data.BaudRate = 9600; %BaudRate
    data.Terminator = 'CR'; %terminator
    fopen(data);
    plot1 = plot(nan);
    hold on
    test = 1;
    numberofPoints = 0;
    resol = 10;
    tic; %Start

    while test ==  1
        numberofPoints = numberofPoints + 1;
        yVal(numberofPoints) = (5/(2^resol))*(str2double(fgets(data)));
%step size, 5 is max voltage
        xVal(numberofPoints) = toc; %time value, time elapsed
        if mod(numberofPoints,10) == 0
            set(plot1,'YData',yVal,'XData',xVal);
            title('Output')
            ylabel('Voltage (V)')
            xlabel('Time(s)')
            axis([xVal(numberofPoints)-2 xVal(numberofPoints) 0 5]);
            pause(1e-16);
        end
    end
    hold off
    fclose(data);
    delete (data);
    clear('data');


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

delete(instrfindall);

    data = serial('COM3'); %COM6 port on laptop
    data.BaudRate = 9600; %BaudRate
    data.Terminator = 'CR'; %terminator
    fopen(data);
    plot1 = plot(nan);
    hold on
    test = 1;
    numberofPoints = 0;
    resol = 10;
    tic; %Start

    while test ==  1
        numberofPoints = numberofPoints + 1;
        yVal(numberofPoints) = (5/(2^resol))*(str2double(fgets(data)));
%step size, 5 is max voltage
        xVal(numberofPoints) = toc; %time value, time elapsed
        if mod(numberofPoints,10) == 0
            set(plot1,'YData',yVal,'XData',xVal);
            title('Output')
            ylabel('Voltage (V)')
            xlabel('Time(s)')
            axis([xVal(numberofPoints)-2 xVal(numberofPoints) 3 4]);
            pause(1e-16);
        end
    end
    hold off
    fclose(data);
    delete (data);
    clear('data');


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(instrfindall);
fclose(data);
delete (data);
clear('data');
```

7) Bibliography

1. http://www.thefreelibrary.com/The+microprocessor+and+medicine.-a0120144426
2. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=95282&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel2%2F727%2F3075%2F00095282.pdf%3Farnumber%3D95282