Instrumentation Oscilloscope

Ankit Aggarwal, Ankit Thareja, Shivank Singhal
Aditya Kumar
Netaji Subhas Institute of Technology
Azad Hind Fauj Marg, Sector 3, Dwarka, New Delhi, DL 110078, India
ankit.aggarwal@nsitonline.in

***Abstract*—** Oscilloscopes are the devices used to observe varying signal voltages. Originally, the oscilloscopes are considered as costly, bulky and standalone devices. The project aims at building a low cost, low power, handheld and portable oscilloscope with the basic utilities for efficient laboratorial use. In its current form, it can be powered either using a 1.5 V-1.8 V battery or the MSP430 MCU's inherent USB power port. Thus, it is designed to be used as a Boosterpack for the MSP430 Launchpad. This device consists primarily of three parts, viz. Signal Conditioning, Power Management and Microcontroller Unit with integrated LCD Display. The project has sampling as the basic technical basis, and the various circuits (Signal Conditioning) being used to ensure that the errors introduced in the system are minimal, power management circuits ensure a stable and efficient use of least possible supply voltage. The LCD Display is used to portray the signal and display the respective frequency and amplitude values. The primary aim of the project was successfully and satisfactorily achieved.

Keywords—Oscilloscope; Instrumentation; Low cost; Boosterpack; ADC; MSP430;

I. INTRODUCTION

An oscilloscope is a type of electronic test instrument that allows observation of constantly varying signal voltages, usually as a two- dimensional graph of one or more electrical potential differences using the vertical or y-axis, plotted as a function of time (horizontal or x-axis). This allows the measurement of peak-to-peak voltage of a waveform, the frequency of periodic signals. Oscilloscopes are used in the sciences, medicine, engineering, and telecommunications industry. General-purpose instruments are used for maintenance of electronic equipment and laboratory work. Originally, the oscilloscopes are considered as high cost, bulky and standalone devices.

The instrumentation oscilloscope has been developed for the users, primarily targeting the various college level laboratories and students. The signal usually encountered in experiments especially from instrumentation labs are primarily the low frequency signals and weak signals. The basic implementation of the idea is displaying the signal clearly on the LCD along with the peak to peak amplitude value and frequency of the signal. The device prototyped is a low cost, low power, handheld and portable device which will measure the commonly used laboratory sensors efficiently. It is a simple solution to facilitate easy availability and less expenditure on oscilloscopes.

This device can be powered either using an AA/Li-Ion 1.5 V-1.8 V battery or MSP430 MCU's inherent USB power port. Thus, it is designed as a boosterpack application for Launchpad. It consists primarily of three parts, viz. Signal Conditioning, Power Management and Microcontroller Unit with integrated LCD Display. Input is taken through scope probes and fed to signal conditioning circuitry, which is then passed on Microcontroller unit which further processes and calibrates the data to be displayed on LCD Display as shown in *Figure 1*. The primary aim of the project is reduction in the cost and power of the oscilloscopes used in the laboratories by replacing them with a low cost, low power and portable device. Thus with the theme of coming up with a low cost and portable substitute of the present day oscilloscope the project was started.

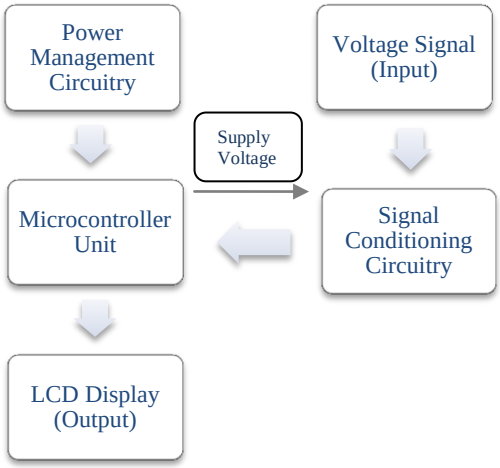


Figure 1- Top Level Block Diagram of the device

The originality of the project lies in its highlight point which is that the device is based on the basic implementation of circuits studied at the undergraduate level. It is simpler form of today's

DSO with a few reduced features which are not used at the primary level of study.
The next section further explores the proposed solution and its hardware as well as software implementation in detail. Finally, the last section explains the results and conclusions of this project along with required appendix and references.

II. PROPOSED SOLUTION

The device proposed and prototyped is a low cost, low power, handheld and portable device which will measure the commonly used laboratory sensors efficiently. It is a simple solution to facilitate easy availability and less expenditure on oscilloscopes. This device can be powered either using an AA/Li-Ion 1.5 V-1.8 V battery or MSP430 MCU's inherent USB power port. Thus, it is designed as a boosterpack application for Launchpad. It consists primarily of three parts, viz. Signal Conditioning, Power Management and Microcontroller Unit with integrated LCD Display.
The device contains a *boost converter* and a *power management circuit* to power the microcontroller efficiently with the least effective power possible. The microcontroller further provides the required supply voltage to other low-powered sections of the circuits (such as Instrumentation Amplifiers and Filters) using appropriate resistor dividers. Thus, this device can be powered either using an AA/Li-Ion 1.5 V-1.8 V battery or MSP430 MCU's inherent USB power port.

In the process to extract the desired signal, following the *Figure 2*, after its acquisition through standard 10x attenuator scope probe, it is first passed through a *clipper circuit* which clips any signals above 0.7 V or below -0.7 V; this circuit safely protects the other sections of the circuit from sudden high voltage changes. Next, it needs to be passed through a signal conditioning circuit, which has the basic function to protect the signal from being affected (modulated) in any respect due to the various external noises as well as due to the presence of circuit noise, which could originate from improper circuitry or ground. The signal conditioning circuit implemented to extract the voltage signal consists of two parts: A *low pass filter* in service with an *instrumentation amplifier*.

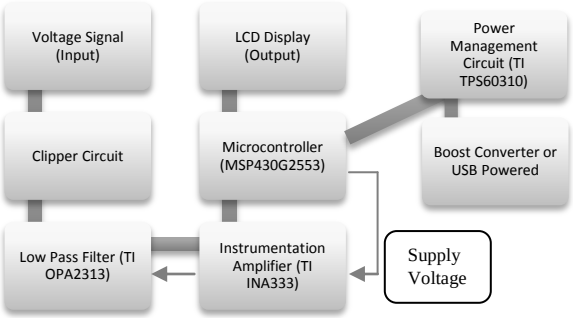


Figure 2 – System Level Block Diagram

The first part i.e. the low pass filter removes and filters the high frequency signal, generally being the noise, while the low frequency part of signal is passed through the circuit unattenuated. The low frequency signal being the physical signal which is converted to voltage signals by various transducers. A low power operational amplifier will be used for this purpose and a 2^{nd} order active low pass filter is realized through the same. The second part of the signal conditioning circuit is the instrumentation amplifier. It not only amplifies the already present signal but also reduces the noise coming on both the terminals of the amplifier. Due to its high CMRR, this circuit would significantly reduce the noise in the circuit. The gain deduced for the required circuit is about 125. It also serves an additional purpose of clamping the signal (by 10 mV) to make it suitable for analog to digital Conversion which requires reference voltage to be greater than 0 V for suitable conversion.
Next this signal is passed to the *microcontroller* in which the signal is fed for further processing. The signal is first sent to the A/D converter, which converts the signal to its equivalent digital values, which is calibrated in the software before being displayed real time using the software module.

The sampling speed of the A/D converter is set ten times the maximum frequency signal such that the signal can be satisfactorily reconstructed without any distortion (at least *twice* the maximum frequency of signal), as given by *Nyquist rate* [2].The sampling speed is fast enough to get an approximate signal up to the frequency of 1 KHz. It has been chosen so that the device gets enough number of samples per time period of the input signal to satisfactorily reconstruct the signal. The Microcontroller Unit is programmed using specific algorithms (see appendix B) to collect data and pass it onto the ADC for sampling, once sampled it sends data to the *LCD Display (Output)* which displays the desired signal with its amplitude and frequency measurements.

Specifications of the proposed device are:
Voltage Measurement: -100 mV to +100 mV
Frequency Measurement: 0-1 KHz

III IMPLEMENTATION

Hardware and *Software* used for implementation and prototyping of the device:

OPA2313 – Low Power Op-Amp
INA333 – Low Power Instrumentation amplifier
TPS60310 – Boost Converter
MSP430G2553 Launch pad - Microcontroller Unit
Nokia LCD 5110 – Low Power LCD Display
Scope Probes – Standard 10x attenuator connector type

Energia: Developing the code for processing in the launchpad.
Eagle: PCB and Schematic designing
TINA: Simulation of various circuits

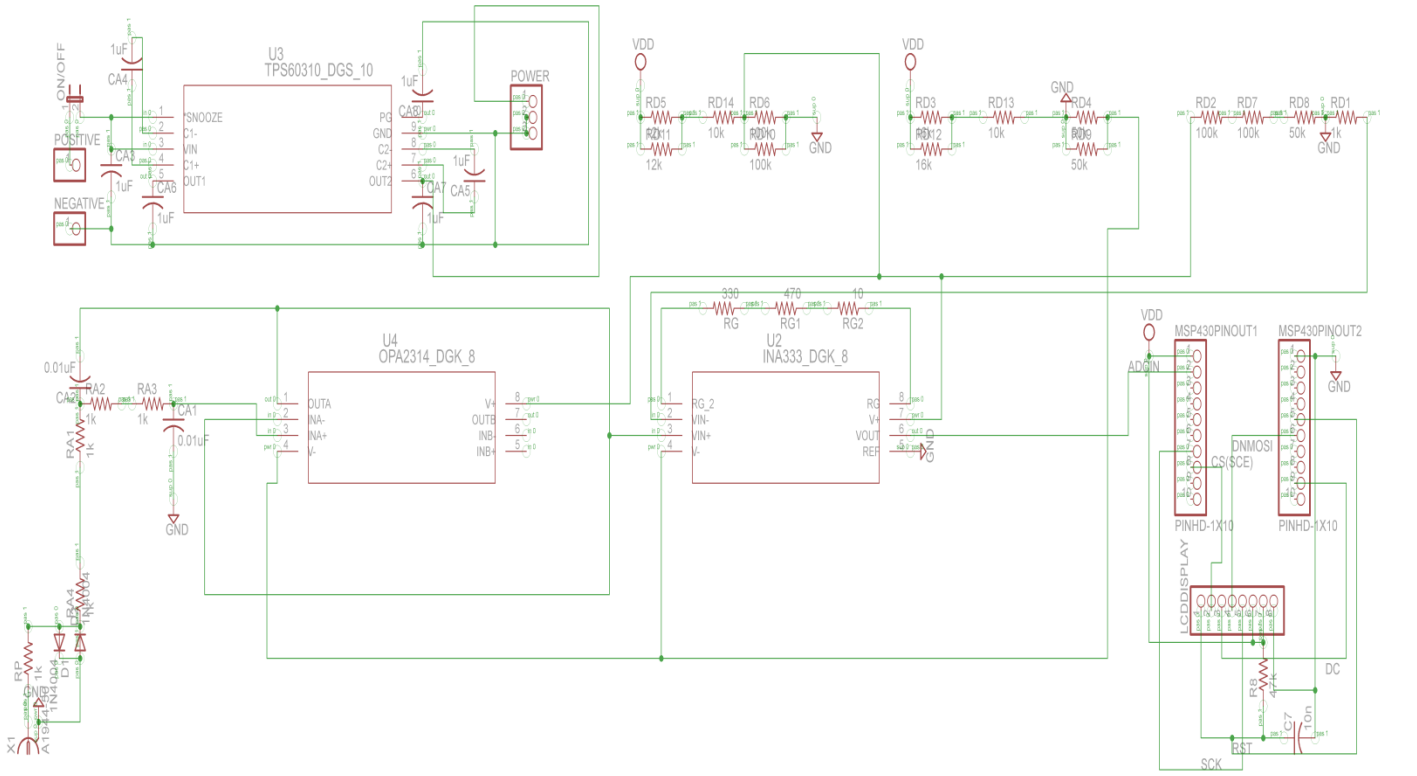


Figure 3 – Schematic for the Instrumentation Oscilloscope

A. Hardware Implementation

Figure 3 shows a schematic of the Instrumentation Oscilloscope (IOS). Hardware as explained earlier consists of three major parts: Signal conditioning circuit, Microcontroller unit with LCD Display and power management circuit. Initially, after its acquisition through standard 10x attenuator scope probe X1 (see *Figure 3*), it is first passed through a *clipper circuit* which clips any signals above 0.7 V or below -0.7 V; this circuit safely protects the other sections of the circuit from sudden high voltage changes. Now, if specifications are followed, the signal is within - 10 mV to +10 mV range, this is sent to the conditioning circuit. The Signal Conditioning circuit has a low-pass filter realized through an analog IC TI OPA-2313 (low power Op-Amp) and a low power Instrumentation Amplifier analog IC, TI INA333. Both ICs are low power ICs, so the power consumption is very minimal and moreover the circuits are easy to implement and integrate. 2nd order low pass filter was realized by attaching the appropriate values of R and C such that the required cutoff frequency of the filter (1 KHz) was achieved and phase lag error is minimized. TI INA333 was chosen due to its high CMRR to provide the noise immunity to the circuit and remove the external noise occurring in the environment. Its gain is set using the gain resistor (RG+RG1+RG2) = 810 Ohm such that the required gain is nearly equal to 125. It also shifts the level of the analog signal (by 10 mV) as analog to digital conversion recognizes only positive data. The signal is now shifted to the range 0 V-2.5 V, which is set as reference voltage for A/D conversion. TI MSP430G2553 was used as microprocessor IC in the MSP430 series Launchpad which is taken as the microcontroller unit for the device. The reason to choose such a MCU lies in its low cost, ultra-low power operations and excellent performance. It is used in processing the analog to digital data, thus, calculating the peak to peak amplitude value and the frequency of the analog signal. The acquired and conditioned signal is applied at pin P1.0 (2) of the launchpad. The MSP430 is connected with the very popular 84*48 pixels Nokia 5110 LCD Display to display the output. It is a very low power, monochrome LCD Display. Moreover, it's very cheap, easily available and requires a very low driving current. The display communicates with the MSP430 using SPI (Serial Peripheral Interface) Communication. The power management circuit primarily utilizes the boost converter analog IC, TI TPS60310, which provides a constant 3.3 V supply to the MSP430. There are numerous reasons behind choosing this IC such as it can implement a single-cell powered application and still maintain low-power performance and high battery life. It works on the principle of conservation and is 75% power efficient. The pump even if given a lower voltage gives higher stable voltage output decreasing the value of current output proportionately. This IC (TI TPS60310) is used to provide a constant 3.3V voltage input to MSP430 which further powers all the other ICs using appropriate resistor dividers. The device is powered either from a 1.5 V – 1.8 V single battery or using MSP430's inherent USB power. A switch ON/OFF (see *Figure 3*) provides a simple method to switch ON the device. Positive and Negative Jumpers are connected to the battery for supply. Common Ground is provided for the entire circuit using GND terminal of MSP430.

The appropriate pin-mapping is used with LCD Display for SPI communication.

Pin-Mapping for SPI Communication:
SCK (Serial Clock) - P1.5 (7)
CS (Chip Select) - P2.0 (8)
DNK (MOSI) - P1.7 (15)
D/C (Data Write) - P2.4 (12)
RST (Reset)-16

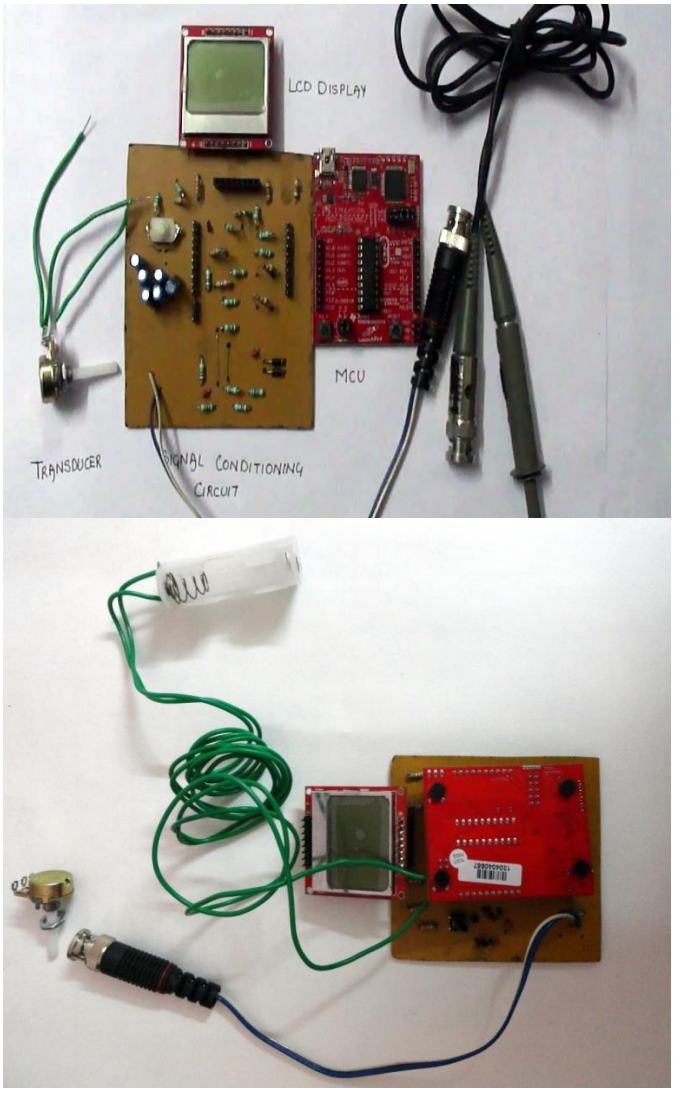


Figure 4 – Final Product Hardware

The final product can be seen in *Figure 4*, it is a home-made PCB, made using the toner-transfer method. The various components are soldered along with the SMD ICs. This PCB was designed in Eagle CAD (see Appendix A). There were several difficulties faced with home-made PCBs and soldering SMD ICs, but these were solved. The product can be easily mounted on the MSP430 Launchpad to work as a Boosterpack.

B. Software Implementation

The software for the device is implemented using the open-source Energia which gives an extremely simple interface to program MSP430 Launchpad.
The Launchpad communicates and displays the output on the LCD Display using the SPI communication.
Algorithm in a Nutshell
1. Sampling of data using MSP430G2553 ADC (10 Samples per Time Period) which uses ADC library specified for MSP430G2553.
2. Scaling and data interpolation for accurate and easily understandable output.
3. Data display using LCD Display which uses SPI Communication specified for MSP430G2553.
Pseudo Code:

```
define PIN_SCE   P2_0
define PIN_RESET 16
define PIN_DC    P2_4
define PIN_SDIN  P1_7
define PIN_SCLK  P1_5
define LCD_C     LOW
define LCD_D     HIGH
define LCD_X     84
define LCD_Y     48
define LCD_CMD   0

def array ASCII[][]
def function LcdCharacter(char)
def function LcdClear()
def function LcdInitialise()
def function LcdString(char *)
def function LcdWrite(byte , byte)
def function gotoXY(int , int )
def function drawLine()
def function markpixel(int ,int )
def function floatToString(char * , float , byte , byte )
def function adc()
def function interpolate(int ,int )

def function setup()
   CALL LcdInitialise()
   CALL LcdClear()
   CALL analogReference(INTERNAL2V5)

def function loop()
  initialize rem=0
  FOR rem<5
    set  sensorValue = adc()
    rem=rem+1
  ENDFOR
  set sensorValue=CALL adc()
  initialize cnt=0
  initialize tst=0;
  set  v1 = sensorValue * (2.5 / 1023.0)
  set  Max = v1
```

```
WHILE tst!=5
  set  sensorValue =adc()
  set voltage equals to the sensorValue * (2.5 / 1023.0)
  IF fabs(voltage-v1)>0.01999 THEN
    IF Max<voltage THEN
      set Max = voltage
    ENDIF
    Increment the cnt
  ELSE IF fabs(voltage-v1)<0.01999
    Increment the tst
  ELSE IF cnt==1000
    break
  ENDIF
ENDWHILE
IF cnt<3
  set freq = zero
ELSE Cnt<1000
  set freq = 10000.0/cnt
ENDIF
Calculate average number, avg_no
CALL floatToString (m,Max, 1, 0)
IF freq>999
  IF freq!=10000
    CALL floatToString(f,freq/1000, 2, 0)
    CALL gotoXY(0,5)
    CALL LcdString(m)
    CALL LcdString ("mv")
    CALL LcdString (",")
    CALL LcdString(f)
    CALL LcdString ("k")
  ELSE
    CALL gotoXY(0,5)
    CALL LcdString(m)
    CALL LcdString ("mv")
    CALL LcdString (",")
    CALL LcdString("N/A")
  ELSE
    CALL floatToString(f,freq/1000, 2, 0)
    CALL gotoXY(0,5)
    CALL LcdString(m)
    CALL LcdString ("mv")
    CALL LcdString (",")
    CALL LcdString(f)
    CALL LcdString ("Hz")
ENDIF
CALL drawLine()
FOR i <=83
  set voltage = 0
    FOR j<avg_no
      set sensorValue =adc()
      set voltage = voltage + sensorValue * (2.5 / 1023.0)
      Increment j by 1
    ENDFOR
  set voltage=voltage/avg_no
  set n=voltage/0.0625
  FOR clr<5
    CALL gotoXY(i,clr)
    IF clr!=2
      CALL LcdWrite (1,0x00)
    ELSE
      CALL LcdWrite (1,0x10)
    ENDIF
  ENDFOR
  According to n
    Calculate bank and n1
    CALL markpixel (n,bank)
  According to absolute difference (npr1-n1)
    According to value of bank and bankpr
      CALL interpolate()
      CALL LcdWrite(1,0xff)
  set npr1=n1
  set npr=n
  set bankpr=bank
 ENDFOR
```

In this code a two dimensional array, static const byte ASCII [] [5], is defined. This array contains a number of one dimensional arrays with each array consisting of 5 Hex-values, which represents a single character.

Every character is displayed in a 5x8 matrix. Since, we have set the display controller to draw the data horizontally, this means that every byte of data is saved row after row and if the controller reaches the 84th byte of data it jumps to the next column.

The data is written into the controller's memory by using the (SPI Communication), i.e., **LcdWrite()** method.

Here the 1st parameter decides whether the passed data is a command for the controller or just data to be written into the memory (like a single character). After the DC pin is set to HIGH or LOW and the chip is enabled to receive data, the actual data transfer starts. The data is shifted bit by bit into the memory of the controller starting with the most significant bit (MSB). That means that the first bit (from left) is put into the memory first. This pixel will be passed bit after bit into the last column of the 5×8 matrix.

The function **LcdCharacter()** does exactly the same for every byte (row of data) for a defined ASCII character of the already mentioned array. It helps to display single character on the LCD at a time and function LcdString() prints complete string on the LCD using the **LcdCharacter()** and **LcdWrite()** functions. Function LcdString_anim() is exactly the same as LcdString() except that it has delay for providing animation while displaying character on the LCD.

The function **LcdInitialise()** sets pinmode of various LCD pins and executes various command for setting temperature coefficient, Vop etc for LCD to function properly and **LcdClear()** is called to clear the contents of the LCD.

Function **gotoXY()** has arguments, column number and row number, which indicates where cursor has to be positioned. **Drawline()** function is declared to draw the reference line for the waveform to be displayed on the LCD.

adc() method returns the value provided by the A/D converter provided on the launchpad which will provide a 10 bit number ranging from 0 to 1023 and **markpixel()** function position the cursor to the location where pixel is to be displayed according to the value returned by the adc() function which draws the waveform of a signal. Here, while plotting waveform, linear interpolation is done which is implemented with the help of **interpolate ()** function whose 1st argument is the previous value of pixel displayed and 2nd argument is the value of pixel that is going to be displayed.
For displaying peak value and frequency on the LCD, peak value and frequency are converted to string using **floatToString()** method.

In the **setup ()** function, LcdInitialise() and LcdClear() are called and then a set of data is sent to controller of the LCD for display. Also, we have to set the reference voltage for the A/D conversion. This is set as 2.5V using analogReference() function.

In the loop (), we have written instructions that we want to repeat continuously.
Here, we have calculated frequency and peak value. Frequency is calculated by 1st storing the 1st value provided by the adc() function in a variable and then comparing it with the next values provided by the adc() function till the same stored value is returned. Meanwhile, we count the number of values provided by the adc() function while the comparison is made. Since analogRead () function provides 10k samples per second, thus, this algorithm (see Appendix B) calculates the value of the frequency. Simultaneously, the peak value of the signal is calculated.

In the 2nd step, we calculate the average number using our specific algorithms (see Appendix B) which is responsible for displaying the complete waveform on the LCD irrespective of the frequency. The minimum value of average number is 1 which displays maximum frequency, while the maximum value of average number is 100, which displays zero frequency.

In the 3rd step, first the reference line is drawn and the values of the amplitude and the frequency are displayed on the LCD and specific pixels are drawn using the algorithms discussed above. For More information refer to code in Appendix B.

IV. RESULTS

The Instrumentation Oscilloscope was tested under various conditions and performance analysis was conducted with the acquired data.
The simulation results for the device are shown in *Figure 5*. The result shows a small phase lag error due to the second order low pass filter. This error was minimized by choosing appropriate values of the time constant RC.

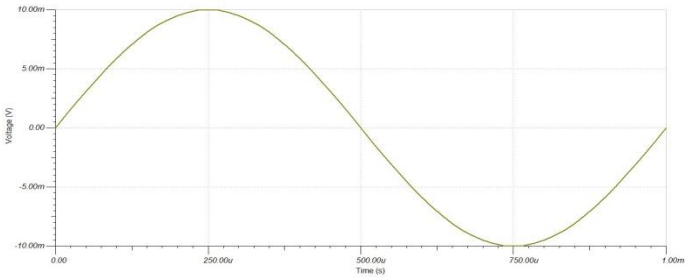


Figure 5(a) –10 mV sinusoidal input to be measured by the device.

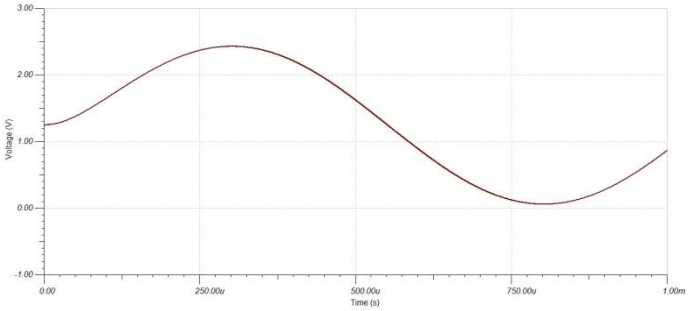


Figure 5(b) – Measured Output by the device (Signal after Instrumentation Amplifier).
The device was tested for basic transducers such as potentiometers, photoresistors, etc. It gave appropriate and satisfactory results, as shown in *Figure 6(a)* for constant DC, *Figure 6(b)* for noise and *Figure 6(c)* for square variation.

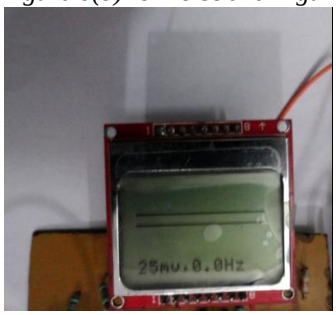


Figure 6 (a)

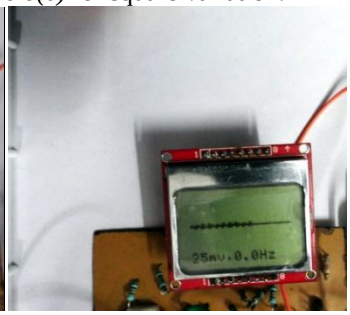


Figure 6 (b)

Figure 6 (c)

The final performance *analysis/ specifications* found after software and hardware testing are:

- Maximum Frequency detected without any distortion: 1 KHz.
- Maximum Amplitude of Signal measured without any distortion: 100 mV.
- Range: -100 mV to +100 mV.
- Waveforms observed successfully: DC; Triangular; Square; Sinusoidal;

V. CONCLUSIONS

Instrumentation Oscilloscope (IOS) is a low cost, low power, portable, efficient and simple device & it can have significant usage for educational purposes, especially in the laboratories.
It is an ideal solution for those looking for a quick and efficient analysis. This project aimed to find a low cost, portable substitute in place of conventional oscilloscopes. It was proposed to be low-power and efficient. The primary aim of the project was successfully and satisfactorily achieved.
Currently, it has been tested with constant, square, triangular and sinusoidal variations, but it can similarly work for any other voltage form/variation as well.
The Specifications which are found after the hardware and software testing along with performance analysis are given as:

Input Signal which can be measured: -100 mV to +100 mV.
Input signal frequency range which can be measured: 0 Hz to 1 KHz.

Few of the strengths of the project are:

1. It is a handheld and portable device.
2. It's very low cost device (see Appendix C).
3. It covers almost all sensors with low frequency output response.
4. It takes almost no power, it can be powered up using a 1.8 V battery and a single battery can provide a long life to the device.
5. It can be easily mounted on MSP430 Launchpad and can be used as a Boosterpack.
6. All the material being ultra - low power reduces the amount of current in the circuit hence, reducing heating effect of the devices.
7. The circuitry used is very simple and device is very easy to use.
8. It's easy to debug and can be easily further developed and programmable.

Future Scope of the Project:

1. The device currently doesn't store results. In future, it could be given an external memory or the results can be stored in the computing device used to power the launchpad using a GUI.
2. Sensors with high frequency response are currently not catered. This can be further improved in the next version.
3. Being a digital oscilloscope, the sampling frequency is what that determines the accuracy the shape of the waveform, maximum frequency is almost 1/10th the sampling frequency (10 Samples per Time Period). A better ADC with better resolution and sampling rate can be added for improved performance.
4. Currently, it cannot be used for the signal with more than 100 mV. This can be further improved in the next version.

ACKNOWLEDGMENTS

Authors would like to thank Vaibhav Gehlot for his guidance and help in PCB Printing.

REFERENCE

1. Nicholas "Nick" Gray .ABCs of ADCs - Rev 3, June 2006. Copyright 2003, 2004, 2006 National Semiconductor.
2. Yves Geerts, Michiel Steyaert, and Willy Sansen. *Design of multi-bit delta-sigma A/D converters*. 2002. Springer. ISBN 1-4020-7078-0.
3. Min Jae Lee .Signal Conditioning Circuit Design. Available from Internet. http://www.egr.msu.edu/classes/ece480/capstone/fall11/group05/docs/app%20notes/Application%20note-Minjae.pdf
4. BoosterPack Design Guide. From Texas Instruments Wiki. Available from Internet. http://processors.wiki.ti.com/index.php/BoosterPack_Design_Guide
5. Alberto Ricci Bitti. How to make PCBs at home in 1 hour & without special materials. Available from Internet. www.riccibitti.com/pcb/pcb.htm
6. Mixed Signal Micro Controller. MSP430G2x53 MSP430G2x13 Data Sheet. Copyright © 2011–2012, Texas Instruments Incorporated. Available from Internet. http://www.ti.com/lit/ds/symlink/msp430g2513.pdf
7. Datasheet, PCD8544 48* 84 pixels matrix LCD controller/driver. Philips Semiconductor. Available from internet. https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf
8. Walt Kester, James Bryant, Walt Jung, Scott Wurcer, Chuck Kitchin. SENSOR SIGNAL CONDITIONING. Available from Internet. http://www.analog.com/library/analogdialogue/archives/39-05/web_ch4_final.pdf
9. Matthew Seaman. Powering an MSP430 from a Single Battery Cell. Application Report SLAA398–September 2008. Available from Internet. http://www.ti.com/lit/an/slaa398/slaa398.pdf
10. Michael Stein. Design Considerations When Using MSP430 Graphics Library. Application Report SLAA548–October

2012. Available from Internet.
http://www.ti.com/lit/an/slaa548/slaa548.pdf
11. SPI experiments with the MSP430. Available on Internet. http://dbindner.freeshell.org/msp430/lcd_spi.html
12. Analog-to-digital converter. Available on Internet. en.wikipedia.org/wiki/Analog-to-digital_converter

APPENDIX A

The PCB Design is made using Eagle CAD and is shown in *Figure 7*.

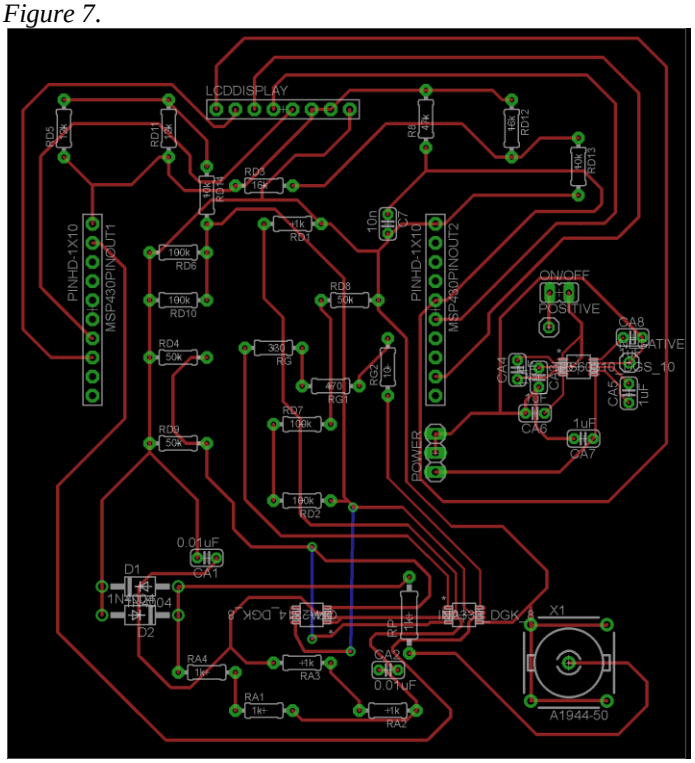
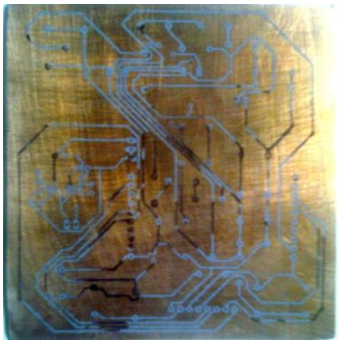
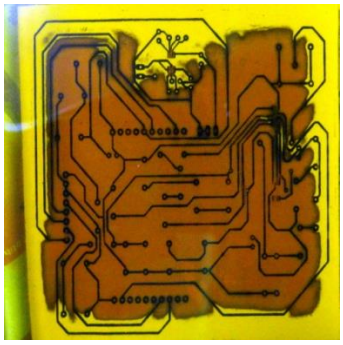


Figure 7 – PCB Design of IOS

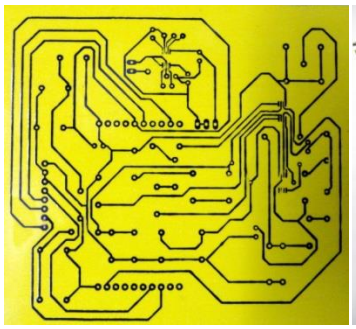
This PCB was developed using toner-transfer method at home and can be seen in *Figure 8(a), 8(b), 8(c), 8(d)*.



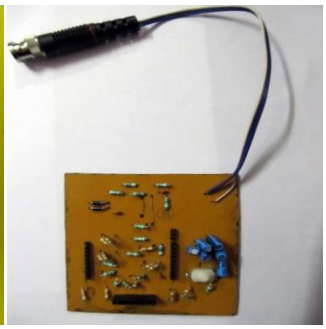
8(a)



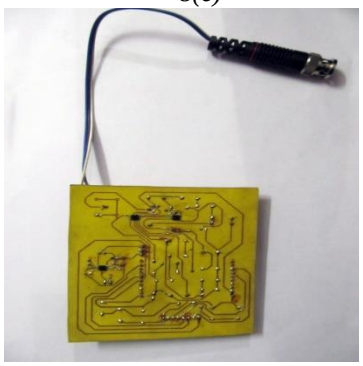
8(b)



8(c)



8(d)



8(e)

Figure 8(a) – PCB on Copper Clad after Pressing.
Figure 8(b) – PCB during Etching.
Figure 8(c) – PCB after being etched.
Figure 8(d) – Final Soldered PCB (Front).
Figure 8(e) – Final Soldered PCB (Back).

APPENDIX B

```
#include<String.h>
#include<Math.h>

#define PIN_SCE   P2_0  // LCD CS  .... Pin 3
#define PIN_RESET 16  // LCD RST .... Pin 1
#define PIN_DC    P2_4  // LCD Dat/Com. Pin 5
#define PIN_SDIN  P1_7  // LCD SPIDat . Pin 6
#define PIN_SCLK  P1_5  // LCD SPIClk . Pin 4
// LCD Gnd .... Pin 20
// LCD Vcc .... Pin 1
// LCD Vlcd ... Pin 7

#define LCD_C     LOW
#define LCD_D     HIGH

#define LCD_X     84
#define LCD_Y     48
#define LCD_CMD   0

int a = 0;
int value=0;

static const byte ASCII[][5] =
{
```

```
{0x00, 0x00, 0x00, 0x00, 0x00} // 20
,{0x00, 0x00, 0x5f, 0x00, 0x00} // 21 !
,{0x00, 0x07, 0x00, 0x07, 0x00} // 22 "
,{0x14, 0x7f, 0x14, 0x7f, 0x14} // 23 #
,{0x24, 0x2a, 0x7f, 0x2a, 0x12} // 24 $
,{0x23, 0x13, 0x08, 0x64, 0x62} // 25 %
,{0x36, 0x49, 0x55, 0x22, 0x50} // 26 &
,{0x00, 0x05, 0x03, 0x00, 0x00} // 27 '
,{0x00, 0x1c, 0x22, 0x41, 0x00} // 28 (
,{0x00, 0x41, 0x22, 0x1c, 0x00} // 29 )
,{0x14, 0x08, 0x3e, 0x08, 0x14} // 2a *
,{0x08, 0x08, 0x3e, 0x08, 0x08} // 2b +
,{0x00, 0x50, 0x30, 0x00, 0x00} // 2c ,
,{0x08, 0x08, 0x08, 0x08, 0x08} // 2d -
,{0x00, 0x60, 0x60, 0x00, 0x00} // 2e .
,{0x20, 0x10, 0x08, 0x04, 0x02} // 2f backslash
,{0x3e, 0x51, 0x49, 0x45, 0x3e} // 30 0
,{0x00, 0x42, 0x7f, 0x40, 0x00} // 31 1
,{0x42, 0x61, 0x51, 0x49, 0x46} // 32 2
,{0x21, 0x41, 0x45, 0x4b, 0x31} // 33 3
,{0x18, 0x14, 0x12, 0x7f, 0x10} // 34 4
,{0x27, 0x45, 0x45, 0x45, 0x39} // 35 5
,{0x3c, 0x4a, 0x49, 0x49, 0x30} // 36 6
,{0x01, 0x71, 0x09, 0x05, 0x03} // 37 7
,{0x36, 0x49, 0x49, 0x49, 0x36} // 38 8
,{0x06, 0x49, 0x49, 0x29, 0x1e} // 39 9
,{0x00, 0x36, 0x36, 0x00, 0x00} // 3a :
,{0x00, 0x56, 0x36, 0x00, 0x00} // 3b ;
,{0x08, 0x14, 0x22, 0x41, 0x00} // 3c <
,{0x14, 0x14, 0x14, 0x14, 0x14} // 3d =
,{0x00, 0x41, 0x22, 0x14, 0x08} // 3e >
,{0x02, 0x01, 0x51, 0x09, 0x06} // 3f ?
,{0x32, 0x49, 0x79, 0x41, 0x3e} // 40 @
,{0x7e, 0x11, 0x11, 0x11, 0x7e} // 41 A
,{0x7f, 0x49, 0x49, 0x49, 0x36} // 42 B
,{0x3e, 0x41, 0x41, 0x41, 0x22} // 43 C
,{0x7f, 0x41, 0x41, 0x22, 0x1c} // 44 D
,{0x7f, 0x49, 0x49, 0x49, 0x41} // 45 E
,{0x7f, 0x09, 0x09, 0x09, 0x01} // 46 F
,{0x3e, 0x41, 0x49, 0x49, 0x7a} // 47 G
,{0x7f, 0x08, 0x08, 0x08, 0x7f} // 48 H
,{0x00, 0x41, 0x7f, 0x41, 0x00} // 49 I
,{0x20, 0x40, 0x41, 0x3f, 0x01} // 4a J
,{0x7f, 0x08, 0x14, 0x22, 0x41} // 4b K
,{0x7f, 0x40, 0x40, 0x40, 0x40} // 4c L
,{0x7f, 0x02, 0x0c, 0x02, 0x7f} // 4d M
,{0x7f, 0x04, 0x08, 0x10, 0x7f} // 4e N
,{0x3e, 0x41, 0x41, 0x41, 0x3e} // 4f O
,{0x7f, 0x09, 0x09, 0x09, 0x06} // 50 P
,{0x3e, 0x41, 0x51, 0x21, 0x5e} // 51 Q
,{0x7f, 0x09, 0x19, 0x29, 0x46} // 52 R
,{0x46, 0x49, 0x49, 0x49, 0x31} // 53 S
,{0x01, 0x01, 0x7f, 0x01, 0x01} // 54 T
,{0x3f, 0x40, 0x40, 0x40, 0x3f} // 55 U
,{0x1f, 0x20, 0x40, 0x20, 0x1f} // 56 V
,{0x3f, 0x40, 0x38, 0x40, 0x3f} // 57 W
,{0x63, 0x14, 0x08, 0x14, 0x63} // 58 X
,{0x07, 0x08, 0x70, 0x08, 0x07} // 59 Y
,{0x61, 0x51, 0x49, 0x45, 0x43} // 5a Z
,{0x00, 0x7f, 0x41, 0x41, 0x00} // 5b [
,{0x02, 0x04, 0x08, 0x10, 0x20} // 5c ¥
,{0x00, 0x41, 0x41, 0x7f, 0x00} // 5d ]
,{0x04, 0x02, 0x01, 0x02, 0x04} // 5e ^
,{0x40, 0x40, 0x40, 0x40, 0x40} // 5f _
,{0x00, 0x01, 0x02, 0x04, 0x00} // 60 `
,{0x20, 0x54, 0x54, 0x54, 0x78} // 61 a
,{0x7f, 0x48, 0x44, 0x44, 0x38} // 62 b
,{0x38, 0x44, 0x44, 0x44, 0x20} // 63 c
,{0x38, 0x44, 0x44, 0x48, 0x7f} // 64 d
,{0x38, 0x54, 0x54, 0x54, 0x18} // 65 e
,{0x08, 0x7e, 0x09, 0x01, 0x02} // 66 f
,{0x0c, 0x52, 0x52, 0x52, 0x3e} // 67 g
,{0x7f, 0x08, 0x04, 0x04, 0x78} // 68 h
,{0x00, 0x44, 0x7d, 0x40, 0x00} // 69 i
,{0x20, 0x40, 0x44, 0x3d, 0x00} // 6a j
,{0x7f, 0x10, 0x28, 0x44, 0x00} // 6b k
,{0x00, 0x41, 0x7f, 0x40, 0x00} // 6c l
,{0x7c, 0x04, 0x18, 0x04, 0x78} // 6d m
,{0x7c, 0x08, 0x04, 0x04, 0x78} // 6e n
,{0x38, 0x44, 0x44, 0x44, 0x38} // 6f o
,{0x7c, 0x14, 0x14, 0x14, 0x08} // 70 p
,{0x08, 0x14, 0x14, 0x18, 0x7c} // 71 q
,{0x7c, 0x08, 0x04, 0x04, 0x08} // 72 r
,{0x48, 0x54, 0x54, 0x54, 0x20} // 73 s
,{0x04, 0x3f, 0x44, 0x40, 0x20} // 74 t
,{0x3c, 0x40, 0x40, 0x20, 0x7c} // 75 u
,{0x1c, 0x20, 0x40, 0x20, 0x1c} // 76 v
,{0x3c, 0x40, 0x30, 0x40, 0x3c} // 77 w
,{0x44, 0x28, 0x10, 0x28, 0x44} // 78 x
,{0x0c, 0x50, 0x50, 0x50, 0x3c} // 79 y
,{0x44, 0x64, 0x54, 0x4c, 0x44} // 7a z
,{0x00, 0x08, 0x36, 0x41, 0x00} // 7b {
,{0x00, 0x00, 0x7f, 0x00, 0x00} // 7c |
,{0x00, 0x41, 0x36, 0x08, 0x00} // 7d }
,{0x10, 0x08, 0x08, 0x10, 0x08} // 7e ←
,{0x00, 0x06, 0x09, 0x09, 0x06} // 7f →
};



void LcdCharacter(char character)
{
  LcdWrite(LCD_D, 0x00);
  for (int index = 0; index < 5; index++)
  {
    LcdWrite(LCD_D, ASCII[character - 0x20][index]);
  }
  LcdWrite(LCD_D, 0x00);
}

void LcdClear(void)
```

```
{
  for (int index = 0; index < LCD_X * LCD_Y / 8; index++)
  {
    LcdWrite(LCD_D, 0x00);
  }
}

void LcdInitialise(void)
{
  pinMode(PIN_SCE,   OUTPUT);
  pinMode(PIN_RESET, OUTPUT);
  pinMode(PIN_DC,    OUTPUT);
  pinMode(PIN_SDIN,  OUTPUT);
  pinMode(PIN_SCLK,  OUTPUT);

  digitalWrite(PIN_RESET, LOW);
  digitalWrite(PIN_RESET, HIGH);

  LcdWrite( LCD_CMD, 0x21 );  // LCD Extended Commands.
  LcdWrite( LCD_CMD, 0xBf );  // Set LCD Vop (Contrast). //B1
  LcdWrite( LCD_CMD, 0x04 );  // Set Temp coefficent. //0x04
  LcdWrite( LCD_CMD, 0x14 );  // LCD bias mode 1:48. //0x13
  LcdWrite( LCD_CMD, 0x0C );  // LCD in normal mode. 0x0d for inverse
  LcdWrite(LCD_C, 0x20);
  LcdWrite(LCD_C, 0x0C);
}

void LcdString(char *characters)
{
  while (*characters)
  {
    LcdCharacter(*characters++);
  }
}
void LcdString_anim(char *characters)
{
  while (*characters)
  {
    LcdCharacter(*characters++);
    delay(500);
  }
}

void LcdWrite(byte dc, byte data)
{
  digitalWrite(PIN_DC, dc);
  digitalWrite(PIN_SCE, LOW);
  shiftOut(PIN_SDIN, PIN_SCLK, MSBFIRST, data);
  digitalWrite(PIN_SCE, HIGH);
}

// gotoXY routine to position cursor
// x - range: 0 to 84
// y - range: 0 to 5

void gotoXY(int x, int y)
{
  LcdWrite( 0, 0x80 | x);  // Column.
  LcdWrite( 0, 0x40 | y);  // Row(banks 0 to 5).

}
void drawLine(void)
{
  unsigned char  j;
  for(j=0; j<84; j++) // Refernce Line
  {
    gotoXY (j,2);
    LcdWrite (1,0x10);
  }
}
void markpixel(int pixel,int er)
{
  if(pixel==0 && er!=2)
  {
    LcdWrite (1,0x80);
  }
  else if(pixel==1 && er!=2)
  {
    LcdWrite (1,0x40);
  }
  else if(pixel==2 && er!=2)
  {
    LcdWrite (1,0x20);
  }
  else if(pixel==3 && er!=2)
  {
    LcdWrite (1,0x10);
  }
  else if(pixel==4 && er!=2)
  {
    LcdWrite (1,0x08);
  }
  else if(pixel==5 && er!=2)
  {
    LcdWrite (1,0x04);
  }
  else if(pixel==6 && er!=2)
  {
    LcdWrite (1,0x02);
  }
  else if(pixel==7 && er!=2)
  {
    LcdWrite (1,0x01);
  }
  if(pixel==0 && er==2)
  {
    LcdWrite (1,0x90);
```

```
    }
    else if(pixel==1 && er==2)
    {
      LcdWrite (1,0x50);
    }
    else if(pixel==2 && er==2)
    {
      LcdWrite (1,0x30);
    }
    else if(pixel==3 && er==2)
    {
      LcdWrite (1,0x10);
    }
    else if(pixel==4 && er==2)
    {
      LcdWrite (1,0x18);
    }
    else if(pixel==5 && er==2)
    {
      LcdWrite (1,0x14 && er==2);
    }
    else if(pixel==6 && er==2)
    {
      LcdWrite (1,0x12);
    }
    else if(pixel==7 && er==2)
    {
      LcdWrite (1,0x11);
    }
}
// Function For converting float to string.
char * floatToString(char * outstr, float val, byte precision, byte
widthp)
{
    char temp[16]; //increase this if you need more digits than 15
    byte i;

    temp[0]='\0';
    outstr[0]='\0';

    if(val < 0.0)
    {
      strcpy(outstr,"-\0");  //print "-" sign
      val *= -1;
    }

    if( precision == 0)
    {
      strcat(outstr,  ltoa(round(val),temp,10));    //prints the int
part
    }
    else
    {
      unsigned long frac, mult = 1;
      byte padding = precision-1;

      while (precision--)
        mult *= 10;

      val += 0.5/(float)mult;      // compute rounding factor

      strcat(outstr, ltoa(floor(val),temp,10));  //prints the integer
part without rounding
      strcat(outstr, ".\0"); // print the decimal point

      frac = (val - floor(val)) * mult;

      unsigned long frac1 = frac;

      while(frac1 /= 10)
        padding--;

      while(padding--)
        strcat(outstr,"0\0");    // print padding zeros

      strcat(outstr,ltoa(frac,temp,10));  // print fraction part
    }

    // generate width space padding
    if ((widthp != 0)&&(widthp >= strlen(outstr)))
    {
      byte J=0;
      J = widthp - strlen(outstr);

      for (i=0; i< J; i++)
      {
        temp[i] = ' ';
      }

      temp[i++] = '\0';
      strcat(temp,outstr);
      strcpy(outstr,temp);
    }

    return outstr;
}

// ADC Function.
int adc(void)
{
    value=analogRead(A0);
    return value;
}

//Interpolation Function.
void interpolate(int npr ,int n)
{
    if(npr==7)
    {
      if(n==5)
      {
        LcdWrite (1,0x0e);
```

```
    }
    else if(n==4)
    {
       LcdWrite (1,0x0f);
    }
    else if(n==3)
    {
       LcdWrite (1,0x8f);
    }
    else if(n==2)
    {
       LcdWrite (1,0xcf);
    }
    else if(n==1)
    {
       LcdWrite (1,0xef);
    }
    else if(n==0)
    {
       LcdWrite (1,0xff);
    }
}
else if(npr==6)
{
   if(n==4)
    {
       LcdWrite (1,0x07);
    }
    else if(n==3)
    {
       LcdWrite (1,0x87);
    }
    else if(n==2)
    {
       LcdWrite (1,0xc7);
    }
    else if(n==1)
    {
       LcdWrite (1,0xe7);
    }
    else if(n==0)
    {
       LcdWrite (1,0xf7);
    }
}
else if(npr==5)
{
   if(n==3)
    {
       LcdWrite (1,0x38);
    }
    else if(n==2)
    {
       LcdWrite (1,0x3c);
    }
    else if(n==1)
    {
       LcdWrite (1,0x3e);
    }
    else if(n==0)
    {
       LcdWrite (1,0x3f);
    }
    else if(n==7)
    {
       LcdWrite (1,0xe0);
    }
}
else if(npr==4)
{
   if(n==2)
    {
       LcdWrite (1,0x1c);
    }
    else if(n==1)
    {
       LcdWrite (1,0x1e);
    }
    else if(n==0)
    {
       LcdWrite (1,0x1f);
    }
    else if(n==6)
    {
       LcdWrite (1,0x70);
    }
    else if(n==7)
    {
       LcdWrite (1,0xf0);
    }
}
else if(npr==3)//always npr greater than n assumed
{
   if(n==1)
    {
       LcdWrite (1,0xe0);
    }
    else if(n==0)
    {
       LcdWrite (1,0xf0);
    }
    else if(n==5)
    {
       LcdWrite (1,0x83);
    }
    else if(n==6)
    {
       LcdWrite (1,0x87);
    }
    else if(n==7)
    {
       LcdWrite (1,0x8f);
```

```
    }
}
else if(npr==2)
{
   if(n==0)
   {
      LcdWrite (1,0x70);
   }
   else if(n==4)
   {
      LcdWrite (1,0xc1);
   }
   else if(n==5)
   {
      LcdWrite (1,0xc3);
   }
   else if(n==6)
   {
      LcdWrite (1,0xc7);
   }
   else if(n==7)
   {
      LcdWrite (1,0xcf);
   }
}
else if(npr==1)
{
   if(n==3)
   {
      LcdWrite (1,0xe0);
   }
   else if(n==4)
   {
      LcdWrite (1,0xe1);
   }
   else if(n==5)
   {
      LcdWrite (1,0xe3);
   }
   else if(n==6)
   {
      LcdWrite (1,0xe7);
   }
   else if(n==7)
   {
      LcdWrite (1,0xef);
   }

}
else if(npr==0)
{
   if(n==2)
   {
      LcdWrite (1,0x70);
   }
   else if(n==3)
      {
         LcdWrite (1,0xf0);
      }
      else if(n==4)
      {
         LcdWrite (1,0xf1);
      }
      else if(n==5)
      {
         LcdWrite (1,0xf3);
      }
      else if(n==6)
      {
         LcdWrite (1,0xf7);
      }
      else if(n==7)
      {
         LcdWrite (1,0xff);
      }
   }
}

void setup(void)
{
   LcdInitialise();
   LcdClear();
   analogReference(INTERNAL2V5);
   gotoXY(0,0);
   LcdString_anim("------------ ");
   gotoXY(0,1);
   LcdString_anim("TI ADC ");
   gotoXY(0,2);
   LcdString_anim("CONTEST 2014");
   gotoXY(0,3);
   LcdString_anim("--->INSTRUMENTATION -");
   gotoXY(0,5);
   LcdString_anim("OSCILLOSCOPE");
   delay(1000);
   LcdClear();
}

void loop(void)
{

   // Displaying The  Waveform.
   int i,k,avg_no,j,sensorValue,n;
   float voltage, freq,v1,Max;
   for(int rem=0; rem<5; rem++) // To remove 1st five value
provided by ADC.
   {
      sensorValue=adc();
   }
   sensorValue=adc();
   //Step1: Finding Frequency.
   int cnt=0;
   int tst=0;
```

```
   v1 = sensorValue * (2.5 / 1023.0);// comparision value for calculating frequency.
   Max=v1;
   while(tst!=5)
   {
      sensorValue=adc();
      voltage = sensorValue * (2.5 / 1023.0);
      if(fabs(voltage-v1)>0.01999)
      {

         if(Max<voltage)
         {
            Max=voltage;
         }
         cnt++;
      }
      else if(fabs(voltage-v1)<0.01999)
      {
         tst++;
      }
      else if(cnt==1000)
      {
         break;
      }
   }
   if(cnt<3)
   {
      freq=0;
   }
   else if(cnt<1000)
   {
      freq=10000.0/cnt;
   }
   else
   {
      freq=10000;
   }
   //Step2: Calculating No Of Average.
   int test_avg=1;
   while(test_avg==1)
   {
      if(freq>=2500)
      {
         avg_no=1;
         test_avg=0;
      }
      else if(freq>=1250 && freq<2500)
      {
         avg_no=2;
         test_avg=0;
      }
      else if(freq>=850 && freq<1250)
      {
         avg_no=3;
         test_avg=0;
      }
      else if(freq>=625 && freq<850)
      {
         avg_no=4;
         test_avg=0;
      }
      else if(freq>=500 && freq<625)
      {
         avg_no=5;
         test_avg=0;
      }
      else if(freq>=420 && freq<500)
      {
         avg_no=6;
         test_avg=0;
      }
      else if(freq>=360 && freq<420)
      {
         avg_no=7;
         test_avg=0;
      }
      else if(freq>=315 && freq<360)
      {
         avg_no=8;
         test_avg=0;
      }
      else if(freq>=280 && freq<315)
      {
         avg_no=9;
         test_avg=0;
      }
      else if(freq>=250 && freq<280)
      {
         avg_no=10;
         test_avg=0;
      }
      else if(freq>=125 && freq<250)
      {
         avg_no=20;
         test_avg=0;
      }
      else if(freq>=85 && freq<125)
      {
         avg_no=30;
         test_avg=0;
      }
      else if(freq>=65 && freq<85)
      {
         avg_no=40;
         test_avg=0;
      }
      else if(freq>=50 && freq<65)
      {
         avg_no=50;
         test_avg=0;
      }
      else if(freq>=35 && freq<50)
```

```
    {
      avg_no=75;
      test_avg=0;
    }
    else
    {
      avg_no=100;
      test_avg=0;
    }
  }
  // Writing Frequency and Max Valve On The LCD.
  char m[4],f[5];
  Max=((Max/12.5)-0.1)*1000;
  floatToString(m,Max, 1, 0);
  if(freq>999)
  {
    if(freq!=10000)
    {
      floatToString(f,freq/1000, 2, 0);
      gotoXY(0,5);
      LcdString(m);
      LcdString("mv");
      LcdString (",");
      LcdString(f);
      LcdString ("k");
    }
    else
    {
      gotoXY(0,5);
      LcdString(m);
      LcdString("mv");
      LcdString (",");
      LcdString("N/A");
    }
  }
  else
  {
    floatToString(f,freq, 1, 0);
    gotoXY(0,5);
    LcdString(m);
    LcdString("mv");
    LcdString (",");
    LcdString(f);
    LcdString ("Hz");
  }
  // Draw a Reference Line.
  drawLine();
  //Step3: Displaying The Waveform.
  //Each Increment In The Row Of Pixel Corresponds To A Inc In The Voltage Of 0.0625V.
  int count=0;
  int bank=0;
  int bankpr=0;
  int npr=0;
  for(i=0; i<=83 ; i++)
  {
    //Mapping Digital Signal Value to Its Corresponding Analog Value.
    voltage=0;
    for(j=0; j<avg_no; j++)
    {
      sensorValue=adc();
      voltage = voltage + sensorValue * (2.5 / 1023.0);
    }
    voltage=voltage/avg_no;
    // Max Value Of n Is 40.
    //If n<20, Plotted Below The Ref Line
    // And If n>=20 , Plotted Above The Ref Line.
    n=voltage/0.0625;
    //Removing all pixel colured previously in a ith column if any.
    for(int clr=0; clr<5; clr++)
    {
      gotoXY(i,clr);
      if(clr!=2)
      {
        LcdWrite (1,0x00);
      }
      else
      {
        LcdWrite(1,0x10);
      }
    }
    int n1=0;
    int npr1=0;
    //Colouring The Pixel Corresponding To The Value Of Signal.
    if(n>=0 && n<=7)
    {
      gotoXY(i,4);
      bank=4;
      n1=n;
      markpixel(n,4);
    }
    else if(n>=8 && n<=15)
    {
      gotoXY(i,3);
      bank=3;
      n1=n;
      n=n-8;
      markpixel(n-8,3);
    }
    else if(n>=16 && n<=23)
    {
      gotoXY(i,2);
      bank=2;
      n1=n;
      n=n-16;
      markpixel(n,2);
    }
    else if(n>=24 && n<=31)
    {
```

```
    gotoXY(i,1);
    bank=1;
    n1=n;
    n=n-24;
    markpixel(n,1);
}
else if(n>=32 && n<=39)
{
    gotoXY(i,0);
    bank=0;
    n1=n;
    n=n-32;
    markpixel(n,0);
}
else if(n==40)
{
    bank=0;
    n1=39;
    n=7;
    gotoXY(i,0);
    markpixel(7,0);
}
else if(n>=41)
{
    gotoXY(0,1);
    LcdString ("OUT OF RANGE");
}
if(abs(npr1-n1)>=2 && i>0)
{
    if(bank==bankpr)
    {
        gotoXY(i,bank);
        interpolate(npr,n);
    }
    else if(bank>bankpr && bankpr-bank==1)
    {
        gotoXY(i,bankpr);
        interpolate(npr,0);
        gotoXY(i,bank);
        interpolate(7,n);
    }
    else if(bank>bankpr && bank-bankpr>1)
    {
        gotoXY(i,bankpr);
        if(bank-bankpr==2)
        {
            gotoXY(i,bankpr+1);
            LcdWrite (1,0xff);
        }
        else if(bank-bankpr==3)
        {
            gotoXY(i,bankpr+1);
            LcdWrite (1,0xff);
            gotoXY(i,bankpr+2);
            LcdWrite (1,0xff);
        }
        else if(bank-bankpr==4)
        {
            gotoXY(i,bankpr+1);
            LcdWrite (1,0xff);
            gotoXY(i,bankpr+2);
            LcdWrite (1,0xff);
            gotoXY(i,bankpr+3);
            LcdWrite (1,0xff);
        }
        gotoXY(i,bank);
        interpolate(7,n);
    }
    else if(bank<bankpr && bankpr-bank==1)
    {
        gotoXY(i,bankpr);
        interpolate(npr,0);
        gotoXY(i,bankpr);
        interpolate(7,n);
    }
    else if(bank<bankpr && bankpr-bank>1)
    {
        gotoXY(i,bankpr);
        interpolate(npr,7);
        if(bankpr-bank==2)
        {
            gotoXY(i,bankpr-1);
            LcdWrite (1,0xff);
        }
        else if(bankpr-bank==3)
        {
            gotoXY(i,bankpr-1);
            LcdWrite (1,0xff);
            gotoXY(i,bankpr-2);
            LcdWrite (1,0xff);
        }
        else if(bankpr-bank==4)
        {
            gotoXY(i,bankpr-1);
            LcdWrite (1,0xff);
            gotoXY(i,bankpr-2);
            LcdWrite (1,0xff);
            gotoXY(i,bankpr-3);
            LcdWrite (1,0xff);
        }
        gotoXY(i,bank);
        interpolate(0,n);
    }
  }
  npr1=n1;
  npr=n;
  bankpr=bank;
 }
}
```

APPENDIX C – BILL OF MATERIALS

Give a table which shows the name of the hardware/software component, number of components in the project, cost per component, whether the component is a TI/non-TI component, total cost of the component, and the total cost of all components.

	Component	Manufacturer	Cost per component	Quantity	Total cost of component	TI Supplied/ Purchased
1	OPA2313	TI	NIL	1	NIL	Supplied
2	INA333	TI	NIL	1	NIL	Supplied
3	TPS60310	TI	NIL	1	NIL	Supplied
4	MSP430G2553 Launchpad	TI	9.99$	1	9.99$	Supplied
5	Nokia 5110 LCD Breakout Board	blitz_tech world (ebay.in)	7$	1	7$	Purchased
	Total Cost of the Project				**17$**	