Project Report

EE318

Pressure Sensing Based Keypad

07007027 SHAMIT MONGA shamitmonga@iitb.ac.in

07D07019 JAY PARIKH jayparikh@iitb.ac.in

07D07021 AAKASH PATIL aakashpatil@iitb.ac.in

07007037 NAGA VAMSI KRISHNA TALUPULA tnvkrishna@iitb.ac.in

Faculty Advisor – Professor Dipankar

Instructor in charge- Professor Jayanta Mukherjee(I)

Professor M.Shojaei Baghini(A)

Electrical Engineering Department

IIT Bombay

Spring 2010

Motivation

The technology that we use in our day to day lives – be it mobiles, laptops, ipods etc—all of these have shrunk to really small handheld devices. But the human hand that operates these gadgets has stayed where it was. The keypads used to input data to these devices are really small. Our initial thoughts were –Why don't we create an alternative keyboard that is portable, not as bulky as the desktop ones and is able to solve this problem? Then following some projects over the internet we started thinking of a virtual keyboard- where there are no keys and you touch with your hand over a piece of paper and image processing is used to detect where you touched. But that idea turned out to be rather flashy and impractical considering the original ideas from where it sprung up. Then we started doing some research about the existing technologies in touch applications looking at various parameters like sensitivity, the science behind it, robustness etc. Following these ideas up with our professors we landed up on pressure sensing using a strain gauge. The existing touch technologies use capacitive sensing – wherein the conductivity properties of the human hand are used to detect the point of touch. Our professors motivated us to explore the potential of pressure based sensing in touch based applications. And from there on after several rounds of discussions, searching and some trial and error was this idea of working with strain gauges was taken up.

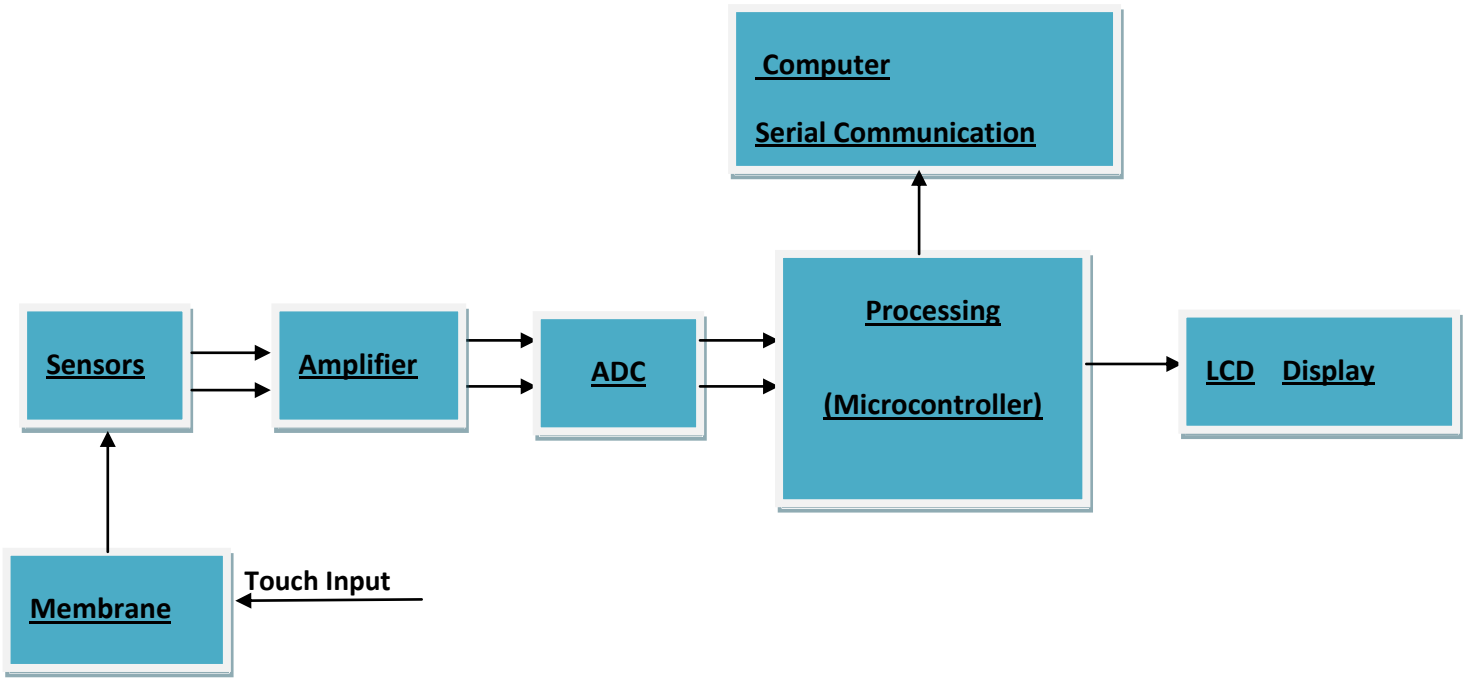
Our project aims to explore the utility of pressure based sensing. Pressure based technology offers a high degree of robustness and can find utility on ATMs, ticket vending machines etc. For pressure technology to really take on the other available technologies we need to integrate the membrane used for such application with micro-cantilevers as opposed to the macro-cantilevers that we are currently using. So in essence our project is more of a prototype displaying the potential of using such a technology.

1. Introduction

The product is designed to work as a key pad that uses strain gauge based cantilevers to detect the key press position. The aim is to implement a num pad (total 16 keys). The key board implementation is based on the proportional change in resistance of strain gauges with applied deflection strain.

The sensors (two in number) sense the strain on the membrane and generate an analog voltage that is proportional to the strain caused by deflection. The microcontroller (Atmega 16) has ADC's that samples the values of these voltages. Characterization of the keypad is then used to define a lookup table with key value regions. Based on these regions the position of key press is detected. The key press value is then displayed in the board LCD and a serial communication option is also available for computer interfacing.

2. System Block Diagram



Major System Blocks

1) Membrane –In our case a laminated piece of cardboard
2) Sensors-Strain Gauge based Cantilevers
3) Opamp's for Strain Gauge output amplification.
4) Microcontroller(ATMEGA16) which does the processing and ADC conversion
5) LCD Display(Hitachi Hd44780)
6) Serial Communication Link

3.Main circuit components used:

1) **Strain Gauge**.

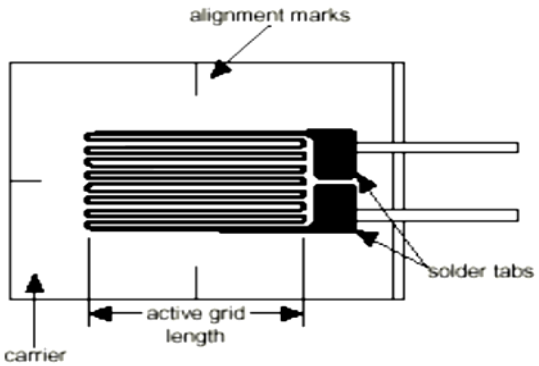
A strain gauge is a device used to convert strain a mechanical quantity to electrical voltage for measurement and processing. The gauge is basically a resistor attached to a support foil. This arrangement is attached to the deformed object by a suitable adhesive (we used Araldite). As the object is deformed the foil bends, causing the wire's length to change which changes its electrical resistance. This resistance change, which can be measured using a wheatstone bridge configuration, is almost proportionally related to the strain by the quantity known as the gauge factor.

(Gauge factor is defined as the ratio of fractional change in electrical resistance to the fractional change in length (strain)):

$$G = \frac{((\Delta R)*l)}{(R*(\Delta l))}$$

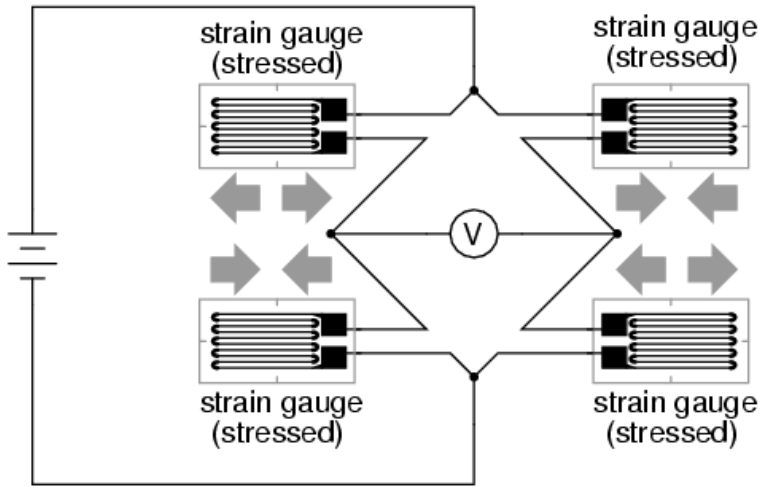
The specifications of the strain gauges used are-

a. Resistance = 330 Ω (in the absence of strain)
b. Gauge factor = 2



The full bridge configuration, in which 4 strain gauges are needed per cantilever, has been employed because of its greater sensitivity over the other configurations, cancellation of the non-linear parts of the strain gauge response, compensation for temperature. The strain gauges are used in wheat stone bridge configuration.

Full-bridge strain gauge circuit



In the above diagram the top two strain gauges and the bottom two strain gauges are mounted on opposite sides of the metallic strip.

2) **Microcontroller**:

The microcontroller, Atmega 16, has been used for ADC conversions of the output voltages of the wheatstone bridge and the soft decisions that are needed to be taken to implement the logic associated with the input device. The microcontroller was chosen keeping in mind the memory required for this code, the cost, availability of LCD header files and ready availability in the lab. Atmega 16 is high performance, low power AVR 8-bit microcontroller. Its important features used by us are:

a. Advanced RISC architecture for faster performance
b. 8-channel 10 bit ADC
    - 8 single-ended channels
    - 2 differential channel with a programmable gain at 1x,10x,200x
c. Four 8-pin ports
d. Internal clock of 1 MHz
e. Byte oriented two wire serial interface
f. 16k bytes of flash memory
g. 512 bytes of EEPROM
h. 1k bytes of internal SRAM

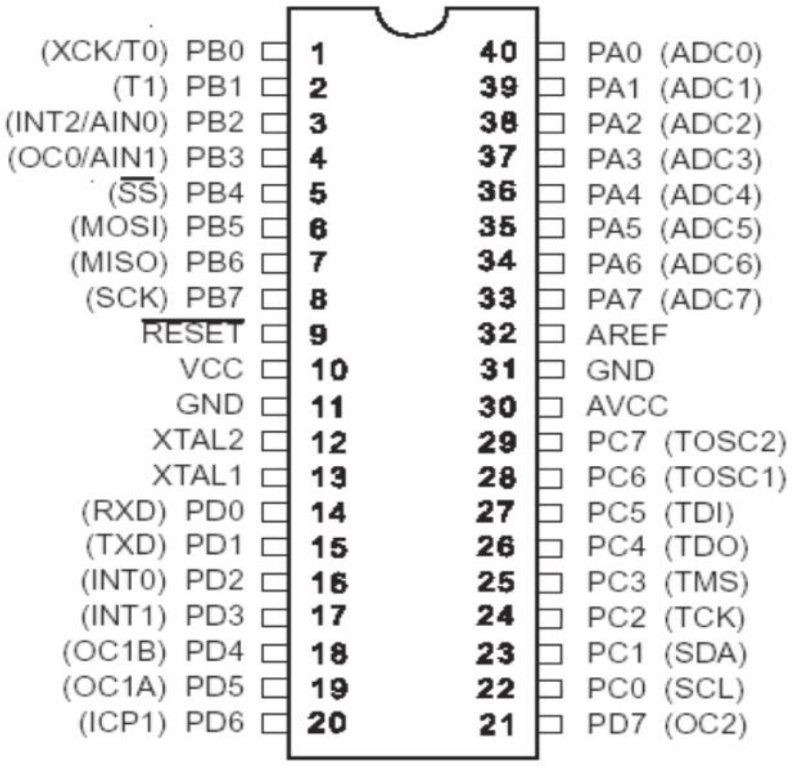


Fig. Atmega 16 Pin Configuration

In our circuit ADC fabricated on the Microcontroller chip to convert the voltage values from the wheatstone bridge circuit into digital values and a programmable logic to determine the position of the "press" from those values.

3) **LM 7805 voltage regulator**:

The LM78XX series of three terminal regulators are available with several fixed output voltages making them useful in a wide range of applications. These regulators are used in a variety of applications which include logic systems, instrumentation and other solid state electronic equipment. Although designed primarily as fixed voltage regulators, these devices can be used to obtain adjustable voltages and currents using external components.

In our circuit, we use this (LM 7805) as a power source to the microcontroller.

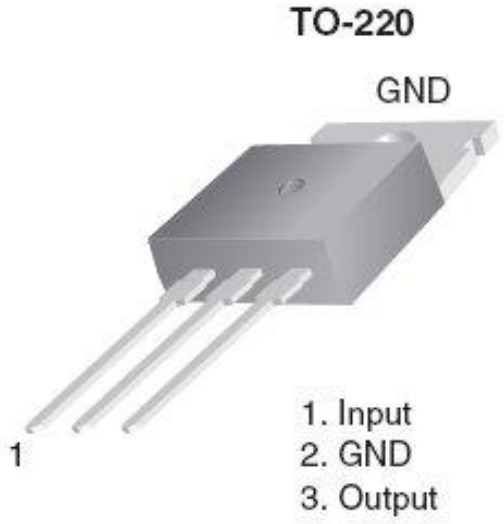
Voltage Range

LM7805C - Input-7 to 12 V

Output-5V

Features : -

a. Output current in excess of 1A
b. Internal thermal overload protection
c. No external components required
d. Output transistor safe area protection
e. Internal short circuit current limit
f. Available in the aluminum TO-3 package



4) <u>Keypad (input device):</u>

The keypad (input device) is made of lamination material. The most important parameters that decided the choice of the material for the membrane were performance under strain, weight and fabrication ease. We tried several materials like tin sheet, flex, cellophane and various configurations to connect them in (see Images), before we finally decided to work with a laminate material in the final configuration. The kind of material that we wanted was one that was light and that didn't create a permanent local depression when touched. The lamination material based membrane has the advantage of being transparent, and is suitable to be used as a touch screen. For the purpose of our project the membrane has been divided into multiple areas which represent different keys on a keyboard.

5) **<u>Amplifier circuit</u>**:

The voltage signals obtained from the wheatstone bridge configuration were in milivolts range and need to be amplified before feeding them to the ADC of the microcontroller. A op-amp based amplification has been employed. An opamp by Philips (LM324N) has been used primarily because of its low, and predictable offset output voltage.

6) **<u>Serial communication</u>**:

Serial communication protocol has been chosen to send the key press data to the computer. Max232 chip and the associated circuit was used to convert the TTL logic levels to RS232 logic levels. A DB9 connector is used to connect the MAX232 with computer.

4.Circuit description and working:

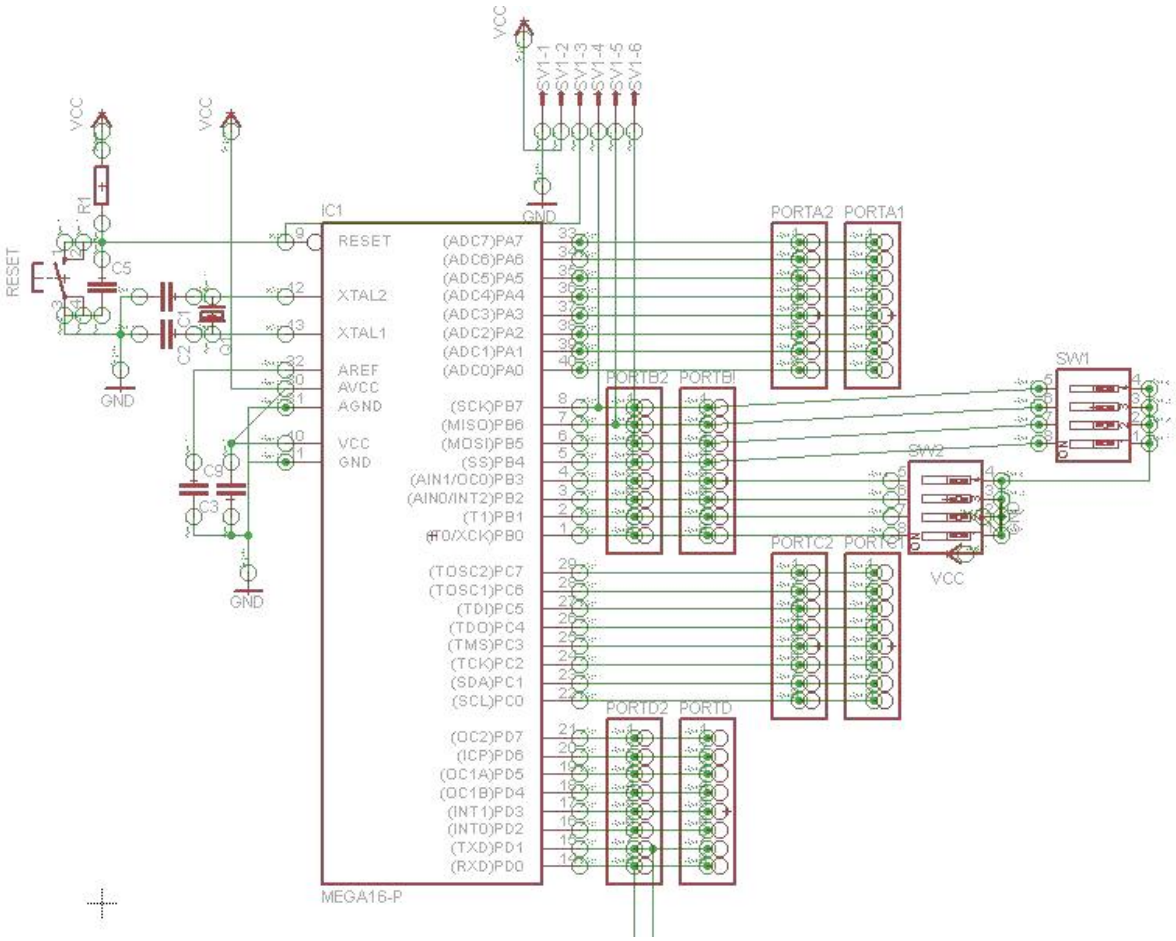
1) **<u>Microcontroller Circuit</u>**:

A microcontroller (Atmega16) is used to convert the analog voltage values from the wheatstone bridge, into digital values using the inbuilt ADCs and to implement the necessary logic required

Pin description :

a. PD0(RXD) and PD1(TXD) are used for serial communication.
b. The pins of PORTA has been used as for taking input for ADC conversion
c. Pins 11 and 31 were grounded and pin 30 and 10 were connected to Vcc(5V) as per the requirement
d. PORTC and PORTD have been used for interfacing with LCD
e. PORTB has been used for taking the input from switches, which are used for calibration of the device

Given below is the diagram of the circuit associated with the microcontroller

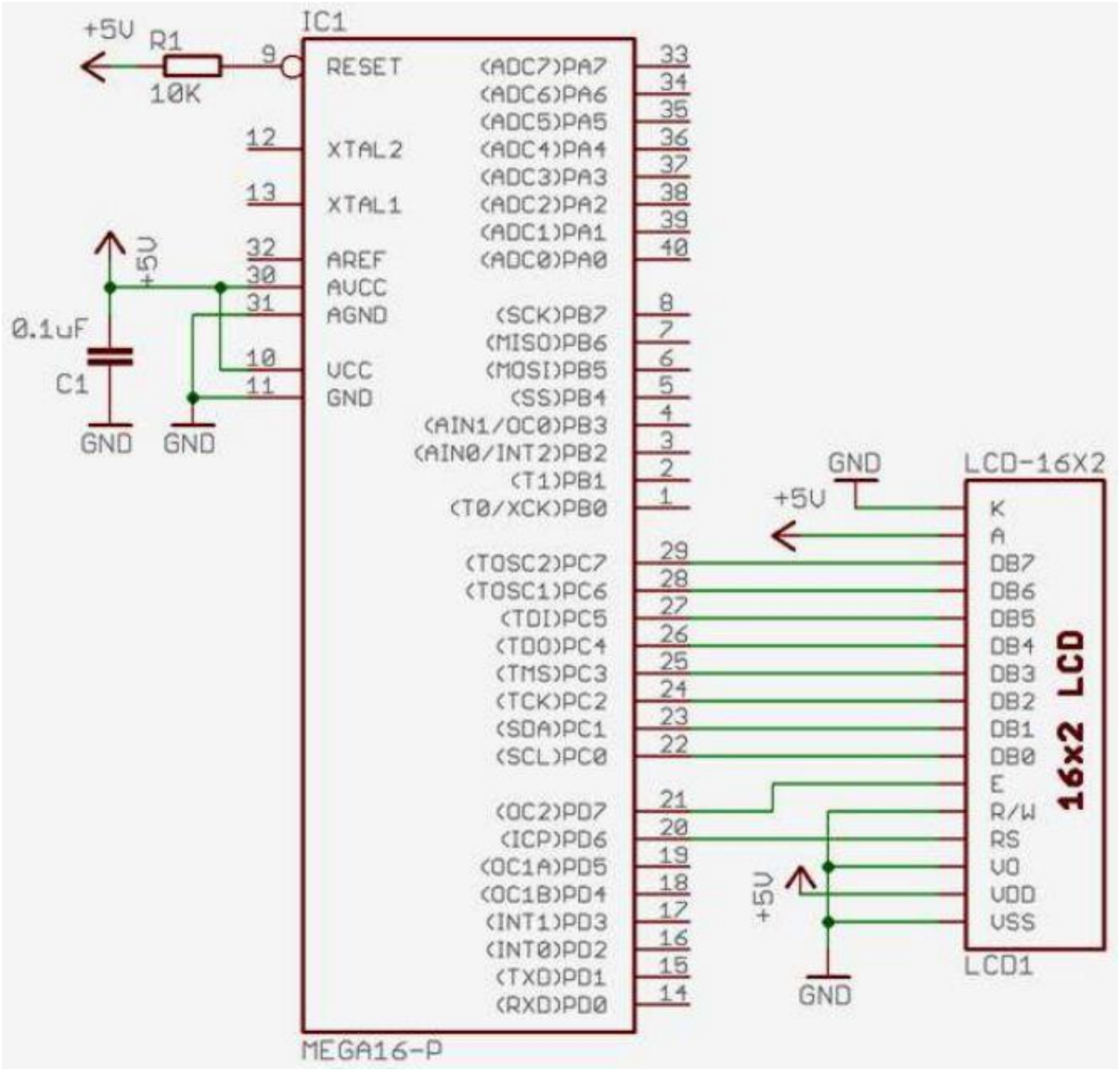


2) **LCD Display (HD44780)**:

The 16 pin LCD Display has been used to assist the user in calibration of the keypad, and to also to display the switch pressed (which was used during the debugging of the circuit, and later replaced by serial communication to the computer).

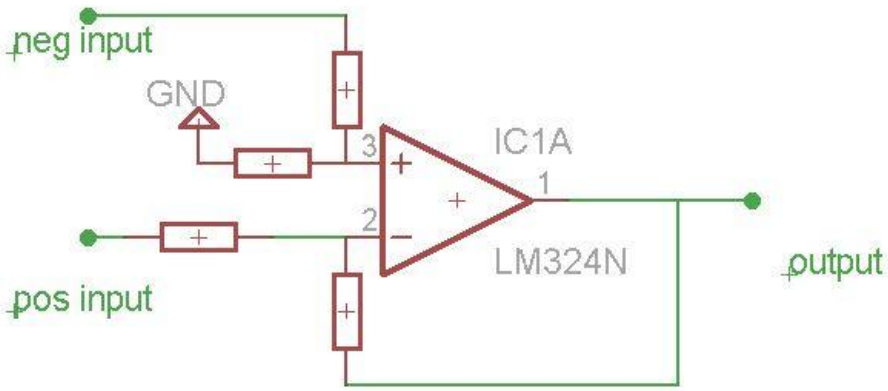
a. It is a 16x2 character LCD, organised as 2 rows and 16 columns.
b. Pin 1,3 need to be grounded
c. Pin 4,5,6 are the control pins. (Since we are only using the write to LCD functionality pin number 5 i.e. R/W has been grounded)
d. Pin 7 to pin 14 are the data pins.
e. The voltage level difference between pin 15 and pin 16 determines the LCD screen brightness (the voltage level difference used in our case is 5V)

Given below is the circuit diagram associated with LCD Display (the pins are numbered from 1 to 16 bottom-up).



3) **Amplifier circuit**:

Given below is the schematic diagram of the opamp based amplifier used in the circuit.



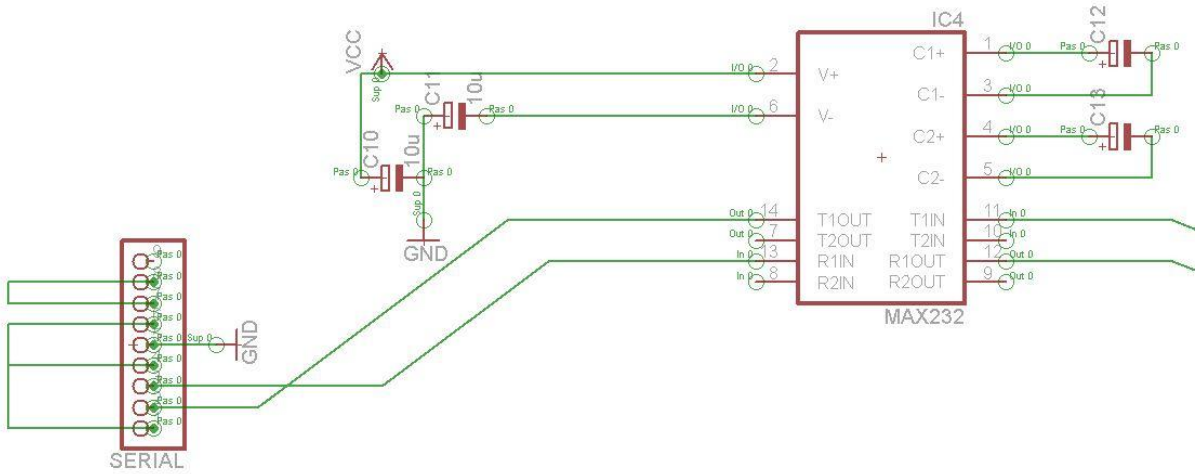
4) **Serial communication**:

A typical circuit suggested in the MAX232's data sheet has been used to convert the TTL logic voltage levels to RS232 voltage levels. And DB9 connector is used to connect the serial port of computer with the serial output pins of MAX232.

DB9 connections:

a. Pin number 3 connected to pin 14 of max232(T1OUT)
b. Pin number 4 connected to pin 13 of max232(R1IN)
c. Pin number 5 connected to ground

Given below is the circuit diagram to use the MAX232:



Material and Design Stages of the Project

1) We started off by testing on an already fabricated strain gauge based cantilever. We used the ADC that comes in built with the microcontroller ATMEGA 16. This testing cleared the use of cantilevers for position detection. The sensitivity was found to be adequate.
2) A rigid membrane based solution was also initially considered, and the required mechanical equations, which calculate the location of press based upon the voltage level readings from cantilevers, were solved. The tilt of the keypad, in such a place, plays a crucial role in determining the point of press. Since, we wanted the keypad to be used in situations like touch screen to laptop (vertically mounted) as well as in railway counters (fairly horizontal configuration), we needed to think of a different idea. Moreover, at least 3 cantilevers are needed in such a situation.
3) Then flexible membrane was used as the input device and various configurations, materials were tested for design appropriateness. To start with a full soft membrane based approach was developed in the following four cantilever design.

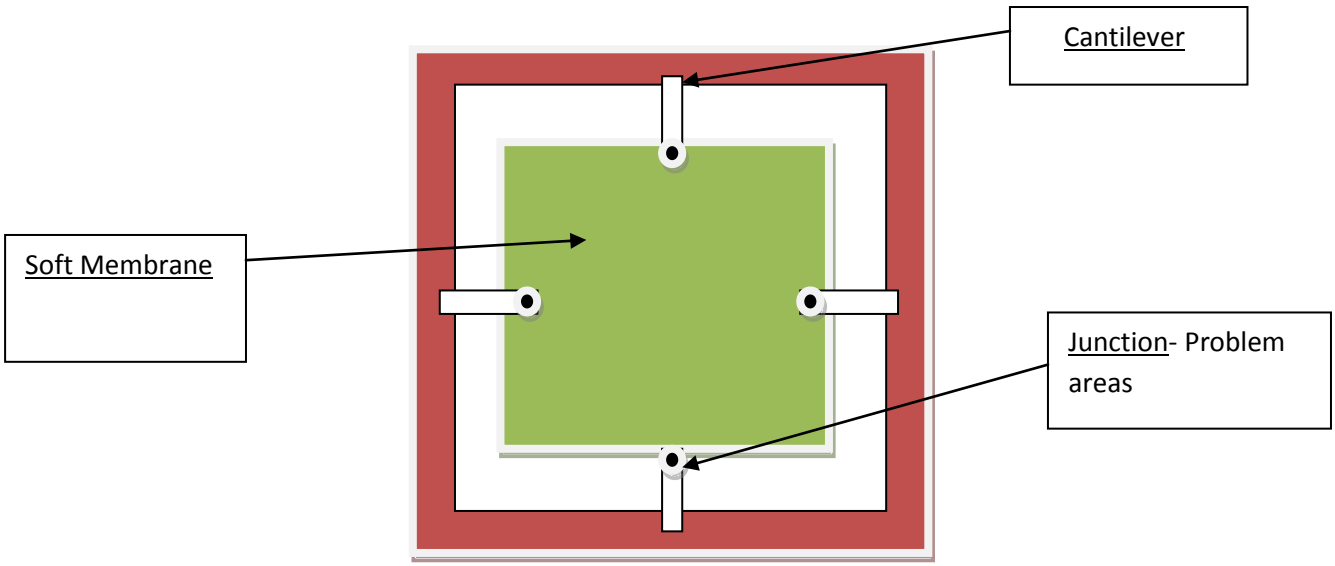
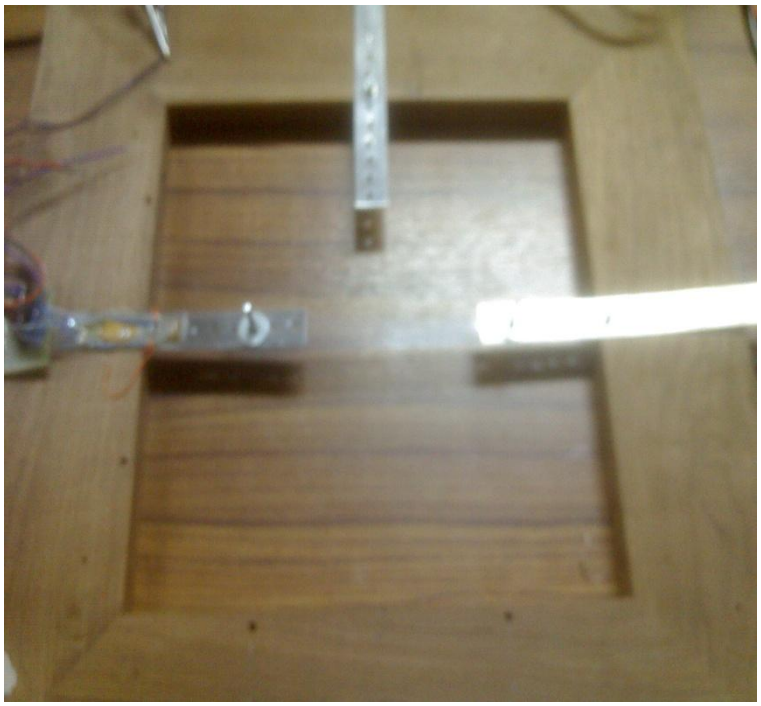


Fig. Four Cantilever Configuration

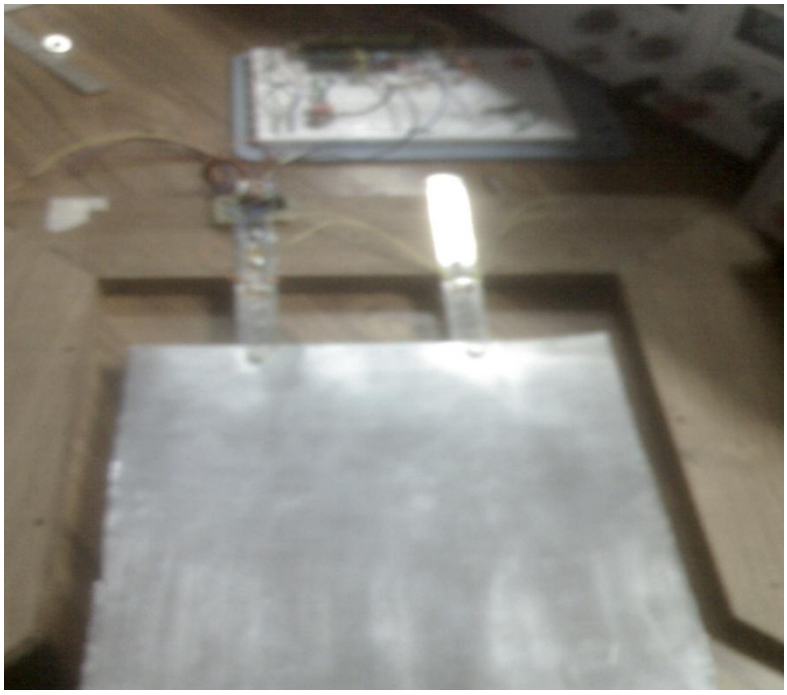
The problem in this configuration was an appropriate material. We tried to work with Cellophane and flex plastic. These were put in tension on the cantilever using skrews . In both these cases localised stretch at the membrane cantilever joints caused the material to deform permanently.



Img1.Testing of Linear Keypad in the 4 cantilever configuration material cellophane

4) Upon the failure of a soft membrane based approach we decided to try out with tougher but depressible materials. We started again with an aluminium sheet of appropriate thickness in a two cantilever configuration.

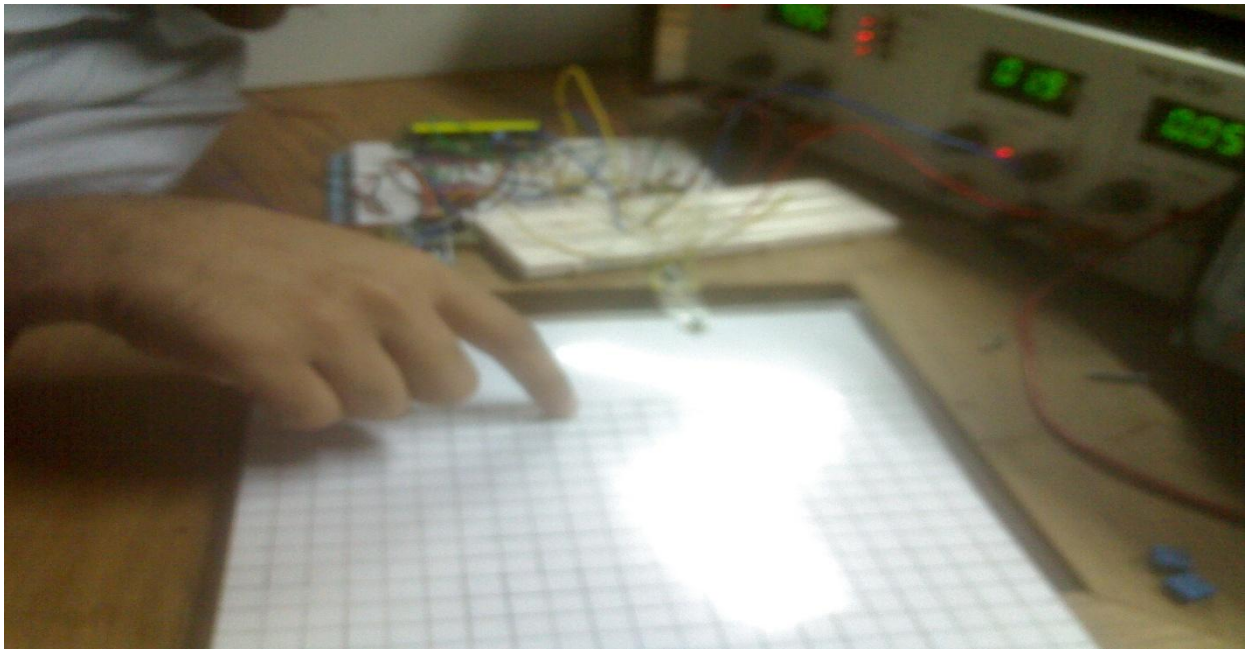
Img2. Aluminium Sheet Based Two Cantilever Approach



This material was suited for this application, with appropriate flexibility for a key press and a constant non deformed value. The problem was with its different equilibrium states which kept on changing the readings for the same key.

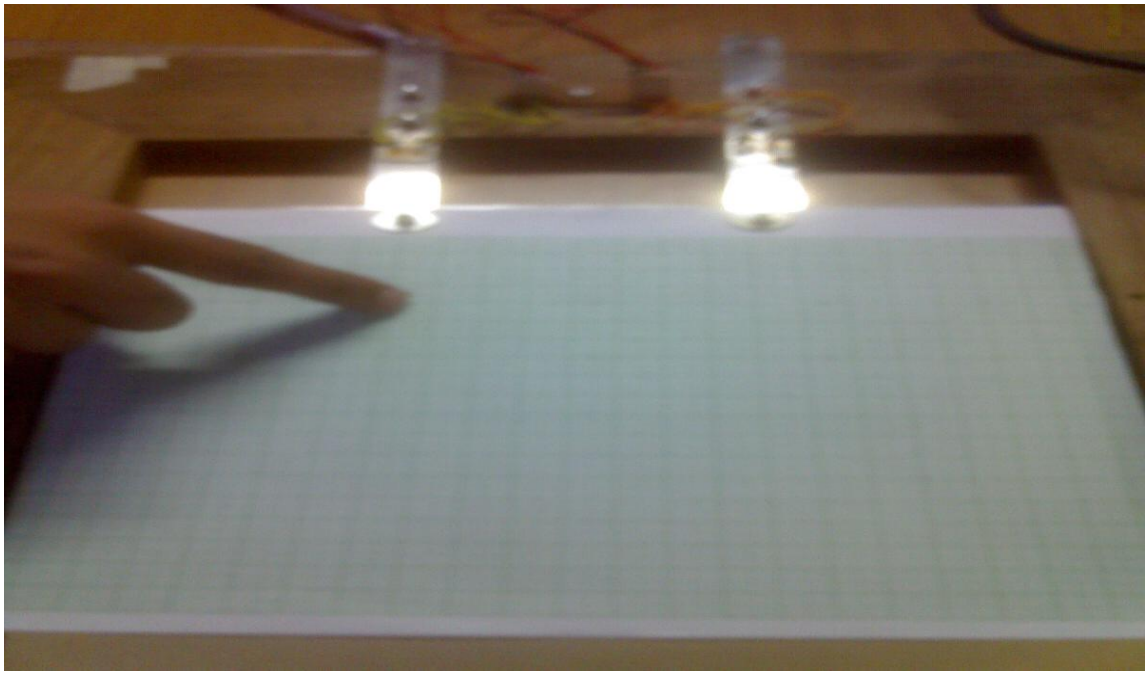
5) After the aluminium based approach a laminated sheet was used instead of the key pad. The advantages were that laminate was flexible, returned to its undeformed position and caused less vibrations.

Img3. Thin Paper based Lamination



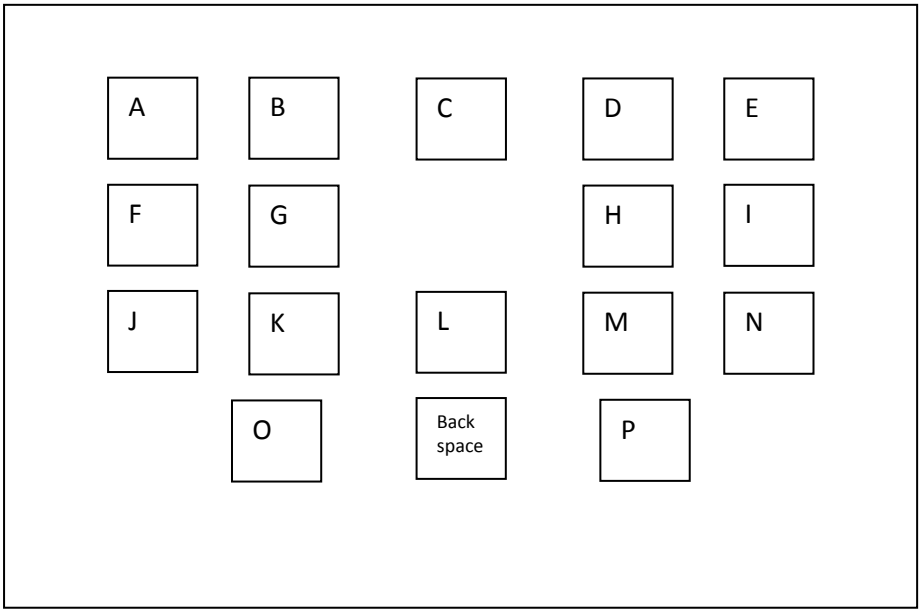
The initial thin paper based approach was not appropriate because of the highly flexible nature of the lamination. It was then decided to use a thicker material like cardboard during lamination. This arrangement fulfilled all the requirements and was used in the final product.

Img4. Cardboard based Lamination approach graph paper on top for characterization purpose

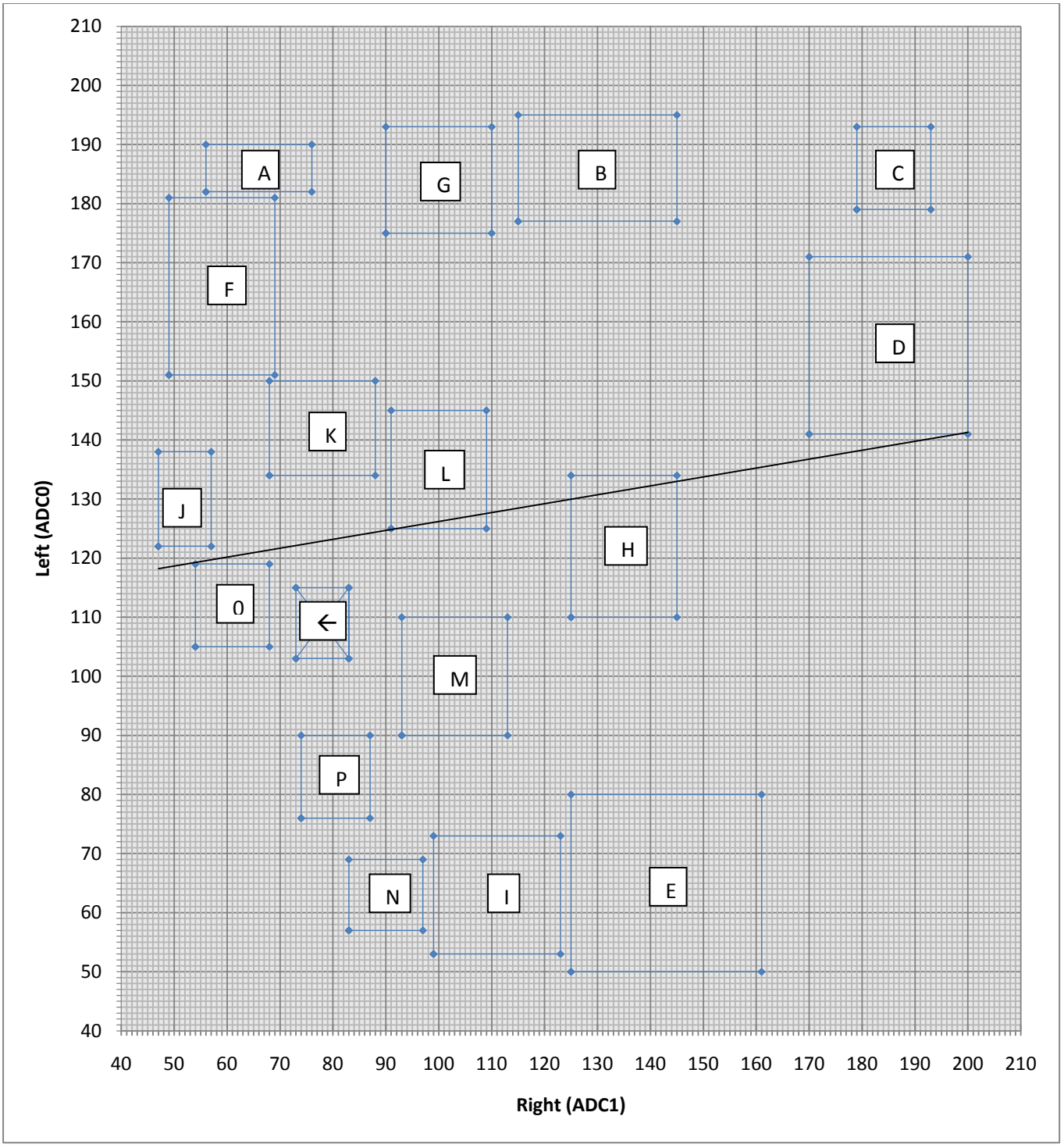


Characterization Procedure

For determining the soft decisions for the individual keys we used graph papers for defining keys on the pad (as seen in **Img4**). Each key was defined for the area of 2cmx2cm. Based on the LCD display values we determined the maximum and minimum values for each of the strain gauges for individual keys. These were then used to determine the decision regions by plotting on a graph paper. Tolerances were added to reduce the chances of error.



Key arrangement on Keypad



Range of ADC values obtained by mapping of ADC values for each 2x2 cm key area and taking into consideration suitable tolerance. We see each key on Keypad each mapped one-to-one onto ADC value area graph

Prototype Cost

IC/Euipment/Material	Quantity	Cost
Strain Gage	2 sets of wheatstone bridge	2x150=Rs.300
Microcontroller Atmega16	1	Rs.165
LCD Display (HD44780)	1	Rs.160
LM324	1	Rs.10
Max232	1	Rs.20
DB9 Male connector	1	Rs.10
7805 voltage regulator	1	Rs.10
LEDs, Resistors, Capacitors, switches, buttons	many	Rs.25
PCB (including various 4 and 8 pin connectors)	2	2x100=Rs.200
Mechanical assembly/ chemicals	Various	Rs.100
Total		Rs.1000

AVR Code Descriptions

For coding have used two readymade routines

lcdroutines.h : Contains predefined functions for LCD

uart0_at16_routines.h : Contains predefined functions for serial communication

Code

We have predefined values for minimum value of ADC and arrays of predefined values for centre and tolerance of 17 keys

Main function

We at first disable JTAG by instruction MCUCSR|=_BV(JTD)

Then we initialised ADC, LCD, UART by corresponding initialisation commands adc_init(), lcd_init(), uart0_init()

We clear the LCD by lcd_clear() and bring LCD cursor to start by lcd_home()

Then depending on state of input at PB3

If PB3=HI, we go into setting mode where

We display word “set” in upper line of LCD indicating user that he is in setting mode

We move LCD cursor to start of 2nd line by move_to(0,1) and display the key presently being set by display_char(key[i])

While PB0 is LO user can change values for corresponding key and once he makes PB0 HI again ADC values get locked

If PB3=LO, we go into operational mode where

We continuously take ADC readings and once atleast one of the ADC values increases above minimum values (indicating user has started to press) set for them it enters infinite while loop while(1)

In infinite while loop we try to find the maximum of ADC readings (readings corresponding to maximum press to upto bottom of surface). We store old ADC values in variable 'adc' and new ADC values in variable 'var'. When key is being pressed we expect new('var') to be greater than old('adc') and when released we expect old('adc') to be greater than new('var'). So when 'var' is less than 'adc' we break out of infinite while loop

With above values of ADC we compare them and observe if they are within tolerance range of any of the keys. If yes we display that key by function LCD(i)

Future Work

There are many improvements in this project that could not be done because of the shortage of time. We could have experimented with more implementations using different sensors (like pressure sensors) other than strain gauges to measure pressure/strain to see what works best.

Another thing could be to make software so that the user can modify/customise the current interpretation pattern for keys (as in what stands for what and also modify dimensions). We could also add USB functionality (instead of the serial communication currently provided) to this device so that this can become a plug and play device.

Further, in a major overhaul, we could have gone for integrating the membrane with microcantilevers that would raise the sensitivity of this device by several orders of magnitude and give a precision close to microns. Then this device will really be able to take on the current capacitive sensing technology. However such a step also increases the complexity and time required for this project by orders of magnitude.

Conclusion

Through this project we have explored various membrane materials and designs for there suitability of use to make a flexible keypad. We found that the properties like flexibility, stiffness, capacity to return to original position after deformation, and fabrication ease are detrimental while deciding the membrane material. For design considerations fabrication facilities available, cost and key pad requirements were important. Overall we feel that strain gauge based systems with there resolution can be efficiently used to make accurate sturdy and easily serviceable keypads. Also micro-strain gauge based micro-cantilevers can be used for their high sensitivity to extend the application to touch screens.

Word of Thanks

Needless to say, this project would not have been complete without the guidance and support of several people. We would especially like to thank our guide Prof Dipankar who provided us with the initial idea of using strain gauges. We would also like to thank professors- Prof P.C. Pandey, Prof Jayanta Mukherjee and Prof M.Shojaei Baghini for their invaluable help in giving ideas and general guidance during the project.

We would also like to thank the lab staff at WEL 4 & Mrs Madhumita Date, Mr V.K.Tandon for their cooperation. Here we would especially like to mention Naresh Sir for enthusiastically helping us with our with the soldering of delicate strain gauges and IC's which helped us solve a lot of our problems.

Appendix

User Guide

1) Connect 2-pin female relimate connector to male connector-1 on PCB-1 for main external power source.
2) Press reset button to toggle between ON-OFF state. On status is also indicated by red LEDs glowing.
3)To display the key pressed on the LCD, connect LCD in 16-pin connector on PCB-2.
4)To display the key pressed on the computer using hyperterminal, connect female DB9 connector to male DB9 connector in computer and other end of it to 4-pin male connector-2 on PCB-2 using 4-pin female relimate connector. In hyperterminal set Baud Rate=2400, Data Length=8, Parity Bits= None, Stop Bits=2
5) Make DIP Switch-4 OFF to go into setting mode and DIP Switch-4 ON to go into working mode
6) We have predefined set of values for each key obtained through characterization so user can directly go into working mode
7) In setting mode line 2 of LCD displays key which is currently being set. Toggle DIP switch-1 ON and OFF to set next key. If want to skip particular key make DIP switch-2 ON. Set ADC values of key by keeping DIP Switch-1 ON and pressing to maximum (to bottom of surface) and lock ADC values by making DIP Switch-1 OFF again. Having set all keys make DIP Switch-4 ON to go into working mode
8) In working mode press keys such that key is pressed to maximum(to bottom of surface) and maintain in that state for atleast 1 ms and release it. Maximum value of ADC are noted on releasing key. Depending on this maximum ADC reading key pressed is detected and displayed on LCD and/or hyperterminal
9) If wrong key is detected or otherwise need be use backspace key provided to clear last key displayed

Program Code

```
// ************************************************************
// ***  LCD_ROUTINES HEADER FILE ********
// ************************************************************

//Connections between uC and LCD
#define DATA_DDR   DDRC
#define DATA_PORT        PORTC

#define CONTROL_DDR          DDRD
#define CONTROL_PORT     PORTD
#define RS               PD6
#define E                PD7

//Functions for interfacing to a 16x2 LCD

void lcd_clear(void);
void lcd_home(void);
void lcd_command(unsigned char command);
void lcd_init(void);

void display_char(unsigned char data);
void display_byte(unsigned char num);
void display_int(unsigned int num);

void move_to(unsigned char x,unsigned char y);



// ************************************************************
// ***  UART_ROUTINES HEADER FILE ********
// ************************************************************

#define TX_NEWLINE            {transmitByte(0x0d); transmitByte(0x0a);}

#define CHAR 0
#define INT  1
#define LONG 2


void uart0_init(unsigned int);
unsigned char receiveByte(void);
void transmitByte(unsigned char);
void transmitString_F(char*);
void transmitString(char*);
void transmitHex( unsigned char dataType, unsigned long data );

// ************************************************************
// ***  MAIN FUNCTION********
// ************************************************************

#include <avr/io.h>
#include "lcdroutines.h"
#include "UART_routines_at16.h"
#include <util/delay.h>

unsigned char key[] = {"ABCDEFGHIJKLMNOxP"};
```

```
unsigned int setL[17]=
{185,185,185,159,65,
168,184,122,64,
130,141,136,96,60,
106,103,58};
unsigned int setR[17]=
{73,145,185,186,159,
66,108,150,128,
60,86,111,112,100,
70,85,74};
unsigned int tolL[17]=
{3,5,3,15,15,
 12,7,10,10,
 10,8,8,10,8,
 6,8,6};
unsigned int tolR[17]=
{10,12,3,7,15,
 8,7,10,10,
 8,8,10,8,7,
 5,8,6};
unsigned int minL=60,minR=65,adcL,adcR,varL,varR,thL,thR;
unsigned char setting[] = {"set "};
unsigned char err[]={"   err   "};
unsigned char pre[]={"pre      "};
unsigned char nxt[]={"      nxt"};

unsigned int adc_read(unsigned char channel)
{
      ADMUX   = (ADMUX&0xe0) + channel;
      ADCSRA |= _BV(ADSC); // Start conversion
      while((ADCSRA & _BV(ADSC)) != 0) {}; // Do nothing until conversion
is done
      return(ADCH); // Return upper 8 bits
}

void adc_init(void)
{
      DDRA   = 0x00;
      PORTA  = 0x00;
      ACSR   = _BV(ACD);
      ADCSRA = _BV(ADEN) |_BV(ADPS2)|_BV(ADPS1);// Enable ADC with
prescaler = 64
      ADMUX  = _BV(REFS0) |_BV(ADLAR); //left adjust,internal 5V Voltage
Reference with external capacitor at AREF pin
}

/* Main function */
int main(void)
{
      MCUCSR|=_BV(JTD);
      MCUCSR|=_BV(JTD);
      MCUCSR|=_BV(JTD);
      MCUCSR|=_BV(JTD);
      DDRB   = 0xf0;
      PORTB  = 0xff;
      uart0_init(25);   //baud rate 2400bps for freq = 1MHz
      lcd_init();
      adc_init();
      unsigned int i,j;
      while(1)
      {
```

```
            while((PINB&0x08)==0x08)      // make PB3 low for settings
            {

                  lcd_clear();
                  lcd_home();
                  for(i=0;i<sizeof(setting)-1;i++)
display_char(setting[i]);
                  for(i=0;i<17;i++)
                  {
                        move_to(0,1);
                        while((PINB&0x01)==0x01);     //make PB0 HI-LOW to
set next key
                        display_char(key[i]);
                        while((PINB&0x03)==0x00)      //make PB1 HI to skip
                        {
                              setL[i]= adc_read(0);
                              setR[i]= adc_read(1);
                              move_to(0,0);
                              display_int(setL[i]);
                              move_to(8,0);
                              display_int(setR[i]);
                              _delay_ms(25);
                        }
                        if(setL[i]<minL) minL=setL[i];
                        if(setR[i]<minR) minR=setR[i];
                  }
            }

            if(minL>8)  minL=minL-8;
            if(minR>8) minR=minR-8;

            while((PINB&0x08)==0x00)      //keep PB3 high for operation
            {
                  move_to(0,1);
                  for(j=0;j<sizeof(nxt)-1;j++) display_char(nxt[j]);
                  while(1)
                  {
                        adcL= adc_read(0);
                        adcR= adc_read(1);
                        if ((adcL>minL)&&(adcR>minR))
                        {
                              while(1)
                              {
                                    varL= adc_read(0);
                                    varR= adc_read(1);
                                    if ((varL<adcL)&&(varR<adcR)) break;
                                    adcL=varL;
                                    adcR=varR;
                              }
                              break;
                        }
                  }

                  for(i=0;i<17;i++)
                  {
                        if(adcL>setL[i]) thL=adcL-setL[i];
                        else thL=setL[i]-adcL;
                        if(adcR>setR[i]) thR=adcR-setR[i];
                        else thR=setR[i]-adcR;
                        if( (thL<tolL[i]) && (thR<tolL[i]) )
                        {
```

```
                              move_to(0,0);
                              display_int(adcL);
                              display_char(key[i]);
                              display_int(adcR);
                              if((PINB&0x04)==0x00) transmitByte(key[i]);
                              else transmitByte('Z');
                        }
                  }
                  move_to(0,1);
                  if(i==17)
                  {
                        for(j=0;j<sizeof(err)-1;j++) display_char(err[j]);
                        _delay_ms(200);
                  }
                  _delay_ms(100);
                  for(j=0;j<sizeof(pre)-1;j++) display_char(pre[j]);
                  _delay_ms(100);
            }
      }
      return 0;
}
```