EE318 – Electronic Design Lab, Project Report, EE Dept, IIT Bombay, April'09

Inertial Stepper

Group - B12

Vasishta Atmuri (06007031)
Harish Reddy (06007032)
Naumaan Nayyar (06007035)

Guide: Prof. Dipankar Sarkar

1. Abstract

An inertial stepper is an electro-mechanical system capable of moving small objects over tiny distances, even to the order of micrometers, with a great accuracy. With good design, precision in sub-micrometer range is possible. Such devices can be used in a variety of applications such as scanning probe microscopes, for measuring material properties, in biomedical applications etc. We have constructed an inertial stepper with a piezotube. Piezomaterials expand and contract when voltage is applied to their terminals. This property is used by us in making a small object move forward or backward on a stand. Due to the exploratory nature of this project, most of the observations are in the form of experimental data obtained from the setup.

2. Introduction

The whole mechanical setup is as shown in the figure below. The piezotube is mounted on graphite rods and connected to the electrical system. Due to the low strain coefficient (measure of relative change in dimension with applied electric field) of the piezotube, high voltages are required to obtain noticeable results. For this reason, we have constructed a boost converter to step up voltages to required levels and converted the stepped up voltage to a saw-tooth waveform to be applied to the piezotube. More on this will be discussed in principles in a later section.

An interface is provided to input the distance to be moved by the object. However, due to the differences in frictional forces on various objects, the speed of the object has to be recalibrated each time the object is changed. Also, due to the differences in the smoothness of different parts of the glass stand we are using, the object may move with different speeds over different areas. All these are detailed in the observations section.

Due to shortage of time, part of the electronics stage had to left on the breadboard and could not be transferred to a PCB.

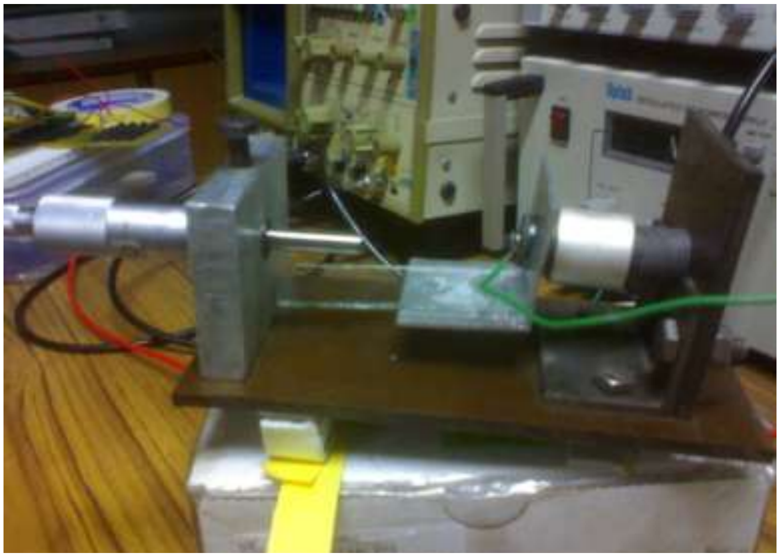
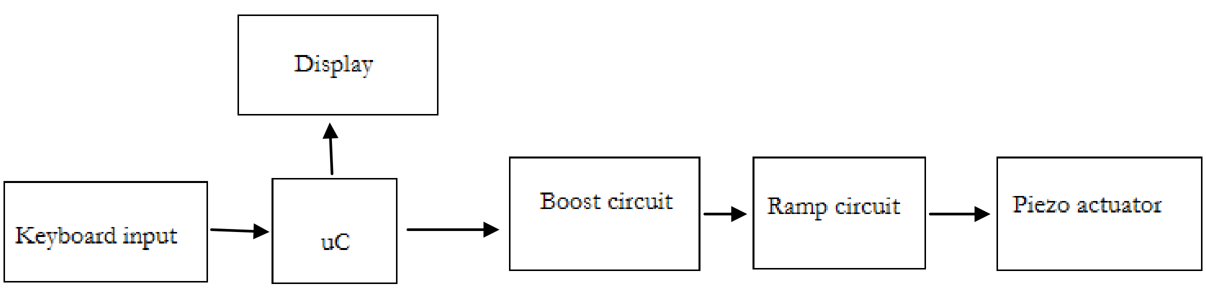


Figure 1: The Inertial Stepper

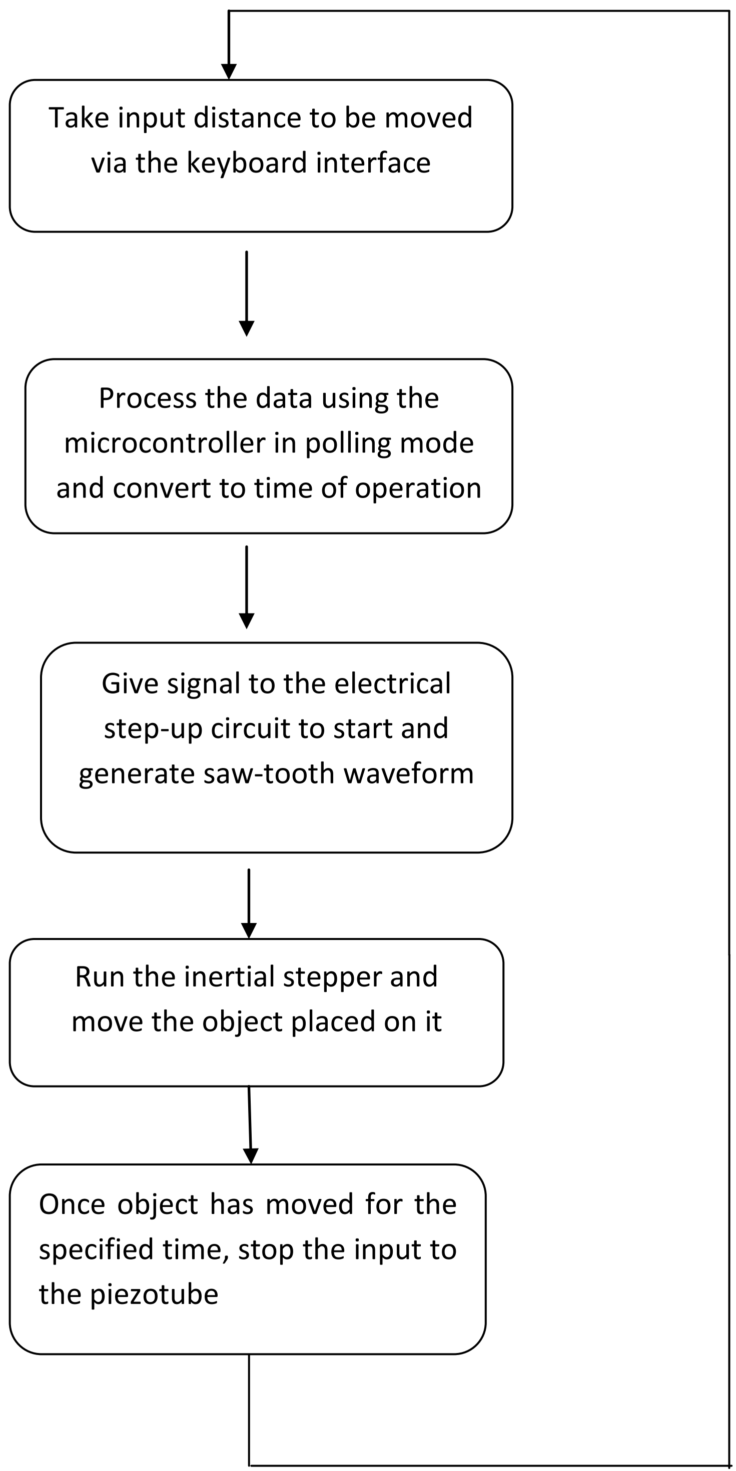
3. System Design

Hardware Description

At a high level, the system consists of three subsystems: the mechanical system which consists of the piezotube and its mount, the electrical system which consists of the high voltage step=up and ramp circuit, and the electronics system which consists of the control circuitry and the input interface. The block diagram of the system is as follows:



The logical flow of direction of the system is given in the figure below:



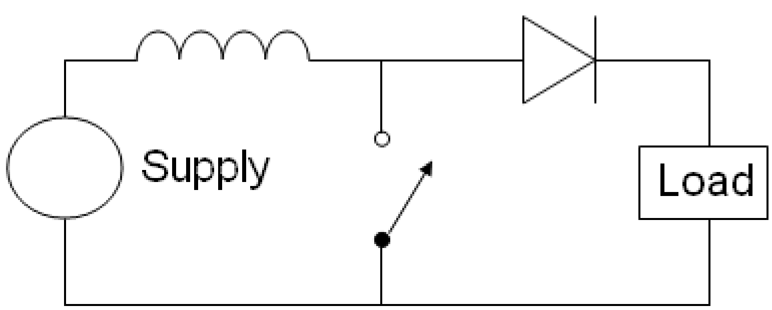
1. Mechanical subsystem

The mechanical subsystem consists of a piezotube which is mounted on both sides with graphite rods (chosen because they are conducting materials). Connections are made to the terminals, at the inside and the outside of the piezotube. A glass stand is attached to the graphite rod at one end on which the object to be moved is placed. The base of the system is made out of Bakelite, a lightweight and rigid polymer. A micrometer head is mounted on another fixture at the other end of the board, which can be retracted and protracted to measure the distance moved by the object. The working of the mechanical system is detailed in the design procedure section. Due to a slight tilt in the fixture which holds the glass slab, the mechanical base had to be tilted slightly in the opposite direction.

2. Electrical subsystem

The electrical subsystem consists of a boost circuit which is used to step up the input DC voltage to a high enough voltage to apply to a piezotube. This is followed by a ramp circuit which converts this stepped up DC voltage to a saw-tooth waveform. This is essential because the piezoelectric tube requires high voltages to show appreciable change in its dimensions.

A basic schematic of the boost converter is as shown below:



The output of the boost converter is fed into a ramp generating circuit made with Resistor-Capacitor networks and power MOSFETs.

Components: Apart from standard resistors and capacitors, power devices such as MOSFETS, BJTs and diodes are used. 555 timer ICs are used for the switching purposes. An inductor of high relatively high current rating is also used as part of the circuit.

555 timer IC

The 555 timer IC is a versatile analog chip which is used to generate signal pulses. We have used the 555 in astable mode to provide the switching waveform for the power MOSFETs in the boost circuit and in the ramp circuit.

Power devices

Power devices such as power MOSFETs, BJT and diode have been used in the high voltage parts of the circuit. The MOSFETs are used for switching in the circuits and the power BJT is used to provide feedback to the 555 to ensure that the voltage at the output of the boost converter does not increase without bound.

3. Electronics Subsystem

As outlined in the block diagram above, the electronic system is used to control the working of the whole system. A keyboard interface is provided for the user to input the distance to be moved by the object. This is taken as input into the microcontroller by polling the port to which the keyboard is connected.

The range in which the distance lies is obtained from the input and the time is appropriately adjusted to the best possible value and a signal is generated to be transmitted to the electrical system to start/stop the saw-tooth waveform generation.

Components: Apart from the microcontroller which is described below, the other components in the electronics system include a 16x2 LCD screen with its driver included in the package and a 4x4 keyboard interface.

4x4 Keyboard

The keyboard is polled by pulling up half the output pins and polling the other half sequentially for a pressed key. The debouncing is done in the software itself. Each key corresponds to a specific port input which the microcontroller sees.

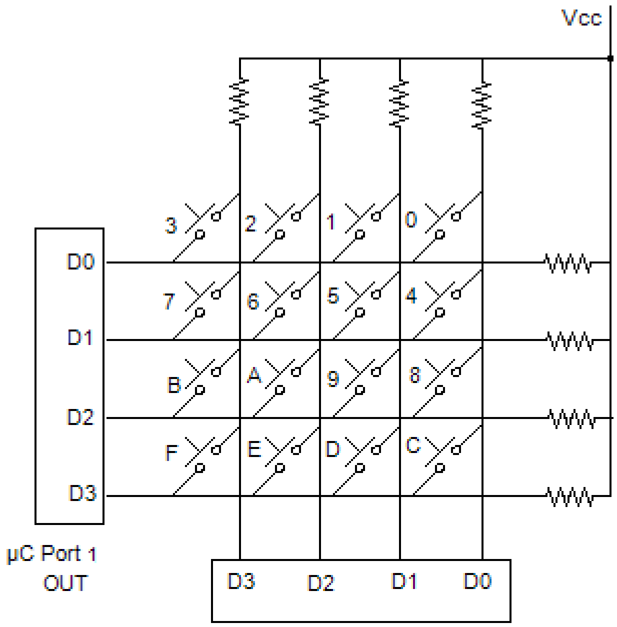


Fig.1

Atmega16

ATMEL's Atmega16 is a low-power, high-performance 8-bit Microcontroller. It has the RISC architecture. It has four ports which can be used as bidirectional input/output as well as other alternative features.

We have used the Atmega16 for the following reasons. It has an internal clocking option by using which we wished to simplify the circuit. It had the desired number of pins we needed for input/output operations. Its pins were conveniently placed which made our external wire connections simpler and allowed us to focus more on our project.

Software

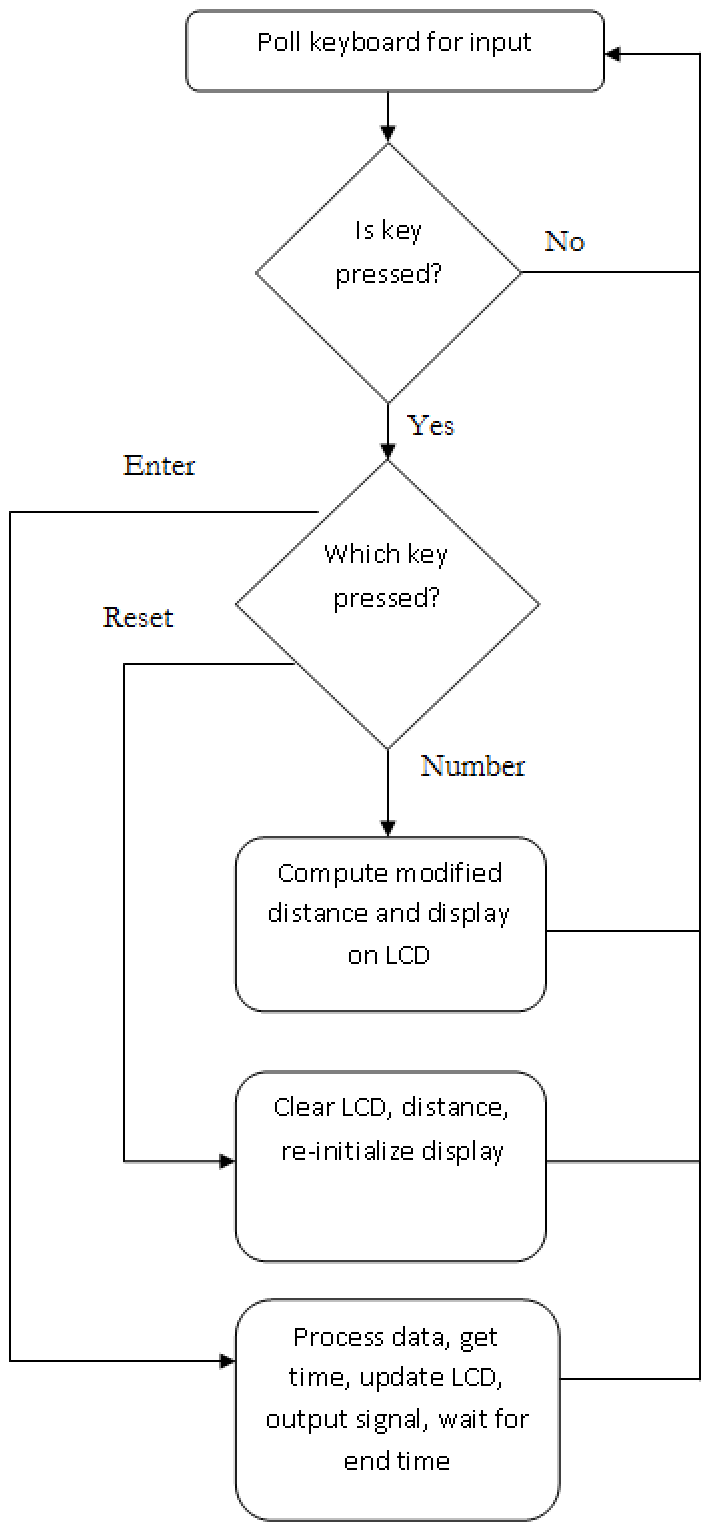
The actual software code is provided in Appendix B. The functioning of the code is detailed in the following:

The input to the microcontroller is a 4x4 keyboard interface, which is connected to a port of the Atmega16. Another port of the microcontroller is connected to the LCD with its control pins also connected. The pins of the keyboard port are continuously polled and the data passes through two stages. First, it is converted into distance at runtime and next, it is given to the LCD to display the key pressed by the user. Once the user finishes his input, indicated by pressing the key marked "E", the range in which the input falls is determined. This is essential because of the limitations in the size of data that can be processed. The "Int" data type is only 2 bytes in size, giving a maximum value of 65535 to its contents.

We observed that the range in which the inertial stepper was to function could not be captured in one single equation due to restrictions on size of “Int” on one hand and on the result of division being an integer on the other. So, we split it into cases and accordingly scaled the time to cover the whole range.

Once the time for which the setup was to run was calculated, an output signal was given high to control the start and stop of the inertial stepper.

A flow diagram of the code is given on the next page.


Poll keyboard for input
Is key pressed?
No
Yes
Enter
Which key pressed?
Reset
Number
Compute modified distance and display on LCD
Clear LCD, distance, re-initialize display
Process data, get time, update LCD, output signal, wait for end time

4. Circuit Diagrams and Principles

Boost circuit with ramp generation: The circuit diagram of the boost converter circuit is:

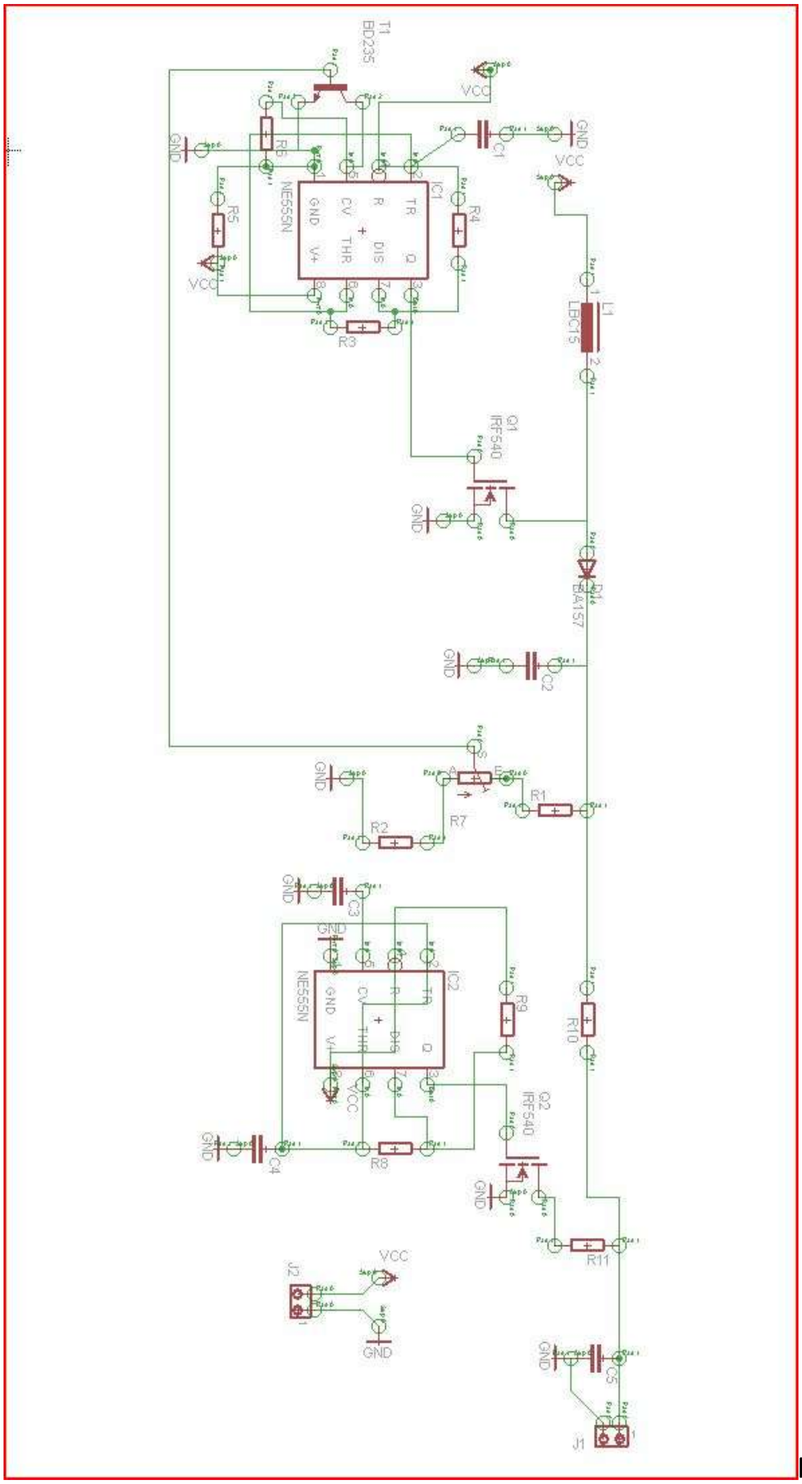


Figure 2: Boost converter with ramp generation

The functioning of the boost circuit is based on the storage and transfer of energy from the inductor. The diode prevents the energy from leaking back into the circuit. On each switching of the MOSFET, the inductor gets charged in the ON cycle and transfers its energy to the capacitor in the OFF cycle. The switching of the MOSFET is done using a 555 timer. A power BJT is connected as a feedback to the control pin of the 555 to control the output voltage of the boost converter.

The output of the boost converter is fed into the Resistor-Capacitor network, with a large charging time path and a small time discharging path through another power MOSFET which is switched with a function generator.

5. Test Procedure

Test procedure to show how the design achieves the requirement of the problem:

The mechanical design is shown in the figure below.

It has a Piezo tube attached at both the sides with graphite, one of them goes to an aluminum " L " clamp, to which the surface on which the objects have to be moved is attached. The other side goes to a fixed plate. To the left side of the set up we have a micrometer, with an accuracy of up to 10 micrometers, which can be used to calibrate the set up for various materials at varying ranges of frequency and voltages.

The Piezo tube has the following characteristics:

Outer Diameter: 22.0 ± 0.15 mm
Inner Diameter: 16.0 ± 0.20 mm
Length: 20.0 ± 0.15 mm
Capacitance at 1KHz: 6300 pF
Resonance Frequency: 82 KHz
Strain Coefficient in direction of polarization (d33): 350 pC/N
Strain Coefficient in direction normal to polarization (d31): 175 pC/N

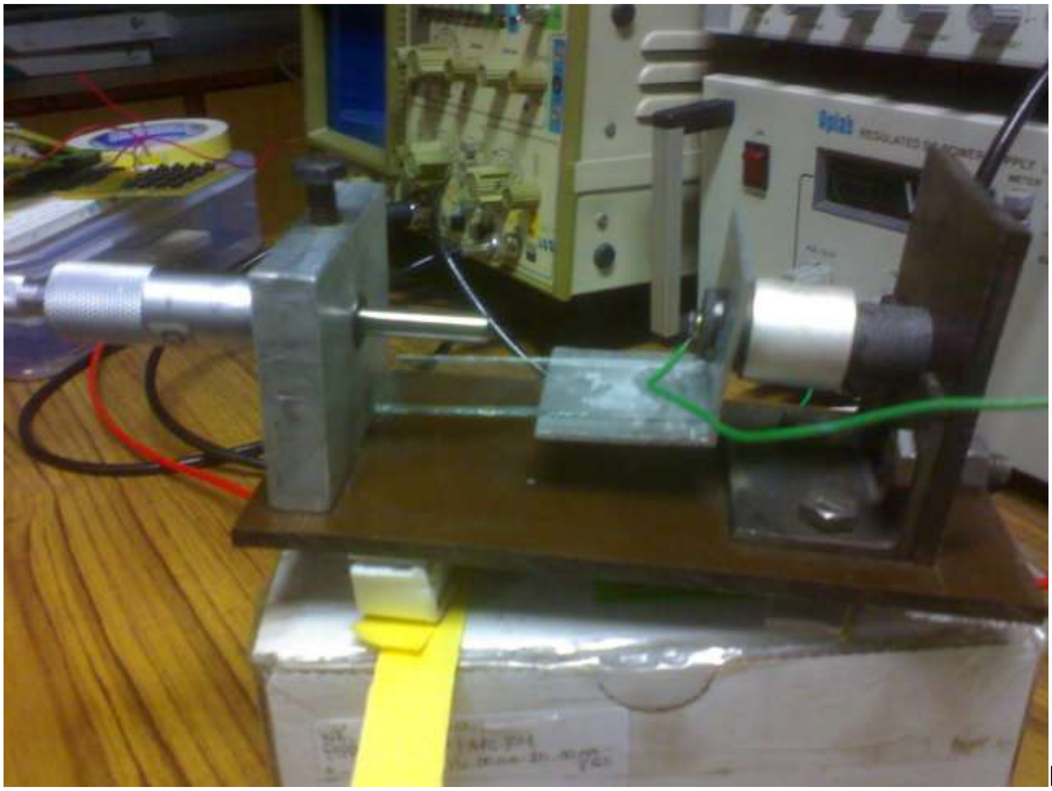


Figure 3: Mechanical System of the Inertial Stepper

So a ramp voltage of peak approximately equal to 200 V is given to the piezo tube. The object to be moved with micrometer range precision is put on the glass slab, attached to the piezo through a clamp. This ramp voltage would produce vibrations in a Piezo tube and since the voltage is a ramp as shown in the figure, the piezo crystal would expand slowly and contract at a much higher pace, hence the object moves along with the piezo tube while expanding but while contracting, it stays there due to inertia.

The object used here for the experiment is a potentiometer, with a tube fixed as a reference point to measure the distance moved.

The object to be moved is placed on the glass slab and the distance to be moved is input to the microcontroller, which in turn would turn on the circuit to the piezo tube for an apt amount of time, hence moving the object to the required distance.

6. Observations and Results

Note: These will be filled with hand as we are recalibrating the device and these readings will be updated and uploaded after the recalibration has been done.

Test results, plots, tables

a. Frequency: 154.6Hz; Voltage (Boost Voltage): 247V

Time (sec) Distance (mm)

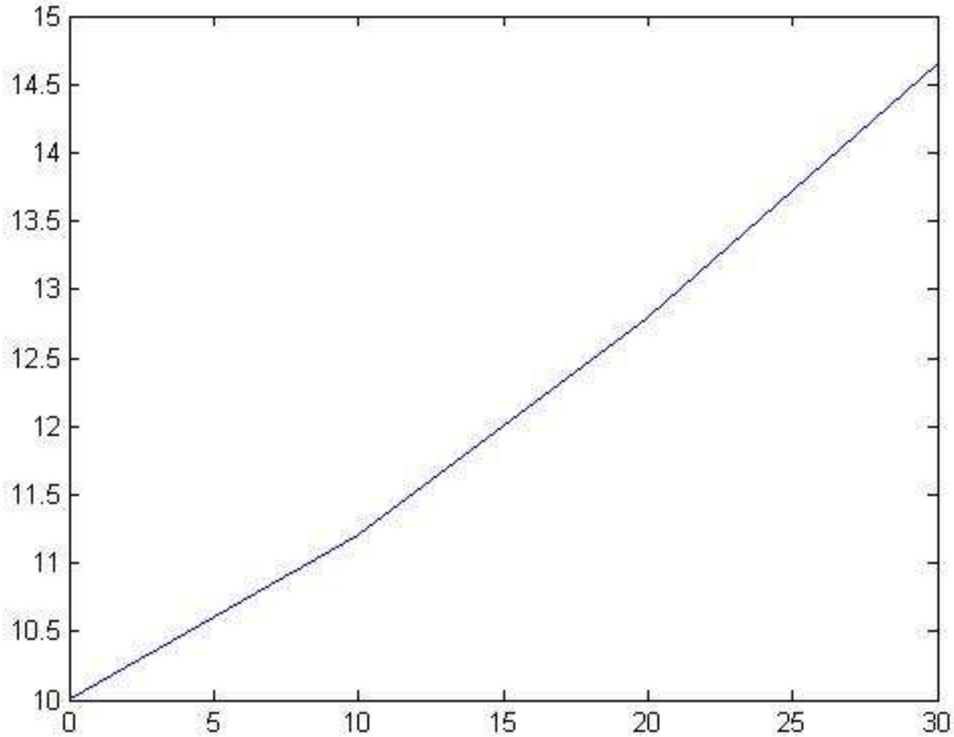
0	0
30	2.5
60	4
90	5.5
120	6.5
150	8.5
180	10.5
210	16.45

Observation: From this and other readings, we observed that the glass surface has a different coefficient of friction after a distance of 10mm. Based on this, we have split our readings into two, before 10mm and after. All the following readings will be of this form.

b. Frequency: 154.6Hz; Voltage (Boost Voltage): 247V

Time (sec) Distance (mm)

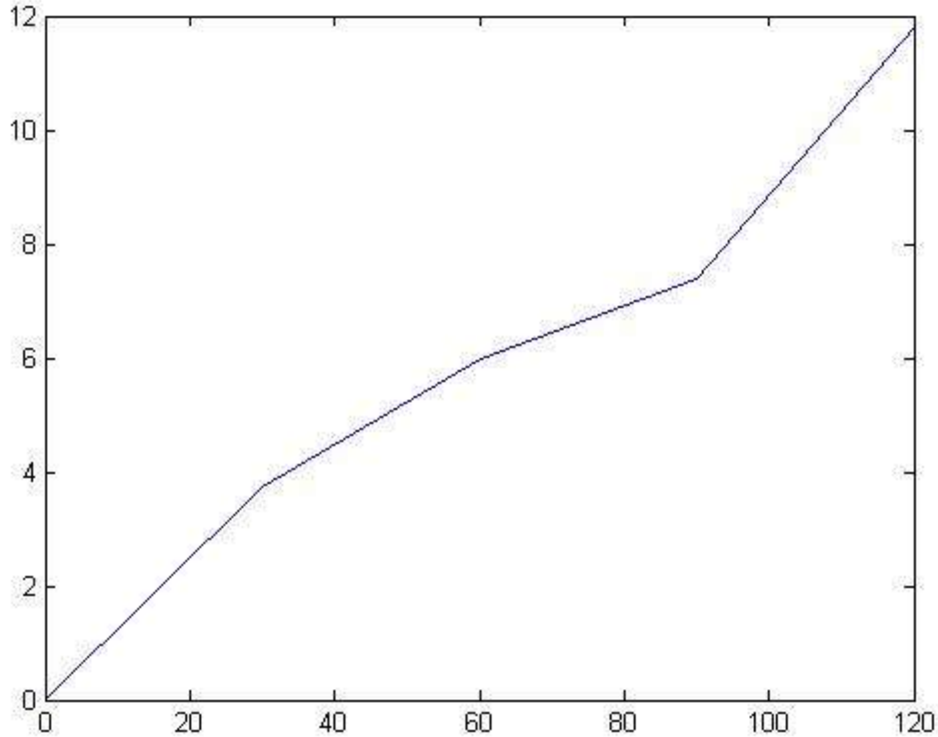
0	10
10	11.45
20	12.06
30	12.5
45	14.5



c. Frequency: 200.1Hz; Voltage (Boost Voltage): 246.6V

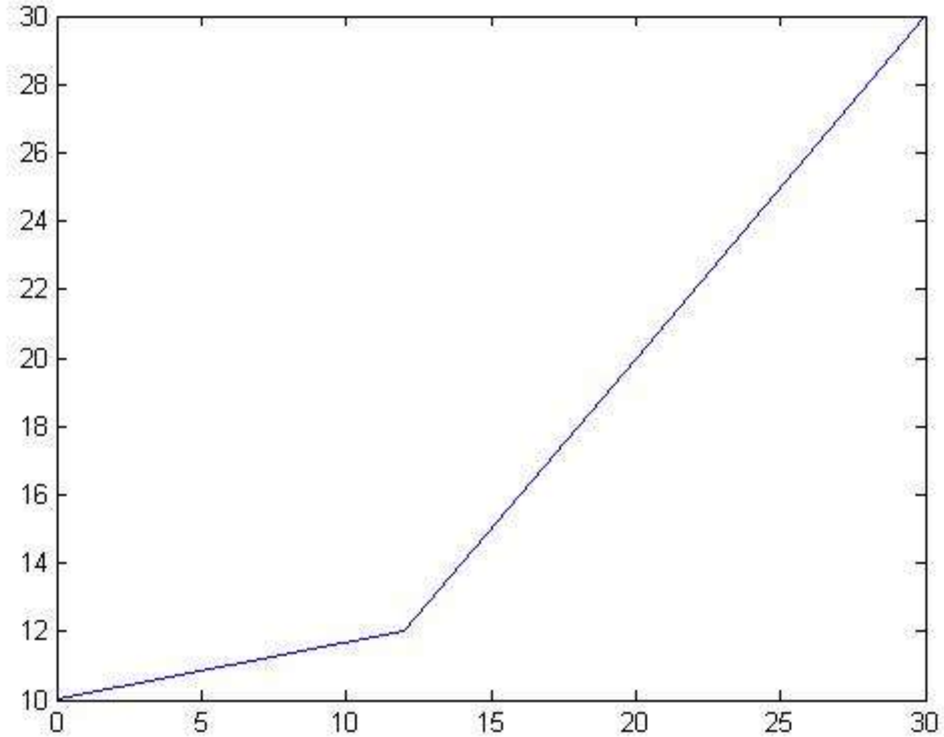
Time (sec) Distance (mm)

0	0
30	3.75
60	6.00
90	7.38
120	11.8



d. Frequency: 200.1 Hz; Voltage (Boost Voltage): 246.6V

Time (sec)	Distance (mm)
0	10
10	11
20	16
30	21



Discussion of the results:

The plots t vs d gives the relation between time and distance moved by the object, but the graph is not linear as expected, this may be because of the uneven forces on the object at different places on the glass slab. The object is also observed not to move exactly straight, this could also be due to the uneven forces at each point on the slab and also uneven forces on the three legs of the object.

However, we can get an approximate relation between the time and distance moved by an object at a given frequency and voltage and the same is used by the microcontroller to move the object, by turning it on for a given amount of time for a required distance.

We can also observe that if we reverse the polarity of the applied voltage, the object would move backwards, this is because the piezo now contracts slowly and expands very fast, which is opposite to the earlier case in which the expansion was slow and the contraction was very fast.

7. Conclusion and Further Improvements.

Suggestions:

The system needs to be made with ultra light objects and a piezo tube of higher d33 would help. The objects to be moved should be have a constant coefficient of friction on its entire surface which touches the slab and the same applies for the slab too.

The position of the object has to be directed, so a groove kind of a structure would make it better. This system can be modified to use as a scanning probe microscope too.

Making the mechanical system more precise would result in greater accuracy of the measurements than we have observed in these experiments.

The interface can be made more compact and neat to result in a finished product. Due to lack of time, the looks of our system are not very well designed.

8. Acknowledgements

We have learnt a lot in this project. We learned of the importance of integration of various types of systems and the dependence of electrical systems and mechanical systems on each other. Although we faced a lot of difficulties to start with, especially with the connection of the two types of systems, in the end, we have a reasonable setup of the Inertial Stepper with which we have moved objects over distances of upto 1.5 cm with a decent accuracy.

We sincerely wish to thank Professor Dipankar for his excellent guidance throughout our project and benefited a lot from his expertise. His organized approach which he asked us to follow in our project as well greatly helped us and also gave an insight into how such projects are done in the real world. We also wish to thank our mentor, Nipun Dave, for his help with the project.

9. References

1. Piezotube characteristics - http://www.piceramic.de/site/piezotubes.html
2. Piezoelectric properties - http://en.wikipedia.org/wiki/Electrostriction
3. Keyboard interfacing - http://engknowledge.com/microcontroller_interfacing_keyboard.aspx
4. ATmega16 datasheet
5. High voltage generation - http://www.dos4ever.com/flyback/flyback.html

Appendix

Appendix A

User’s Manual:

Contents in the project:

1) Mechanical system
2) Keyboard Input
3) LCD display

Description of Components:

Mechanical System: The mechanical system includes-

- Piezoelectric tube as an actuator, responsible for the motion of the object.
- Bakelite board and mounts.
- Micrometer.

The object to be moved is placed on the glass plate provided. The micrometer is used to measure the distance moved by the object.

Caution: The piezoelectric tube and the components connected to it should not be touched at any point of time during the experiment due to high voltages in these parts of the system.

Keyboard Input: The user can provide the distance the object has to be moved using a keyboard interface provided, which contains the numbers 0-9 and keys marked ‘E’ for “Enter” and ‘R’ for “Reset”.

LCD display: The LCD is used to display the input given by the user so that he has a comfortable usage and also to indicate the status and completion of the experiment.

Initialization:

1. Connect the voltage sources as indicated. The 12V D.C supply is connected to the High Voltage Ramp Circuit as indicated on the board. Make sure not to touch the board once the supply is connected.
2. Connect the output of the board to the piezoelectric tube as follows:
   Forward motion:
   Connect the wire (already attached) emanating from the graphite cylinder (connected to inner side of the piezoelectric tube) to the ground. Connect the wire attached on the outer side of the piezoelectric tube to the capacitor pin.
   Reverse motion:
   Connect the wire (already attached) emanating from the graphite cylinder (connected to inner side of the piezoelectric tube) to the ground. Connect the wire attached on the outer side of the piezoelectric tube to the resistor pin.
3. Place the object at a desired position on the glass plate.
4. Note down the micrometer reading at the position of the object.
5. Switch on the power supply.

Experiment:

1. Enter the desired distance to be moved in the keyboard in units of 100's of μm and press the enter key 'E'.
   Note: The object is designed to move in the range of $100\mu m$ -$1.5cm$. Enter the values in this range only.
2. Press the reset 'R' button if the distance entered is erroneous.
3. The LCD displays "Processing..." once the input is entered and the inertial stepper is functioning.
4. The LCD displays "Done!" once the object has moved the required distance.
5. Note down the reading of the micrometer with the object at this position. Check if the distance moved is same as the one entered as input.

The experiment is now complete.

Appendix B

Source code for keyboard and LCD interface:

```
#include<avr/io.h>
#include<util/delay.h>

#define sbi(x,y) x |= _BV(y) //set bit - using bitwise OR
operator
#define cbi(x,y) x &= ~(_BV(y)) //clear bit - using bitwise AND
operator
#define tbi(x,y) x ^= _BV(y) //toggle bit - using bitwise XOR
operator
#define is_high(x,y) (x & _BV(y) == _BV(y))
 //check if the y'th bit of register 'x' is high ... test if its
AND //with 1 is 1

#define data PORTC //change in wait_lcd too
#define RS PD5
#define RW PD6
#define EN PD7
#define SG PD0

#define speed 66//speed with which object moves, in um/s


int distance=0;
//distance goes from 100um to 1.5cm, input in 100um units

unsigned int flag;

void wait_lcd(void){
     flag=0xff;
     while(flag){
          cbi(PORTD,EN);
          cbi(PORTD,RS);
          sbi(PORTD,RW);
          DDRC=0x00;
          data=0x00;
          sbi(PORTD,EN);
          sbi(PORTD,EN);
          flag=PINC;
          flag=PINC;
          flag&=0x80;
     }
     DDRC=0xff;
     cbi(PORTD,EN);
     cbi(PORTD,RW);
}

void init_lcd()
```

```
{
     cbi(PORTD,RW);
     cbi(PORTD,RS);
     data=0x38;
     sbi(PORTD,EN);
     sbi(PORTD,EN);
     cbi(PORTD,EN);
     wait_lcd();
     cbi(PORTA,RS);
     data=0x0e;
     sbi(PORTD,EN);
     sbi(PORTD,EN);
     cbi(PORTD,EN);
     wait_lcd();
     cbi(PORTD,RS);
     data=0x04;
     sbi(PORTD,EN);
     sbi(PORTD,EN);
     cbi(PORTD,EN);
     wait_lcd();
}

void write_lcd(unsigned char val){
     sbi(PORTD,RS);
     data=val;
     sbi(PORTD,EN);
     sbi(PORTD,EN);
     cbi(PORTD,EN);
     wait_lcd();
}

void set_cursor(unsigned int pos){
     cbi(PORTD,RS);
     data=pos;
     sbi(PORTD,EN);
     sbi(PORTD,EN);
     cbi(PORTD,EN);
     wait_lcd();
}

void clear_lcd()
{
     cbi(PORTD,RS);
     data=0x01;
     sbi(PORTD,EN);
     sbi(PORTD,EN);
     cbi(PORTD,EN);
```

```
    wait_lcd();
    set_cursor(0x80);
}


void lcdata(unsigned char value)
{
    int val;
    val = value-48;
    if(distance==0) clear_lcd();
    write_lcd(value);
    distance=distance*10 + val;
}

void reset()
{
    unsigned char input[17] = "Distancex100um:";
    distance = 0;
    clear_lcd();
    for(int i=0;i<16;i++)
        write_lcd(input[i]);
    set_cursor(0x80);
}

void process()
{
    unsigned char sent[14] = "Processing...";
    unsigned char done[6] = "Done!";
    int num,num1;
    unsigned int time,time1;
    int i;
    if(distance<10) num=1;//multiply by 1000
    else if(distance<100) num=10;//multiply by 100
    else if(distance<1000) num=100;//multiply by 10
    else num=1000;
    num1=1000/num;
    time = distance*num1;
    time=time/speed;
    time=time*100;
    clear_lcd();
    sbi(PORTD,SG);
    time1=time;
    for(i=0;i<13;i++)
        write_lcd(sent[i]);
    for(i=0;i<num;i++)
    {
    for(int j=0;j<time;j++)
```

```
          _delay_ms(1);
     }
     cbi(PORTD,SG);
     clear_lcd();
     for(i=0;i<5;i++)
          write_lcd(done[i]);
     _delay_ms(2000);
     reset();
}

int main()
{
          int i;
          char keyCode,upperNibble;
          unsigned char keyPressed;
          DDRB = 0x0f; //Key-board port, higer nibble - input,
lower nibble - output
          PORTB = 0xff; //pull-up enabled for higher nibble
          DDRC = 0xff; //lcd
          DDRD = 0xff; //control
        upperNibble = 0xff;
          init_lcd();
          clear_lcd();
          reset();
while(1)
{
     for(i=0; i<4; i++)
     {
     _delay_ms(1);
     PORTB = ~(0x01 << i);
     _delay_ms(1); //delay for port o/p settling
     upperNibble = PINB | 0x0f;

     if (upperNibble != 0xff)
     {
     _delay_ms(20); //key debouncing delay
     upperNibble = PINB | 0x0f;
     if(upperNibble == 0xff) continue;

     keyCode = (upperNibble & 0xf0) | (0x0f & ~(0x01 << i));

     while (upperNibble != 0xff)
     upperNibble = PINB | 0x0f;

     _delay_ms(20); //key debouncing delay
```

```
    switch (keyCode) //generating key characetr to display on
LCD
    {
    case (0xee): keyPressed = '0';
         lcdata(keyPressed);
         break;
    case (0xed): keyPressed = '1';
         lcdata(keyPressed);
         break;
    case (0xeb): keyPressed = '2';
         lcdata(keyPressed);
         break;
    case (0xe7): keyPressed = '3';
         lcdata(keyPressed);
         break;
    case (0xde): keyPressed = '4';
         lcdata(keyPressed);
         break;
    case (0xdd): keyPressed = '5';
         lcdata(keyPressed);
         break;
    case (0xdb): keyPressed = '6';
         lcdata(keyPressed);
         break;
    case (0xd7): keyPressed = '7';
         lcdata(keyPressed);
         break;
    case (0xbe): keyPressed = '8';
         lcdata(keyPressed);
         break;
    case (0xbd): keyPressed = '9';
         lcdata(keyPressed);
         break;
    case (0xbb): keyPressed = 'E';
         process();
         break;
    case (0xb7): keyPressed = 'R';
         reset();
         break;
    case (0x7e): keyPressed = 'C';
         break;
    case (0x7d): keyPressed = 'D';
         break;
    case (0x7b): keyPressed = 'E';
         break;
    case (0x77): keyPressed = 'F';
         break;
```

```
      default : keyPressed = 'X';
      }//end switch
      }//end if
      }
}
}
```