

Voice Controlled Transcriber

ESD Project

Assignment -3

Guided By: Prof. Anurag Lakhani

Group Members

Maitrey Mehta	1401040
---------------	---------

Jay Shah	1401053
----------	---------

Parth Shah	1401054
------------	---------

Bhavya Patwa	1401063
--------------	---------



**AHMEDABAD UNIVERSITY**

## **Index**

Introduction.....	3
Project Background.....	3
Market Survey.....	3
Block Diagram.....	4
Selection Criteria.....	5
Components.....	5
Costing Table.....	6
Circuit Diagram.....	7
Major Troubleshooting.....	7
Snapshots.....	8
Flow Chart.....	11
Code.....	13
Conclusions.....	37
Future Work.....	37
Project Timeline.....	37
References.....	38

## **Introduction**

Voice controlled transcriber is a unique device that converts speech of user to physical text on paper.

Speech to text in digital module i.e. in a PC or mobile device, are already available. They work on a very simple concept of converting speech into text or any particular command is to be done.

Similarly a simple transcriber i.e. if user presses certain buttons certain text is printed or written on a paper.

Our device aims at combining both of these features creating a unique transcriber that can actually write what user has commanded.

## **Project Background**

Microcontroller is the heart of the device, it connects all the necessary components like stepper motor, Bluetooth device etc.

A computing device (PC or mobile) sends speech to the microcontroller via near field communication (NFC) device in our case Bluetooth device (Wi-Fi device can also be used).

This speech is then converted to text and displayed on LCD screen.

Stepper motors takes command as this speech and functions accordingly to give a physical textual output.

## **Market Survey**

This device has very huge potential in judicial courts. If successfully implemented this device can write every proceedings of the court without human help.

Also this device is extremely useful for blinds and physically challenged people who cannot write they can just command via speech and physical text can be written for them.

Also these device can act as to-do list jotter which jot's down every task to be done and we don't require a pen and paper every time just *say the words and get them written.*

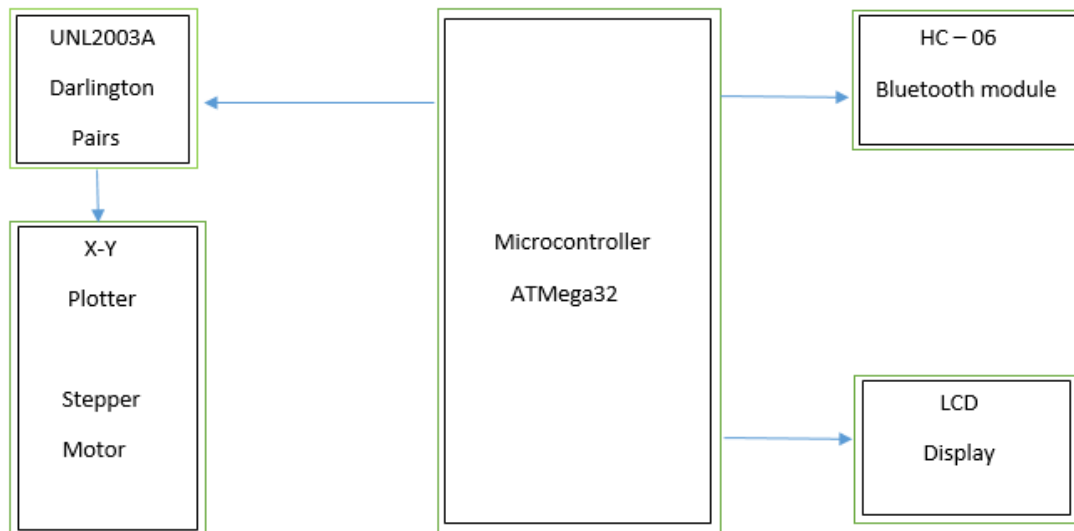
No need to worry about a person who types for you, if you got this just *say the words and get them written.*

Thus these product has a huge potential and can replace humans in many tasks.

## **Block Diagram**

The following is the block diagram of our transcriber it contains

- Micro controller
- Stepper Motors
- Darlington Pairs
- Bluetooth module



## **Selection Criteria**

### **➤ Meeting Computational Needs**

- ✓ As we are mainly concerned about the correct detection and transmission of data.
- ✓ We do not require very high memory.
- ✓ Moreover all the mechanical device we are using can easily work at 5-12V so power requirement is also not high.
- ✓ The Bluetooth device which we are using is having a decent range, which is enough for us.
- ✓ Moreover, stepper motors Darlington pairs and actuators which we are using are cost efficient and also meet our requirement.

### **➤ Availability of Software Development Tools**

- ✓ Our microcontroller should be such that it interfaces with all the other components we are using.
- ✓ Also the components like stepper motors, LCD screen, and actuators have to be from a nearby vendor and thus providing us technical help if needed.

## **Components**

**Atmega32:** This microcontroller undoubtedly meets our all selection criteria, having capacity to withhold required power supply, enough memory (32K), and a decent number of I/O pins for our project.

**HC-05:** A simple Bluetooth device having a decent range, can connect to a single device at a time i.e. either can receive or transmit but cannot do both simultaneously, which is fine for us as we use Bluetooth only to receive the command (speech) from the user.

**Uln2003A:** 7 Darlington pairs helping as a perfect intermediate for our stepper motor and Atmega32, as we are using stepper motors having only 4 coils this fits perfectly. Moreover Uln2003A comes in PDIP, SOIC, SOP and TSSOP packaging as we are using Atmega32 (PDIP) socket our need is satisfied.

**Stepper Motor:** A stepper motor working at 12V DC supply and 7.5 degree.

**Batteries:** 12V 1A external battery for stepper motors, this battery acts as voltage source.

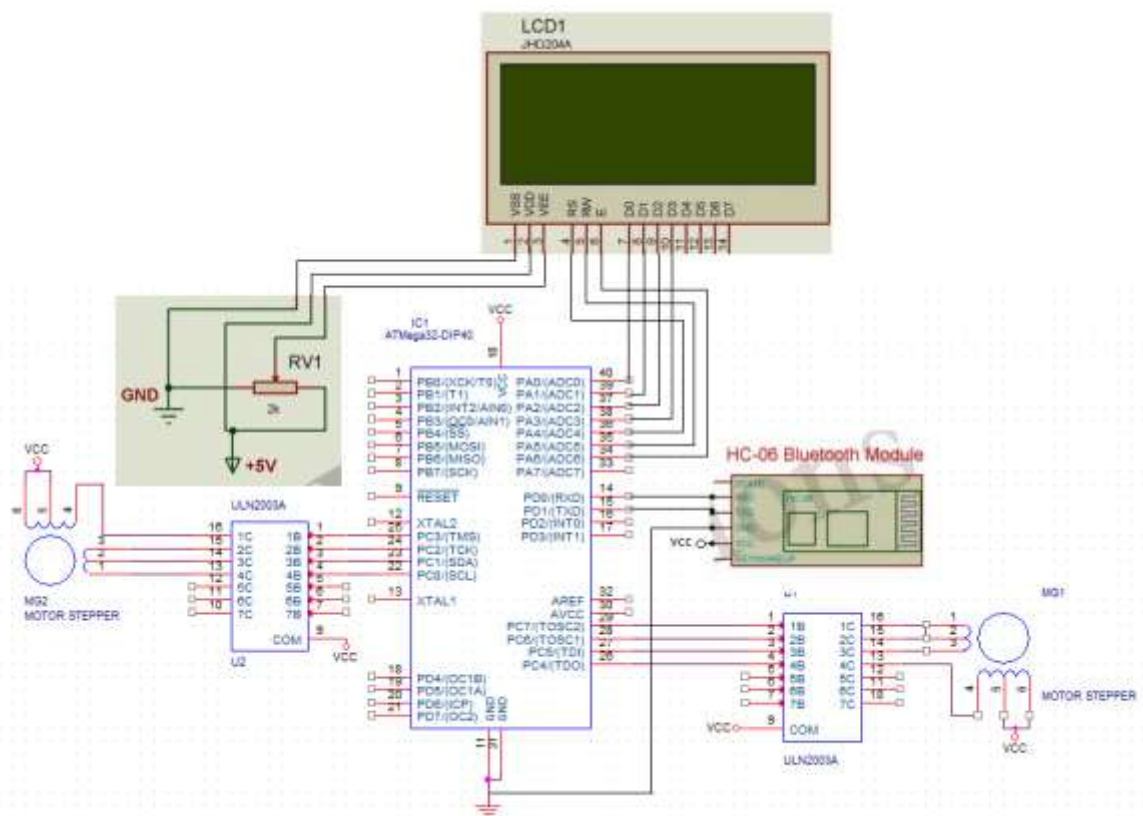
**LCD Board:** LCD Board of (16X2) for a desirable output to display.

The above information about the components is taken from [1], [2], [3].

### **Costing Table**

Components	Cost (in Rs)	Source
ATmega 32	500	College
Stepper Motor and motor Drivers	500 (250 each motor) 30 (15 each driver)	Shantinath Stores(Nr ManekBaug BRTS)
Bluetooth Device	300	Shantinath Stores(Nr ManekBaug BRTS)
LCD Screen	160	Delta electronics(Nr Shivranjani BRTS)
Battery, Wires and other electronic items	300 (battery) 100(wires and other items)	Lucky electronics(Relief Road)
Bread boards and Circuit Boards	100 (Circuit board) 100 (25 each bread board)	Delta electronics(Nr Shivranjani BRTS)
Other Miscellaneous expenses	60	
<b>TOTAL</b>	<b><u>2150</u></b>	

## Circuit Diagram



Instead of HC-06 HC-05 has been used.

The above diagram has partial components from [1], [4].

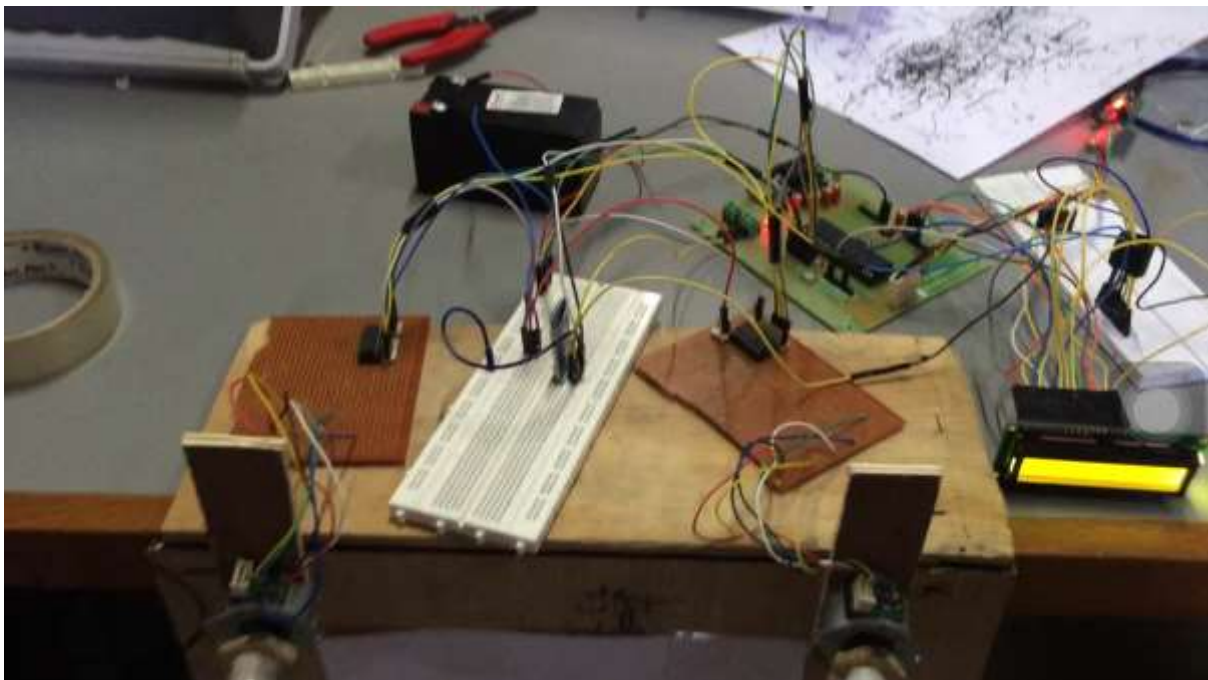
## Major Troubleshooting

- The only problem we faced during the whole process was to design the mechanical model of the transcriber.

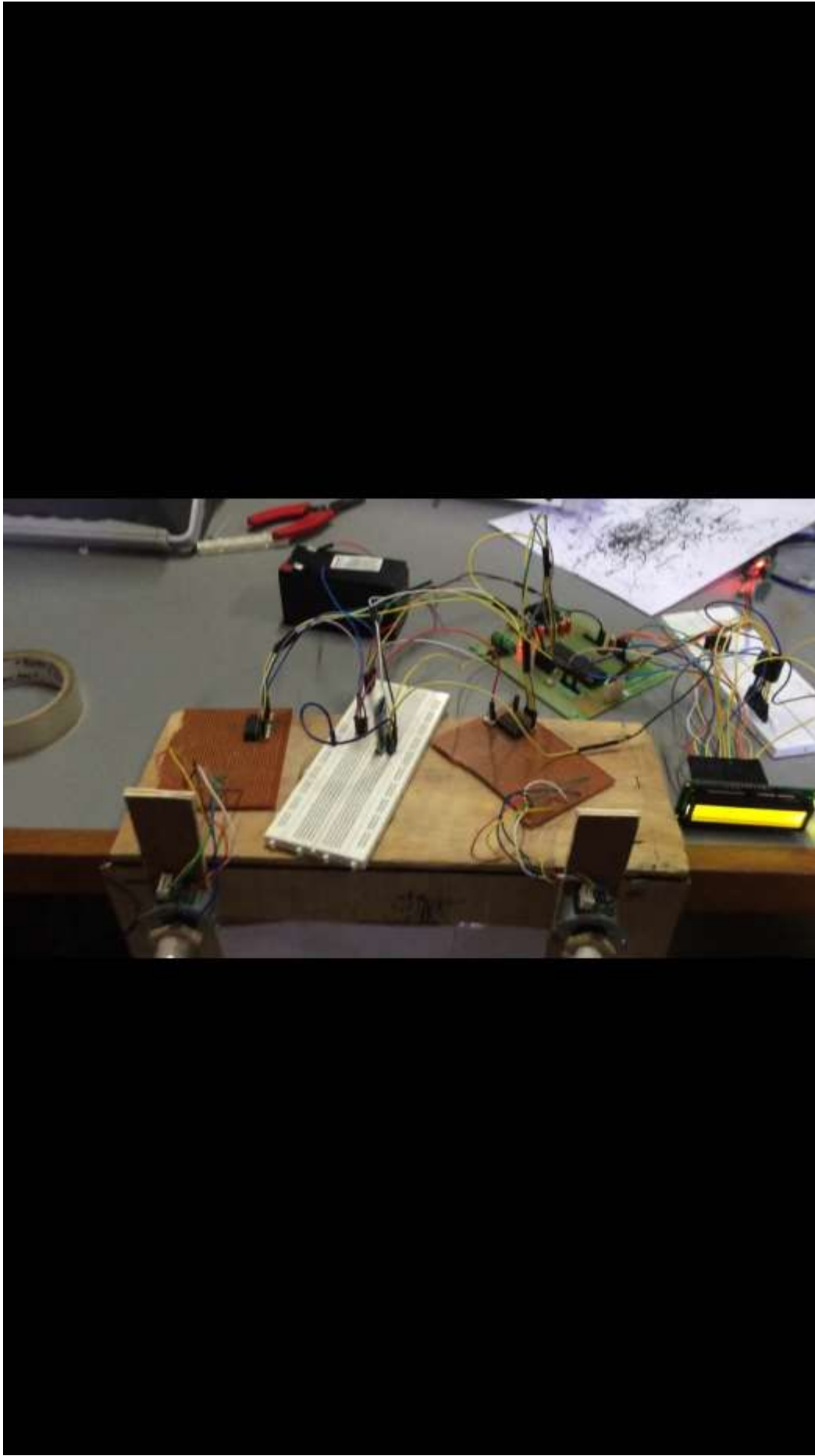
- As the writing of any letter requires a rigour and stable mechanical model, initially writing of the letters was not possible but we then designed a stable model using wooden planks which helped us to solve this problem up to a certain extent.
- Writing of letters is not at its best, it's just satisfactory this is only because we couldn't develop a stable mechanical model.
- All other modules were successfully implemented without much effort, improper working of any module and errors occurring in code due to some minor mistakes were faced but they were identified with little or sometimes somewhat more debugging.

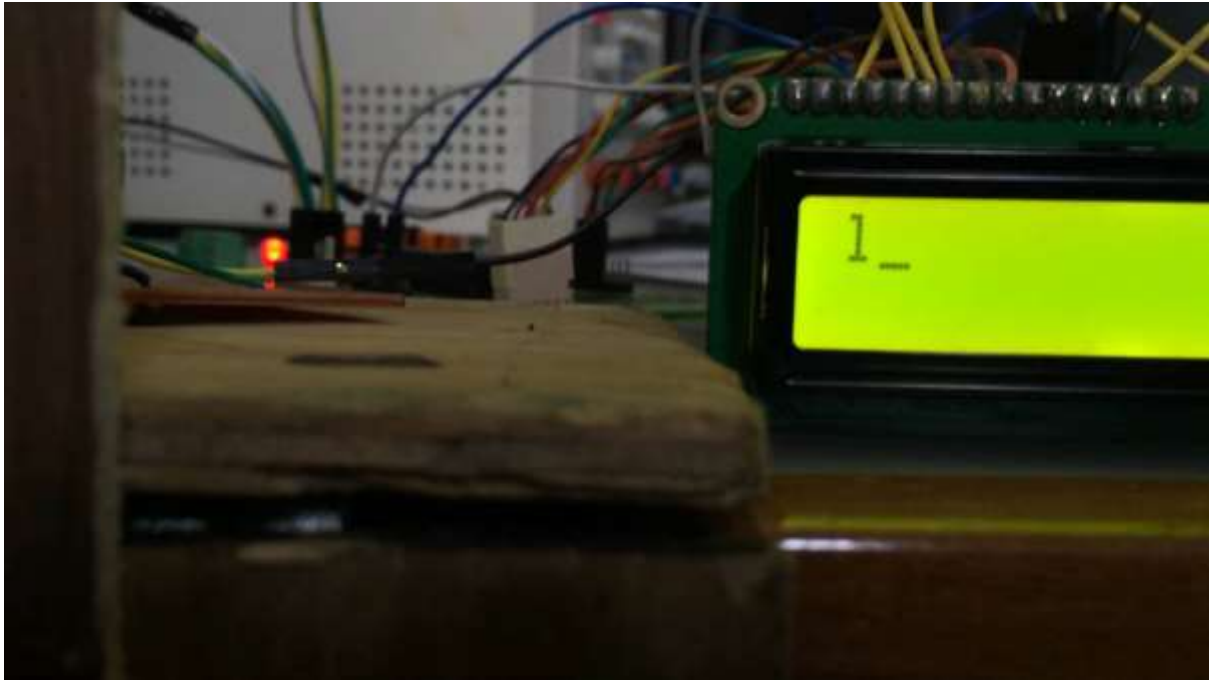
## **Snapshots**

The circuitry for voice controlled transcriber was partially taken from [5].



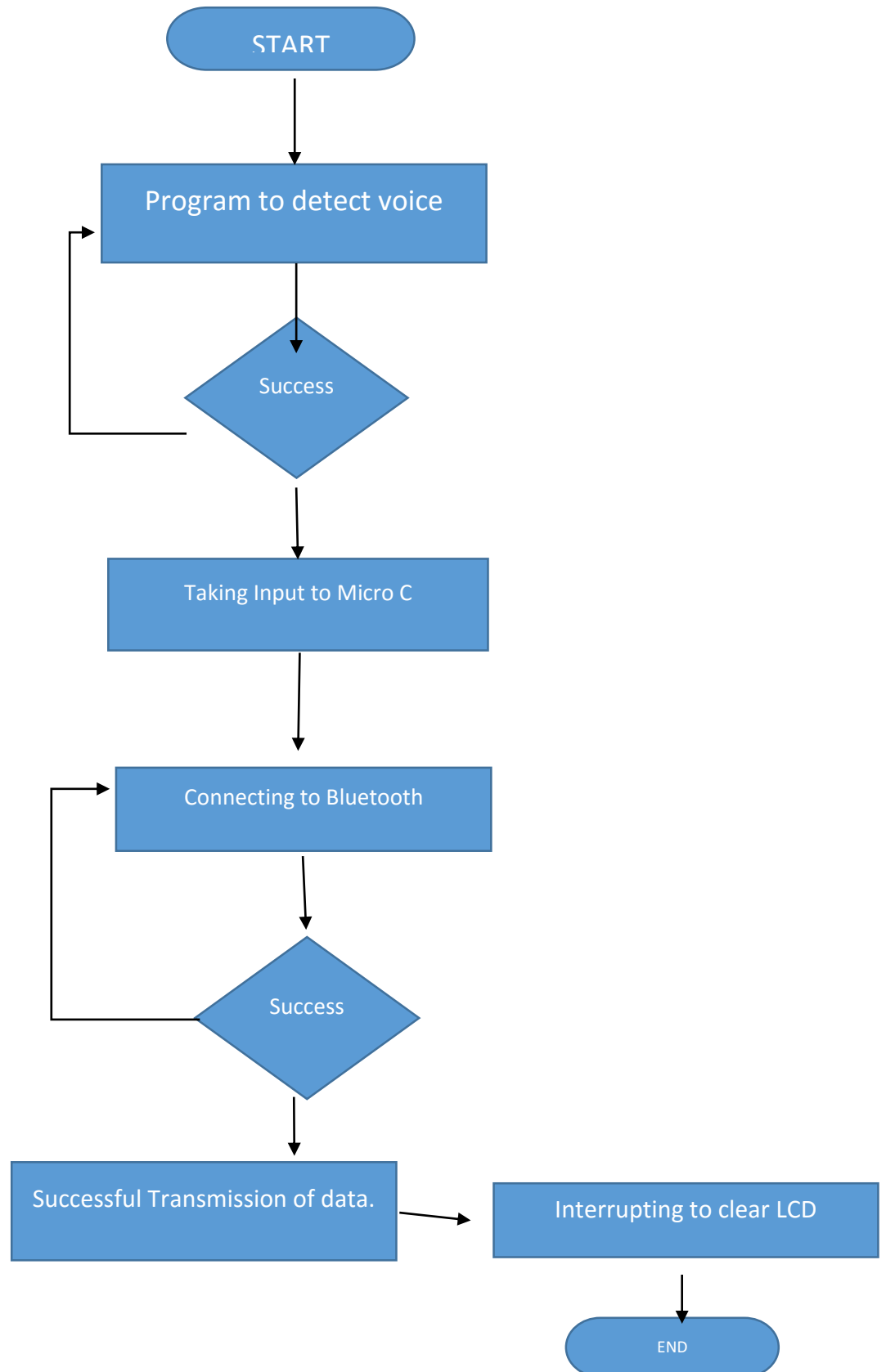




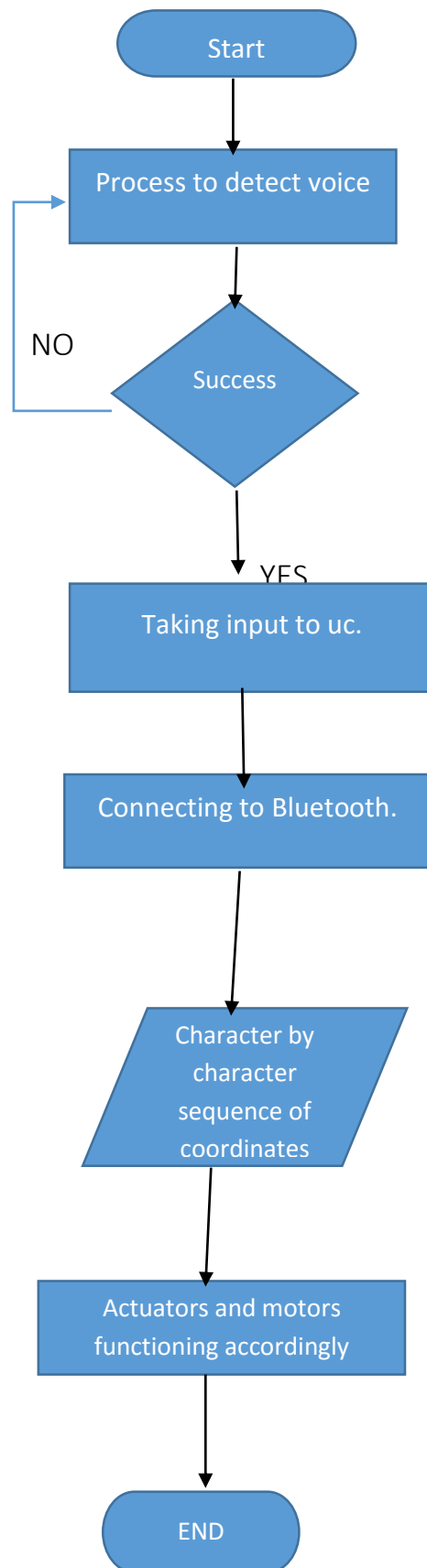


## Flow Chart

### Speech to Text:



## **Physical Transcriber:**



## Codes

### **###Speech to text (Python is used):**

```
import bluetooth
import sys
import speech_recognition as sr #Speech Recognition
import pyttsx #Text to speech

# Global Variables
googleRec = ">>>Google Speech Recognition thinks you said: "
say = ">>>Say Something"
notUnderstood = ">>>Google Speech Recognition could not understand what you said."
continueOrNot = ">>>Would you like to continue or not? yes or no."

#Text-to-Speech
engine = pyttsx.init()
rate = engine.getProperty('rate') #default = 200
engine.setProperty('rate', rate-20)
voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id)

target_name = "HC-05"
target_address = None

nearby_devices = bluetooth.discover_devices()
#print nearby_devices

for bdaddr in nearby_devices:
#    print bluetooth.lookup_name( bdaddr )
    if target_name == bluetooth.lookup_name( bdaddr ):
        target_address = bdaddr
        break

# Function to recognize audio
def recognize_audio(printString):
    r = sr.Recognizer()
    with sr.Microphone() as source:
        r.adjust_for_ambient_noise(source)
        engine.say(printString)
        engine.runAndWait()
        print printString

    audio = r.listen(source)
```

```

try:
    rec = r.recognize_google(audio, None, "en-IN", False)
    print googleRec + rec
    return rec
    #print rec
except sr.UnknownValueError:
    print notUnderstood
    return notUnderstood
except sr.RequestError as e:
    print "Could not request results from Google Speech Recognition Service;
{0}".format(e)
    return notUnderstood

if __name__ == '__main__':

    if target_address is not None:
        #Instantiating bluetooth socket
        sock=bluetooth.BluetoothSocket(bluetooth.RFCOMM)
        print ">>Trying connection to HC-05..."
        port = 1

        #Connecting to the predefined Bluetooth Address of our HC-05 module
        sock.connect(('98:D3:32:20:50:9F', port))
        print ">>CONNECTED!"
#    print "found target bluetooth device with address ", target_address

    rec = ""
    flag = True #Flag for re-asking the user whether to continue or not
    rec = recognize_audio(say)
    while 1:
        if rec!=notUnderstood and flag:
            #rec = 'hello'
            rec = rec.encode('utf-8') #encoding in 8-bit character format

            if target_address is not None:
                print ">>Trying sending: %s" % rec
                sock.send(rec + '\r') #Sending recognized string over bluetooth
                print ">>Finished sending!"

        rec1 = recognize_audio(continueOrNot)

        if rec1 == "no": #Exit condition
            if target_address is not None:
                sock.close()
                exit()
        elif rec1 == "yes": #Continue condition
            rec = recognize_audio(say)
            flag = True
        else:
            flag = False

```

The libraries taken for speech to text was being studied from [6].

### **###Transcriber and LCD display:**

```
/*
 * FinalProject.c
 *
 * Created: 25-04-2016 12:30:52
 * Author : User
 */

#ifndef F_CPU
    #define F_CPU 8000000UL
#endif
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <math.h>

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//Functions to move the pointer up, down, left, right, diagonally left/right up and diagonally
left/right down
void up(int iteration) {
    int i=0;
    while(i!=5*iteration) {
        /*The following sequence makes motor 1 work clockwise,
        and motor 2 to work anticlockwise where each step has
        a 25 ms delay*/
        PORTC=0x9C;
        _delay_ms(25);
        PORTC=0x36;
        _delay_ms(25);
        PORTC=0x63;
        _delay_ms(25);
        PORTC=0xC9;
        _delay_ms(25);
        i++;
    }
}

void down(int iteration) {
```

```

    int i=0;
    while(i!=5*iteration) {
        /*The following sequence makes motor 1 work clockwise,
        and motor 2 to work anticlockwise where each step has
        a 25 ms delay*/
        PORTC=0xC9;
        _delay_ms(25);
        PORTC=0x63;
        _delay_ms(25);
        PORTC=0x36;
        _delay_ms(25);
        PORTC=0x9C;
        _delay_ms(25);
        i++;
    }
}

void right(int iteration) {
    int i=0;
    while(i!=5*iteration) {
        /*The following sequence makes motor 1 work clockwise,
        and motor 2 to work anticlockwise where each step has
        a 25 ms delay*/
        PORTC=0x99;
        _delay_ms(25);
        PORTC=0x33;
        _delay_ms(25);
        PORTC=0x66;
        _delay_ms(25);
        PORTC=0xCC;
        _delay_ms(25);
        i++;
    }
}

void left(int iteration) {
    int i=0;
    while(i!=5*iteration) {
        /*The following sequence makes motor 1 work clockwise,
        and motor 2 to work anticlockwise where each step has
        a 25 ms delay*/
        PORTC=0xCC;
        _delay_ms(25);
        PORTC=0x66;
        _delay_ms(25);
        PORTC=0x33;
        _delay_ms(25);
        PORTC=0x99;
        _delay_ms(25);
        i++;
    }
}

```



```

    }
}

void rightdiagup(int iteration)
{
    int i=0;
    while(i!=5*iteration){
        /*The following sequence makes motor 1 work clockwise,
        and motor 2 to work anticlockwise where each step has
        a 25 ms delay*/
        PORTC=0x09;
        _delay_ms(25);
        PORTC=0x03;
        _delay_ms(25);
        PORTC=0x06;
        _delay_ms(25);
        PORTC=0x0C;
        _delay_ms(25);
        i++;
    }
}

void leftdiagdown(int iteration) {
    int i=0;
    while(i!=5*iteration) {
        /*The following sequence makes motor 1 work clockwise,
        and motor 2 to work anticlockwise where each step has
        a 25 ms delay*/
        PORTC=0x0C;
        _delay_ms(25);
        PORTC=0x06;
        _delay_ms(25);
        PORTC=0x03;
        _delay_ms(25);
        PORTC=0x09;
        _delay_ms(25);
        i++;
    }
}

void leftdiagup(int iteration) {
    int i=0;
    while(i!=5*iteration) {
        /*The following sequence makes motor 1 work clockwise,
        and motor 2 to work anticlockwise where each step has
        a 25 ms delay*/
        PORTC=0xC0;
        _delay_ms(25);
        PORTC=0x60;
        _delay_ms(25);
    }
}

```



```

// to provide data to LCD
void LCD_Data(unsigned char Data) {
    PORTB=Data&0xF0; // Send Higher nibble (D7-D4)

    PORTA|=(1<<2);    // Register Select =1 (for data select register)
    PORTA|=(1<<6);    //Enable=1 for H to L pulse

    _delay_us(5);

    PORTA^=(1<<6);
    PORTB=((Data<<4)&0xF0); // Send Lower nibble (D3-D0)
    PORTA|=(1<<6);    //Enable=1 for H to L pulse

    _delay_us(5);

    PORTA^=(1<<6);

    _delay_us(100);
}

// to print string to LCD
void LCD_Print(char * str) {
    unsigned char i=0;
    // Till NULL character is reached, take each character
    while(*(str+i)!=0) {
        LCD_Data(*(str+i)); // Data sent to LCD data register
        i++;
        _delay_ms(10);
    }
}

// to provide command to LCD
void lcdcommand(unsigned char command) {
    PORTB=command&0xF0; // Send Higher nibble (D7-D4)

    PORTA&=~(1<<2); // Register Select =0 (for Command register)
    PORTA|=(1<<6); //Enable=1 for H to L pulse

    _delay_us(5);
    PORTA^=(1<<6);
    _delay_us(100);

    PORTB=((command<<4)&0xF0); // Send Lower nibble (D3-D0)
    PORTA|=(1<<6); //Enable=1 for H to L pulse
    _delay_us(5);
    PORTA^=(1<<6);

    _delay_us(40);
}

```

```

// Cursor Position
void Cursor_Position(unsigned short int x,unsigned short int y) {
    unsigned char firstcharadd[] ={0x80,0xC0}; // First line address 0X80
    //Second line address 0XC0
    lcdcommand((firstcharadd[x-1]+y-1)); // to get address line on LCD
}

// To clear LCD previous data
void clear() {
    lcdcommand(0x01); // to clear LCD
    _delay_ms(2);
}

//LCD Initialize
void LCD_Initialize() {
    PORTA&=~(1<<6);

    lcdcommand(0x33); // Initialize LCD for 4 bit mode
    lcdcommand(0x32); // Initialize LCD for 4 bit mode
    lcdcommand(0x28); // Initialize LCD for 5X7 matrix mode
    lcdcommand(0x0E); //Display on,cursor blinking
    clear();
    lcdcommand(0x06); //Shift cursor to right
    lcdcommand(0x80);
}

int main(void) {
    char pass[100]={ };
    unsigned char flag=0,i=0,k=0,j=0;
    usart_initialize();
    //Set-up PORTS for LCD
    DDRB=0xF0; // For D3-D0
    DDRC=0xFF; //Port C as output Pin0-3 controls motor 1 while Pin 4-7 controls
motor 2
    PORTC=0x00;
    DDRA|=(1<<2); //PORTA for LCD RS
    DDRA|=(1<<6); //PORTA for LCD EN

    LCD_Initialize(); //Initialize

    clear();

    while(flag!=1) {
        while((UCSRA&(1<<RXC))==0);
        pass[i++] = UDR;
        usart_send(pass[i-1]);
        if(UDR=='r'||pass[i-1]=='r')
            { flag=1;pass[i-1]='\0';}
    }
}

```

```

    }
    // i = received character length + 1

    //Dynamic for-loop for printing characters on 20x2 LCD with cases like overflow of
    characters printed on LCD handled.
    for(k=0;k<2;k++) {
        for(j=1;j+(k*16)<=(i-1)&& j<=16;j++) {
            Cursor_Position(k+1,j);
            LCD_Data(pass[j-1+(k*16)]);
            _delay_ms(20);
        }
    }

    for(k=0;pass[k]!='\0';k++) {
        if(pass[k]=='a' || pass[k]=='A') {
            up(2);
            right(2);
            down(1);
            left(2);
            right(2);
            down(1);
        }
        else if(pass[k]=='b' || pass[k]=='B') {
            up(2);
            right(2);
            down(1);
            left(2);
            right(2);
            down(1);
            left(2);
            right(2);
        }
        else if(pass[k]=='c' || pass[k]=='C') {
            up(2);
            right(2);
            left(2);
            down(2);
            right(2);
        }
        else if(pass[k]=='d' || pass[k]=='D') {
            up(2);
            right(2);
            down(2);
            left(2);
            right(2);
        }
        else if(pass[k]=='e' || pass[k]=='E') {
            up(2);
            right(2);
            left(2);
        }
    }

```

```

        down(1);
        right(2);
        left(2);
        down(1);
        right(2);
    }
    else if(pass[k]=='f' || pass[k]=='F') {
        up(2);
        right(2);
        left(2);
        down(1);
        right(1);
        left(1);
        down(1);
    }
    else if(pass[k]=='g' || pass[k]=='G') {
        up(2);
        right(2);
        left(2);
        down(2);
        right(2);
        up(1);
        down(1);
    }
    else if(pass[k]=='h' || pass[k]=='H') {
        up(2);
        down(1);
        right(2);
        up(1);
        down(2);
    }
    else if(pass[k]=='i' || pass[k]=='I') {
        right(1);
        up(2);
        left(1);
        right(2);
        left(1);
        down(2);
        right(1);
    }
    else if(pass[k]=='j' || pass[k]=='J') {
        up(1);
        down(1);
        right(2);
        up(2);
        left(2);
        right(2);
        down(2);
    }
    else if(pass[k]=='k' || pass[k]=='K') {

```

```

        up(2);
        down(1);
        rightdiagup(1);
        leftdiagdown(1);
        right(1);
        down(1);
        right(1);
    }
    else if(pass[k]=='I' || pass[k]=='L') {
        up(2);
        down(2);
        right(2);
    }
    else if(pass[k]=='m' || pass[k]=='M') {
        up(2);
        right(1);
        down(1);
        right(1);
        down(2);
    }
    else if(pass[k]=='n' || pass[k]=='N') {
        up(2);
        right(1);
        down(2);
        right(1);
        up(2);
        down(2);
    }
    else if(pass[k]=='o' || pass[k]=='O') {
        right(1);
        leftdiagup(1);
        rightdiagup(1);
        rightdiagdown(1);
        leftdiagdown(1);
        right(1);
    }
    else if(pass[k]=='p' || pass[k]=='P') {
        up(2);
        right(2);
        down(1);
        left(2);
        down(1);
        right(1);
    }
    else if(pass[k]=='q' || pass[k]=='Q') {
        right(2);
        up(2);
        left(2);
        down(1);
        right(2);
    }

```

```

        down(1);
    }
    else if(pass[k]=='r' || pass[k]=='R') {
        up(2);
        right(2);
        down(1);
        left(2);
        right(1);
        down(1);
        right(1);
    }
    else if(pass[k]=='s' || pass[k]=='S') {
        right(2);
        up(1);
        left(2);
        up(1);
        right(2);
        left(2);
        down(1);
        right(2);
        down(1);
    }
    else if(pass[k]=='t' || pass[k]=='T') {
        right(1);
        up(2);
        down(1);
        right(1);
        left(1);
        down(1);
        right(1);
    }
    else if(pass[k]=='u' || pass[k]=='U') {
        up(2);
        down(2);
        right(2);
        up(2);
        down(2);
    }
    else if(pass[k]=='v' || pass[k]=='P') {
        right(1);
        leftdiagup(1);
        up(1);
        down(1);
        rightdiagdown(1);
        rightdiagup(1);
        up(1);
        down(1);
        leftdiagdown(1);
        right(1);
    }
}

```



```

else if(pass[k]=='w' || pass[k]=='W') {
    up(2);
    down(2);
    right(1);
    up(1);
    down(1);
    right(1);
    up(2);
    down(2);
}
else if(pass[k]=='x' || pass[k]=='X') {
    right(1);
    up(1);
    left(1);
    up(1);
    down(1);
    right(1);
    up(1);
    right(1);
    left(1);
    down(1);
    right(1);
    down(1);
}
else if(pass[k]=='y' || pass[k]=='Y') {
    right(1);
    up(1);
    leftdiagup(1);
    rightdiagdown(1);
    rightdiagup(1);
    leftdiagdown(1);
    down(1);
    right(1);
}
else if(pass[k]=='z' || pass[k]=='Z') {
    up(1);
    right(2);
    up(1);
    left(2);
    right(2);
    down(1);
    left(2);
    down(1);
    right(2);
}
}
while(1) {
    Cursor_Position(1,16);
    LCD_Print(pass);
    lcdcommand(0x18);

```

```

        _delay_ms(300);
    }
    return 0;
}

```

### **More Stable Code (Future Work\*):**

```

/*
 * FinalProject.c
 *
 * Created: 25-04-2016 12:30:52
 * Author : User
 */

#ifndef F_CPU
    #define F_CPU 8000000UL
#endif
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <math.h>

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
//Note that all the values in the coming functions and variables are in terms of cms

float strlenX = 11, strlenY = 23; // Global Variables for accessing string lengths of Stepper
Motors X and Y
float lenA = 3.5, lenB = 13.5; // Horizontal and Vertical displacement of pointer w.r.t Stepper
Motor X
float lenXY = 24; // Distance between Stepper Motors X and Y

//Previous motor positions and respective arrays of stepvalues of stepper motors
unsigned char prevMotorPosX = 3;
unsigned char prevMotorPosY = 3;
unsigned char stepValuesX[] = {0X90,0XC0,0X60,0X30};
unsigned char stepValuesY[] = {0X09,0X0C,0X06,0X03};

//Explicit function definition
void rotateMotor(float newX, float newY);

// moveXY function will receive variables a and b as positive X displacement and negative Y
displacement in cms respectively

```

// and it will update string lengths of stepper motors X and Y and displacement of pointer according to the Pythagoras theorem.

// Negating the values passed in arguments will result in reverse motion of stepper motors.

```
void moveXY(float a, float b) {  
    float newstrX,newstrY;  
    lenA += a;  
    lenB += b;  
    newstrX = sqrt((lenA*lenA)+(lenB*lenB));  
    newstrY = sqrt(((lenXY-lenA)*(lenXY-lenA))+(lenB*lenB));  
    rotateMotor(newstrX,newstrY); //calling the function with new string values  
}
```

// According to the new string values received, the rotateMotor function will rotate motors with 4 different cases

// of combination of decreased-increased new motor lengths X and Y w.r.t old motor lengths and with precision of 0.325 cms

// while updating new String lengths and previous motor positions in real-time

```
void rotateMotor(float newX, float newY) {  
  
    while(newX-strlenX>0&&newY-strlenY>0){  
        _delay_ms(100);  
        if(newX-strlenX>0.325&&newY-strlenY>0.325) {  
            prevMotorPosX = (prevMotorPosX + 1) % 4;  
            prevMotorPosY = (prevMotorPosY - 1 + 4) % 4;  
  
            PORTC =  
stepValuesX[prevMotorPosX]|stepValuesY[prevMotorPosY];  
            strlenX += 0.325;  
            strlenY += 0.325;  
        }  
        else if(newX-strlenX>0.325) {  
            prevMotorPosX = (prevMotorPosX + 1) % 4;  
            PORTC = stepValuesX[prevMotorPosX]|(PORTC&0x0F);  
            strlenX += 0.325;  
        }  
        else if(newY-strlenY>0.325) {  
            prevMotorPosY = (prevMotorPosY - 1 + 4) % 4;  
            PORTC = stepValuesY[prevMotorPosY]|(PORTC&0xF0);  
            strlenY += 0.325;  
        }  
        else  
            break;  
    }  
  
    while(newX-strlenX<0&&newY-strlenY>0){  
        _delay_ms(100);  
        if(newX-strlenX<-0.325&&newY-strlenY>0.325) {  
            prevMotorPosX = (prevMotorPosX - 1 + 4) % 4;  
            prevMotorPosY = (prevMotorPosY - 1 + 4) % 4;
```

```

        PORTC =
stepValuesX[prevMotorPosX]|stepValuesY[prevMotorPosY];
        strlenX -= 0.325;
        strlenY += 0.325;
    }
    else if(newX-strlenX<-0.325) {
        prevMotorPosX = (prevMotorPosX - 1 + 4) % 4;
        PORTC = stepValuesX[prevMotorPosX]|(PORTC&0x0F);
        strlenX -= 0.325;
    }
    else if(newY-strlenY>0.325) {
        prevMotorPosY = (prevMotorPosY - 1 + 4) % 4;
        PORTC = stepValuesY[prevMotorPosY]|(PORTC&0xF0);
        strlenY += 0.325;
    }
    else
        break;
}

while(newX-strlenX>0&&newY-strlenY<0){
    _delay_ms(100);
    if(newX-strlenX>0.325&&newY-strlenY<-0.325) {
        prevMotorPosX = (prevMotorPosX + 1) % 4;
        prevMotorPosY = (prevMotorPosY + 1) % 4;

        PORTC =
stepValuesX[prevMotorPosX]|stepValuesY[prevMotorPosY];
        strlenX += 0.325;
        strlenY -= 0.325;
    }
    else if(newX-strlenX>0.325) {
        prevMotorPosX = (prevMotorPosX + 1) % 4;
        PORTC |= stepValuesX[prevMotorPosX]|(PORTC&0x0F);
        strlenX += 0.325;
    }
    else if(newY-strlenY<-0.325) {
        prevMotorPosY = (prevMotorPosY + 1) % 4;
        PORTC |= stepValuesY[prevMotorPosY]|(PORTC&0xF0);
        strlenY -= 0.325;
    }
    else
        break;
}

while(newX-strlenX<0&&newY-strlenY<0){
    _delay_ms(100);
    if(newX-strlenX<-0.325&&newY-strlenY<-0.325) {
        prevMotorPosX = (prevMotorPosX - 1 + 4) % 4;
        prevMotorPosY = (prevMotorPosY + 1) % 4;

```

```

        PORTC =
stepValuesX[prevMotorPosX]|stepValuesY[prevMotorPosY];
        strlenX -= 0.325;
        strlenY -= 0.325;
    }
    else if(newX-strlenX<-0.325) {
        prevMotorPosX = (prevMotorPosX + 1) % 4;
        PORTC = stepValuesX[prevMotorPosX]|(PORTC&0x0F);
        strlenX -= 0.325;
    }
    else if(newY-strlenY<-0.325) {
        prevMotorPosY = (prevMotorPosY + 1) % 4;
        PORTC = stepValuesY[prevMotorPosY]|(PORTC&0xF0);
        strlenY -= 0.325;
    }
    else
        break;
}
}

//Defining motion of pointer for each character
void printChar(unsigned char ch) {
    float one=3,two=6,three=9,lena=3*lenA; //prescalar in cms for one, two and three
units
    if(ch=='a' || ch=='A') {
        moveXY(0,-two);
        moveXY(two,0);
        moveXY(0,one);
        moveXY(-two,0);
        moveXY(two,0);
        moveXY(0,one);
    }
    else if(ch=='b' || ch=='B') {
        moveXY(0,-two);
        moveXY(two,0);
        moveXY(0,one);
        moveXY(-two,0);
        moveXY(two,0);
        moveXY(0,one);
        moveXY(-two,0);
        moveXY(two,0);
    }
    else if(ch=='c' || ch=='C') {
        moveXY(0,-two);
        moveXY(two,0);
        moveXY(-two,0);
        moveXY(0,two);
        moveXY(two,0);
    }
    else if(ch=='d' || ch=='D') {

```

```

        moveXY(0,-two);
        moveXY(two,0);
        moveXY(0,two);
        moveXY(-two,0);
        moveXY(two,0);
    }
    else if(ch=='e' || ch=='E') {
        moveXY(0,-two);
        moveXY(two,0);
        moveXY(-two,0);
        moveXY(0,one);
        moveXY(two,0);
        moveXY(-two,0);
        moveXY(0,one);
        moveXY(two,0);
    }
    else if(ch=='f' || ch=='F') {
        moveXY(0,-two);
        moveXY(two,0);
        moveXY(-two,0);
        moveXY(0,one);
        moveXY(one,0);
        moveXY(-one,0);
        moveXY(0,one);
    }
    else if(ch=='g' || ch=='G') {
        moveXY(0,-two);
        moveXY(two,0);
        moveXY(-two,0);
        moveXY(0,two);
        moveXY(two,0);
        moveXY(0,-one);
        moveXY(0,one);
    }
    else if(ch=='h' || ch=='H') {
        moveXY(0,-two);
        moveXY(0,one);
        moveXY(two,0);
        moveXY(0,-one);
        moveXY(0,two);
    }
    else if(ch=='i' || ch=='I') {
        moveXY(one,0);
        moveXY(0,-two);
        moveXY(-one,0);
        moveXY(two,0);
        moveXY(-one,0);
        moveXY(0,two);
        moveXY(one,0);
    }
}

```

```

else if(ch=='j' || ch=='J') {
    moveXY(0,-one);
    moveXY(0,one);
    moveXY(two,0);
    moveXY(0,-two);
    moveXY(-two,0);
    moveXY(two,0);
    moveXY(0,two);
}
else if(ch=='k' || ch=='K') {
    moveXY(0,-two);
    moveXY(0,one);
    moveXY(one,-one);
    moveXY(-one,one);
    moveXY(one,one);
}
else if(ch=='l' || ch=='L') {
    moveXY(0,-two);
    moveXY(0,two);
    moveXY(two,0);
}
else if(ch=='n' || ch=='N') {
    moveXY(0,-two);
    moveXY(two,two);
    moveXY(0,-two);
    moveXY(0,two);
}
else if(ch=='o' || ch=='O') {
    moveXY(one,0);
    moveXY(-one,-one);
    moveXY(one,-one);
    moveXY(one,one);
    moveXY(-one,one);
}
else if(ch=='p' || ch=='P') {
    moveXY(0,-two);
    moveXY(two,0);
    moveXY(0,one);
    moveXY(-two,0);
    moveXY(0,one);
    moveXY(one,0);
}
else if(ch=='q' || ch=='Q') {
    moveXY(two,0);
    moveXY(0,-two);
    moveXY(-two,0);
    moveXY(0,one);
    moveXY(two,0);
    moveXY(0,one);
}

```

```

else if(ch=='r' || ch=='R') {
    moveXY(0,-two);
    moveXY(two,0);
    moveXY(0,one);
    moveXY(-two,0);
    moveXY(one,one);
}
else if(ch=='s' || ch=='S') {
    moveXY(two,0);
    moveXY(0,-one);
    moveXY(-two,0);
    moveXY(0,-one);
    moveXY(two,0);
    moveXY(-two,0);
    moveXY(0,one);
    moveXY(two,0);
    moveXY(0,one);
}
else if(ch=='t' || ch=='T') {
    moveXY(one,0);
    moveXY(0,-two);
    moveXY(0,one);
    moveXY(one,0);
    moveXY(-one,0);
    moveXY(0,one);
    moveXY(one,0);
}
else if(ch=='u' || ch=='U') {
    moveXY(0,-two);
    moveXY(0,two);
    moveXY(two,0);
    moveXY(0,-two);
    moveXY(0,two);
}
else if(ch=='v' || ch=='V') {
    moveXY(one,0);
    moveXY(-one,-one);
    moveXY(0,-one);
    moveXY(0,one);
    moveXY(one,one);
    moveXY(one,-one);
    moveXY(0,-one);
    moveXY(0,one);
    moveXY(-one,one);
    moveXY(one,0);
}
else if(ch=='w' || ch=='W') {
    moveXY(0,-two);
    moveXY(0,two);
    moveXY(one,0);

```



```

        moveXY(0,-one);
        moveXY(0,one);
        moveXY(one,0);
        moveXY(0,-two);
        moveXY(0,two);
    }
    else if(ch=='x' || ch=='X') {
        moveXY(two,-two);
        moveXY(-one,one);
        moveXY(-one,-one);
        moveXY(two,two);
    }
    else if(ch=='y' || ch=='Y') {
        moveXY(one,0);
        moveXY(0,-one);
        moveXY(-one,-one);
        moveXY(one,one);
        moveXY(one,-one);
        moveXY(-one,one);
        moveXY(0,one);
        moveXY(one,0);
    }
    else if(ch=='z' || ch=='Z') {
        moveXY(two,-two);
        moveXY(-two,0);
        moveXY(two,0);
        moveXY(-two,two);
        moveXY(two,0);
    }
    else if(ch==' '&&lenXY-lenA>2) {
        moveXY(one,0);
    }
    else if(ch==' '&&lenXY-lenA<=2) {
        moveXY(0,one);
        moveXY(lena,0);
        moveXY(0,three);
        moveXY(one,0);
    }
    if(ch!=' '&&lenXY-lenA>2) {
        moveXY(one,0);
    }
    else if(ch!=' ') {
        moveXY(0,one);
        moveXY(lena,0);
        moveXY(0,three);
        moveXY(one,0);
    }
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////

```

```

//Initializing usart
void usart_initialize() {
    UCSRB=0x18;
    UCSRC=0x86;
    UBRRL=0x33;
}

void usart_send(unsigned char ch) {
    while(!(UCSRA & (1<<UDRE))); // Wait till UDR is empty
    UDR=ch; //Write the value to be Tx
}

/*
//LCD Connections
#define LCD_RS PORTA.B2 // RS
#define LCD_EN PORTA.B6 //Enable
#define LCD_D4 PORTB.B4 //Data Bit 4
#define LCD_D5 PORTB.B5 //Data Bit 5
#define LCD_D6 PORTB.B6 //Data Bit 6
#define LCD_D7 PORTB.B7 //Data Bit 7

*/
// to provide data to LCD
void LCD_Data(unsigned char Data) {
    PORTB=Data&0xF0; // Send Higher nibble (D7-D4)

    PORTA|=(1<<2); // Register Select =1 (for data select register)
    PORTA|=(1<<6); //Enable=1 for H to L pulse

    _delay_us(5);

    PORTA^=(1<<6);
    PORTB=((Data<<4)&0xF0); // Send Lower nibble (D3-D0)
    PORTA|=(1<<6); //Enable=1 for H to L pulse

    _delay_us(5);

    PORTA^=(1<<6);

    _delay_us(100);
}

// to print string to LCD
void LCD_Print(char * str) {
    unsigned char i=0;
    // Till NULL character is reached, take each character
    while(*(str+i)!=0) {
        LCD_Data(*(str+i)); // Data sent to LCD data register
        i++;
    }
}

```

```

        _delay_ms(10);
    }
}

// to provide command to LCD
void lcdcommand(unsigned char command) {
    PORTB=command&0xF0; // Send Higher nibble (D7-D4)

    PORTA&=~(1<<2); // Register Select =0 (for Command register)
    PORTA|=(1<<6); //Enable=1 for H to L pulse

    _delay_us(5);
    PORTA^=(1<<6);
    _delay_us(100);

    PORTB=((command<<4)&0xF0); // Send Lower nibble (D3-D0)
    PORTA|=(1<<6); //Enable=1 for H to L pulse
    _delay_us(5);
    PORTA^=(1<<6);

    _delay_us(40);
}

// Cursor Position
void Cursor_Position(unsigned short int x,unsigned short int y) {
    unsigned char firstcharadd[]={0x80,0xC0}; // First line address 0X80
    //Second line address 0XC0
    lcdcommand((firstcharadd[x-1]+y-1)); // to get address line on LCD
}

// To clear LCD previous data
void clear() {
    lcdcommand(0x01); // to clear LCD
    _delay_ms(2);
}

//LCD Initialize
void LCD_Initialize() {
    PORTA&=~(1<<6);

    lcdcommand(0x33); // Initialize LCD for 4 bit mode
    lcdcommand(0x32); // Initialize LCD for 4 bit mode
    lcdcommand(0x28); // Initialize LCD for 5X7 matrix mode
    lcdcommand(0x0E); //Display on,cursor blinking
    clear();
    lcdcommand(0x06); //Shift cursor to right
    lcdcommand(0x80);
}

```

```

int main(void) {
    char pass[100]={ };
    unsigned char flag=0,i=0,k=0,j=0;
    usart_initialize();
    //Set-up PORTS for LCD
    DDRB=0xF0; // For D3-D0
    DDRC=0xFF;
    PORTC=0x00;
    DDRA|=(1<<2);          //Port D as output Pin0-3 controls motor 1 while Pin 4-7
controls motor 2
    DDRA|=(1<<6);

    LCD_Initialize(); //Initialize

    clear();

    while(flag!=1) {
        while((UCSRA&(1<<RXC))==0);
        pass[i++] = UDR;
        usart_send(pass[i-1]);
        if(UDR=='r' || pass[i-1]=='r')
            { flag=1;pass[i-1]='\0';}
    }

    for(k=0;k<2;k++) {
        for(j=1;j+(k*16)<=(i-1)&&j<=16;j++) {
            Cursor_Position(k+1,j);
            LCD_Data(pass[j-1+(k*16)]);
            _delay_ms(20);
        }
    }
    // Cursor_Position(1,1);
    //LCD_Print(pass);

    for(k=0;pass[k]!='\0';k++) {
        printChar(pass[k]);
    }

    return 0;
}

```






## Conclusion

Concluding we would state that the output was not as accurate as expected, nevertheless our speech to text module worked fine also the stepper rotated according to the letter which has to be written, the only problem was of mechanical model. Due to inefficient mechanical model at times when motor rotated our marker lifted up which should not happen.

## Future Work

Building a more stable mechanical model where the marker do not lift up while the motor rotates, we have already started working in that direction and the sample code ( last code) is for this case only. We have taken proper measurements of the page and applied the Pythagoras theorem for successful implementation of our transcriber.

## Project TimeLine

Dates Work completed	01/4/2016	04/4/2016	12/4/2016	25/4/2016	8/5/2016
Circuit Diagram					
Speech to text transmission					
Simple connections and plotter code					
Program completion					
Testing and Demo					

## **References**

- [1] <http://www.ablab.in/how-to-interface-a-hc-05-bluetooth-module-with-avr-atmega32-microcontroller/>
- [2] <http://www.atmel.com/images/doc2503.pdf>
- [3] [http://www.tec.reutlingen-university.de/uploads/media/DatenblattHC-05\\_BT-Modul.pdf](http://www.tec.reutlingen-university.de/uploads/media/DatenblattHC-05_BT-Modul.pdf)
- [4] <http://www.ablab.in/16x2-alphanumeric-lcd-interfacing-with-avr-atmega32-microcontroller/>
- [5] <https://courses.cit.cornell.edu/ee476/FinalProjects/s2001/vp2/>
- [6] <https://docs.python.org/2/>