Control of DC Motor through Incremental Encoder

Efficiently Reading Quadrature with Interrupts

Once you start wanting to control the position of a motor through feedback control, you will likely end up using a rotary encoder as the feedback device. Encoders can be broadly categorized as:

- Absolute position
- Incremental position

Absolute position encoders will tell you what the angular position is all the time, even between power cycles. Most use "gray code" (a modified form of binary) with several "tracks" (bits) to read position.

Incremental position encoders will tell you what the angular position is, but only relative to where it was when you started paying attention (usually on power-up). Two common types of incremental outputs are:

- Incremental
- Quadrature

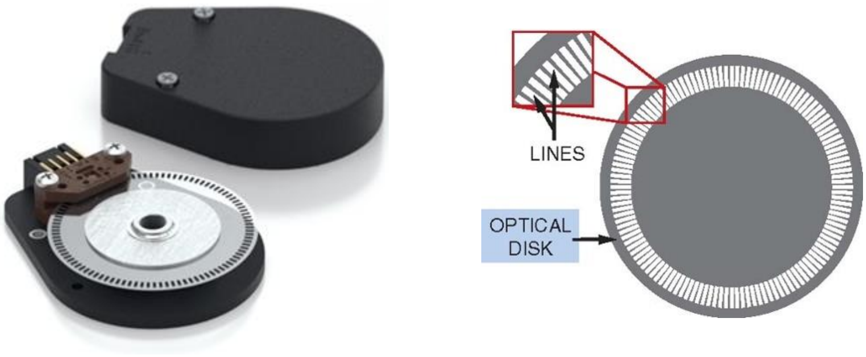


Figure 1 : Incremental and Quadrature Encoder

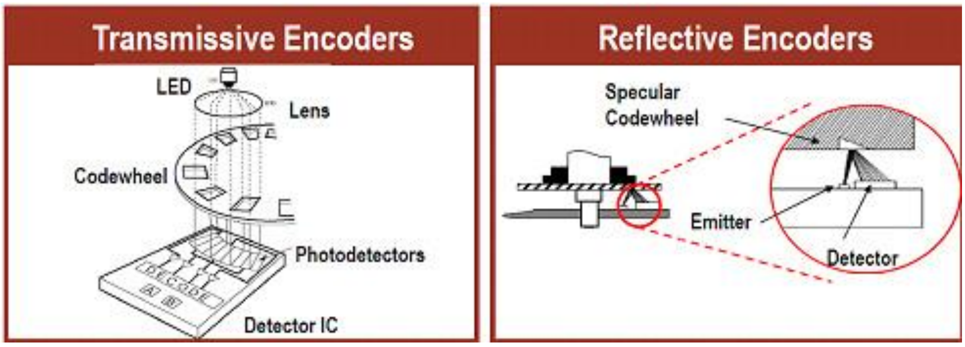
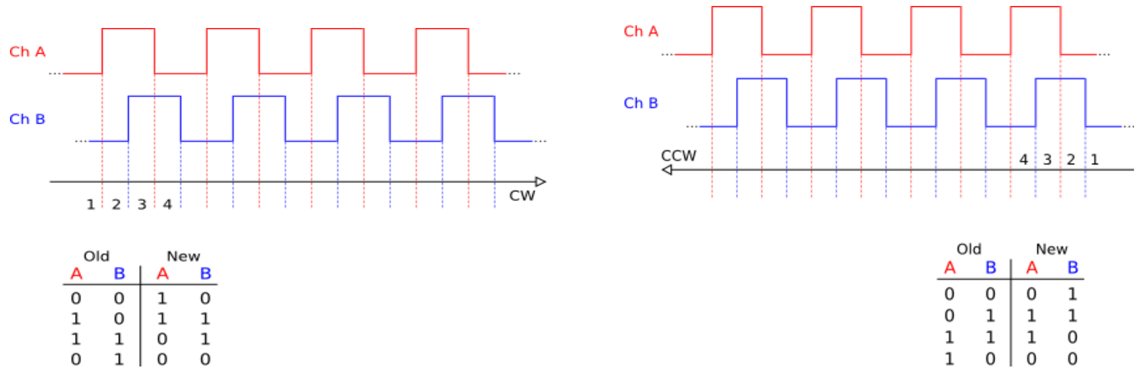


Figure 2: Types of Encoders

Incremental encoders are used for measuring speed only because it doesn’t give you any information about what direction you are turning, just that you are turning. Quadrature encoders give you direction as well as incremental position.

What is Quadrature?



Old		New	
A	B	A	B
0	0	1	0
1	0	1	1
1	1	0	1
0	1	0	0

Old		New	
A	B	A	B
0	0	0	1
0	1	1	1
1	1	1	0
1	0	0	0

Figure 3: Two square waves in quadrature.

There are two channels of output in quadrature lovingly referred to as channels A and B. They are each a square wave, but are offset from each other by 90 degrees. Whether channel A is leading or lagging channel B depends on the direction the shaft is turning, which is what allows you to determine direction. For example, both channels are low and then channel A goes high, you know that you are spinning CCW. If channel B had instead gone high before channel A, you would then know you are spinning CW. Of course, this can be deduced starting from any state as can be seen in the diagram. The output channels can be produced by a variety of means, usually either magnets in a disk attached to the shaft and a pair of hall effect sensors, or a disk with slots cut out and a pair of optical sensors looking through the slots. A google image search of "quadrature encoder" will come up with plenty of pictures to explain it. How many magnets or slots are in a revolution of the encoder is known as pulses per revolution or p/r. Only the pulses on one channel are counted, the other channel will of necessity have the same number of pulses. Encoders can range anywhere from <10 p/r to 1000's of p/r, the higher numbers giving a higher resolution. If you are looking for maximum resolution, multiply the p/r by 4 since for each pulse there are two detectable events (rising edge and falling edge) on each channel as shown in figure 1.

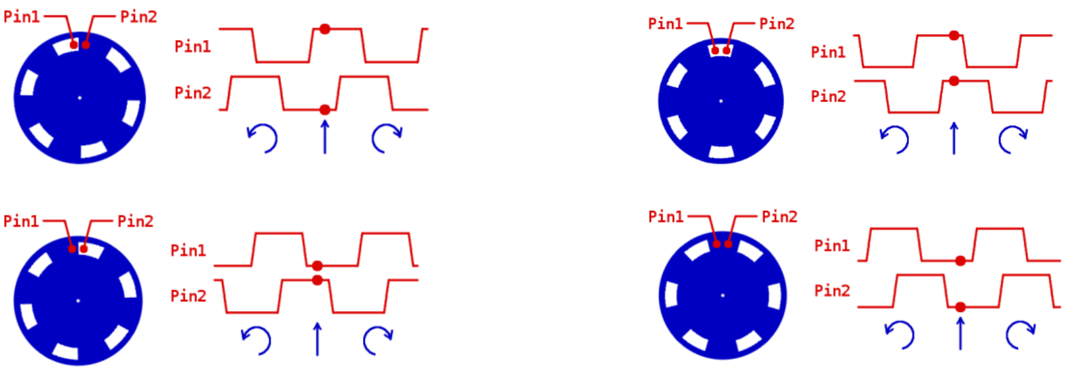


Figure 4: Rotary encoder, with corresponding channel A/B signal states shown on the right

Specification of Motor Encoder System

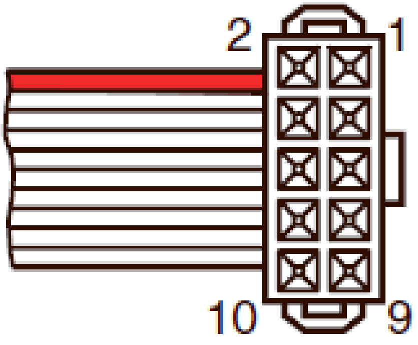
Motor: Maxon DC Motor

Motor type: 2260 881 216 200

Gear Type: 110506

Gear Ratio 139:1

Encoder Type: 136748, HEDS-6540, 1000 PPR



1	Channel A
2	V_{CC}
3	GND
4	N.C. oder GND
5	N.C. oder GND
6	GND
7	V_{CC}
8	Channel B
9	V_{CC}
10	Channel I

***Figure 5**: HEDS-6540 Encoder Connector Pin Connections*

Calculation:

$$1000\ Pulses\ \Leftrightarrow\ 360°\ \ of\ Encoder\ Movement \Leftrightarrow \left(\frac{360}{139}\right)^{0}\ Motor\ Movement$$

$$1000\ Pulses\ \times\ 4\frac{events}{pulse} \Leftrightarrow \left(\frac{360}{139}\right)^{0}\ Motor\ Movement$$

$$\therefore\ 1°\ Motor\ Movement\ =\ \frac{4000 \times 139}{360} = 1544.44\ events$$

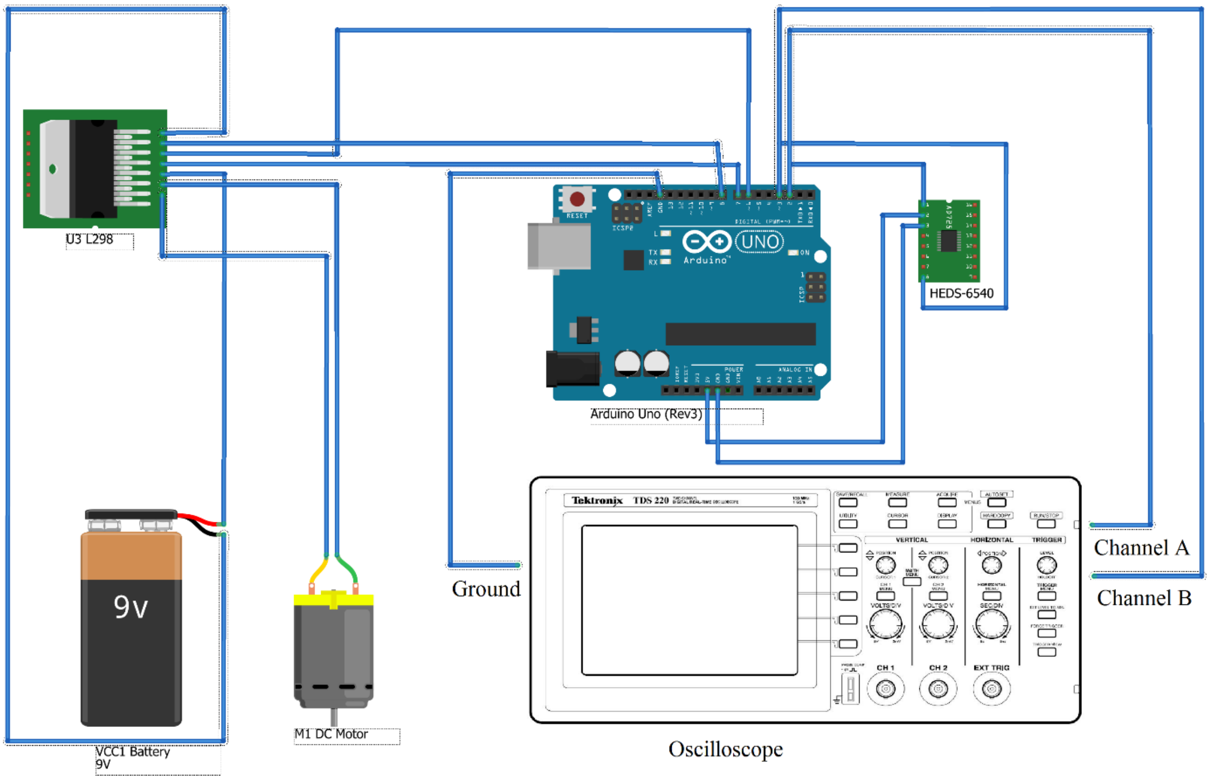


Figure 6 : Schematic diagram of circuit

ARDUINO CODE

```
//setting voltage=8.8V
#include <Encoder.h>
Encoder myEnc(2,3);
int set_pos;
float currentpos;
float error1, error2;
int m1_enable=6,m1_dir1=7,m1_dir2=8;
void setup()
{
Serial.begin(115200);
pinMode(m1_enable, OUTPUT);
pinMode(m1_dir1, OUTPUT);
pinMode(m1_dir2, OUTPUT);
}

void loop()

{set_pos = Serial.parseInt();
   while(Serial.available()>0)
   {

       Serial.print("set_pos");
       Serial.println(set_pos);
        if (set_pos > 0)
        {
        digitalWrite(m1_dir1, LOW);
        digitalWrite(m1_dir2, HIGH);
        analogWrite(m1_enable,65);
```

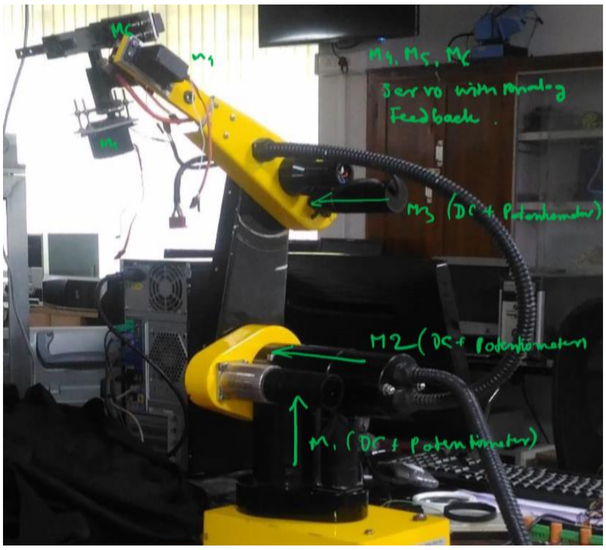
```
      currentpos =  myEnc.read();
     Serial.print("currentpos:");
     Serial.println(currentpos/1544.44);
     error1= abs(abs(set_pos)-abs(currentpos/1544.44));
     if(error1<=1)
     {
     digitalWrite(m1_dir1, HIGH);
     digitalWrite(m1_dir2, HIGH);
     analogWrite(m1_enable,65);
     delay(3000);
     currentpos =  myEnc.read();
     Serial.print("currentpos:");
     Serial.print(currentpos/1544.44);
     }


     }


     if (set_pos < 0){
     analogWrite(m1_enable,65);
     digitalWrite(m1_dir1, HIGH);
     digitalWrite(m1_dir2, LOW);
     currentpos =  myEnc.read();
      Serial.print("currentpos:");
     Serial.print(currentpos/1544.44);

     error2= abs(abs(set_pos)-abs(currentpos/1544.44));
     if(error2<=1)
     {
       digitalWrite(m1_dir1, HIGH);
       digitalWrite(m1_dir2, HIGH);
       analogWrite(m1_enable,65);
     delay(3000);
     Serial.print("currentpos:");
     Serial.println(currentpos/1544.44);
     }
}
}
}
```

Study the given DC motor plus incremental encoder circuit. Go through the program of microcontroller and understand how it used to activate motor for various applications.

Run the program and do the following:

1. Note down the time required to reach the desired angle with various PWM.
2. Suggest the possible source of errors.
3. How will you calculate the angular resolution of the motor?
4. The different strategy you can apply for direction control of motor.
5. What is your maximum speed you need to be able to track.

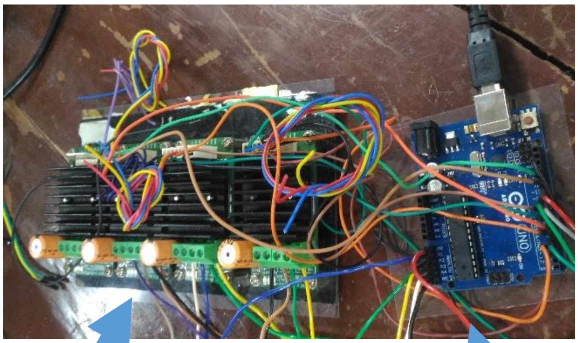
ACTIVE ROBOTIC ARM BLOCK DIAGRAM DESCRIBING WORKING PRINCIPLE



6 DEGREE ARM MECHANICAL SETUP
(3 DC MOTOR + POTENTIOMETER,
3 FEEDBACK SERVO MOTOR)

DRIVER & MICROCONTROLLER TO MECHANICAL ARM CONNECTION FOR FEEDING CURRENT & VOLTAGE TO THE MOTORS AND GETTING POSITIONAL FEEDBACK

MICROCONTROLLER + MOTOR DRIVER CIRCUIT



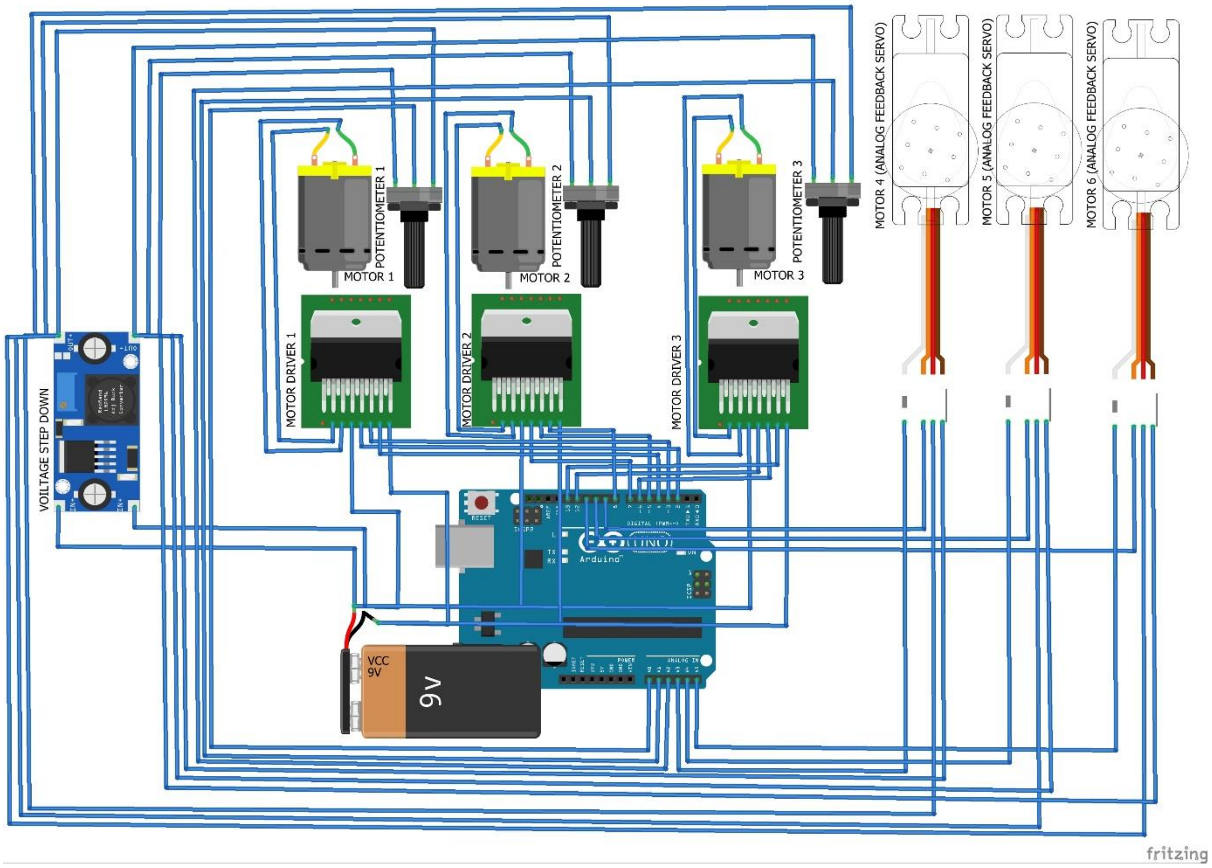
POWER SUPPLY CONNECTION FOR FEEDING IN MOTOR THROUGH DRIVER CIRCUIT

COMPUTER TO MICROCONTROLLER CONNECTION FOR PROGRAMING OF ARM CONTROL & FEEDING OF THE ANGLE VALUE REQUIRED FOR MOTOR MOVEMENT

POWER SUPPLY

COMPUTER

ACTIVE ROBOTIC ARM BLOCK DIAGRAM DESCRIBING WORKING PRINCIPLE



ARDUINO CODE:-

```
#include <Servo.h>
Servo myservo1, myservo2, myservo3;
int IN1=2;
int IN2=4;
int EN1=3;
int value1=A0;
int desiredangle1=120;
int IN3=7;
int IN4=8;
int EN2=5;
int value2=A1;
int desiredangle2=120;
int IN5=12;
int IN6=13;
int EN3=6;
int value3=A2;
int desiredangle3=120;
int servo1pin=9;
int servo2pin=10;
int servo3pin=11;
int gripperangle1=120;
int analogfeedback1=A3;
int gripperangle2=120;
int analogfeedback2=A4;
int gripperangle3=120;
int analogfeedback3=A5;


void setup() {
```

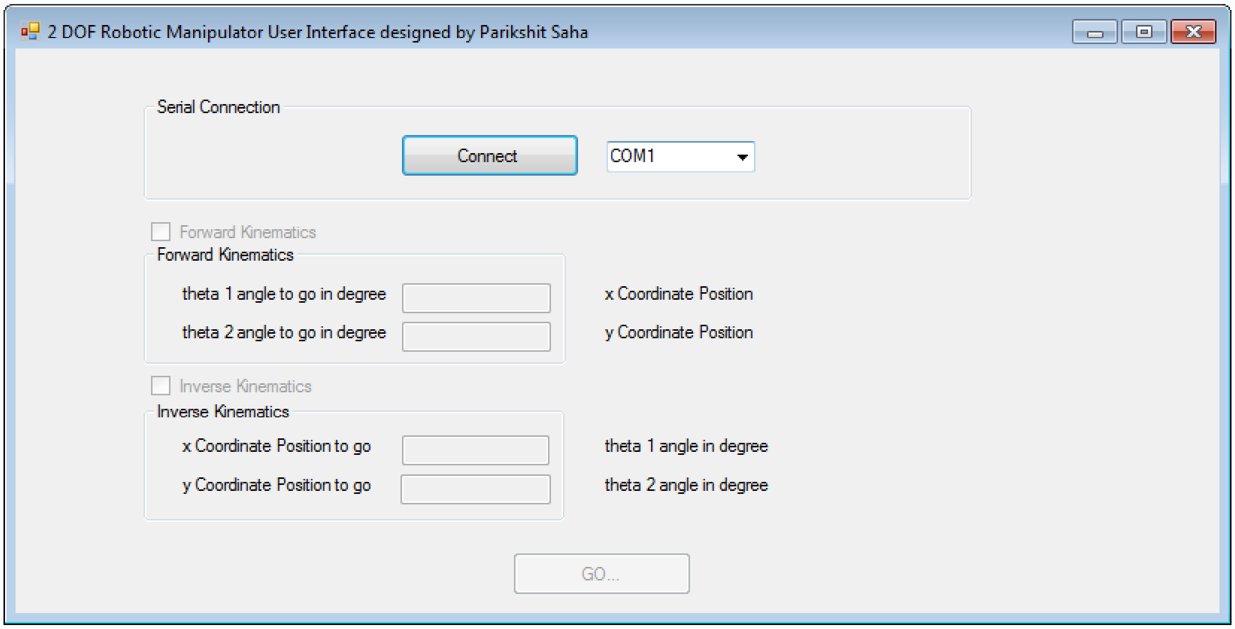
ACTIVE ROBOTIC ARM BLOCK DIAGRAM DESCRIBING WORKING PRINCIPLE

```
    Serial.begin(9600);
    pinMode(IN1,OUTPUT);
    pinMode(IN2,OUTPUT);
    pinMode(value1,INPUT);
      pinMode(IN3,OUTPUT);
    pinMode(IN4,OUTPUT);
    pinMode(value2,INPUT);
      pinMode(IN5,OUTPUT);
    pinMode(IN6,OUTPUT);
    pinMode(value3,INPUT);
      myservo1.attach(servo1pin);
      pinMode(analogfeedback1,INPUT);
      myservo2.attach(servo2pin);
      pinMode(analogfeedback2,INPUT);
      myservo3.attach(servo3pin);
      pinMode(analogfeedback3,INPUT);
    }

void loop() {
    int sensorvalue1=analogRead(value1);
    int sensorvalue2=analogRead(value2);
    int sensorvalue3=analogRead(value3);
    int sensorvalue4=analogRead(analogfeedback1);
    int sensorvalue5=analogRead(analogfeedback2);
    int sensorvalue6=analogRead(analogfeedback3);
    //Serial.println(sensorvalue1);
    int potvoltage1=map(sensorvalue1,0,1023,700,4600);
    int potvoltage2=map(sensorvalue2,0,1023,700,4600);
    int potvoltage3=map(sensorvalue3,0,1023,0,4700);
    int voltage4=map(sensorvalue4,109,463,550,2250);
    int voltage5=map(sensorvalue5,109,463,550,2250);
    int voltage6=map(sensorvalue6,109,463,550,2250);
    int presentangle1=map(potvoltage1,700,4600,0,270);
    int presentangle2=map(potvoltage2,700,4600,0,360);
    int presentangle3=map(potvoltage3,0,4700,0,360);
    int presentangle4=map(voltage4,550,2250,0,180);
    int presentangle5=map(voltage5,550,2250,0,180);
    int presentangle6=map(voltage6,550,2250,0,180);
    int error1=0.15*(desiredangle1-presentangle1);
    int error2=0.15*(desiredangle2-presentangle2);
    int error3=0.15*(desiredangle3-presentangle3);
    int error4=gripperangle1-presentangle4;
    int error5=gripperangle2-presentangle5;
    int error6=gripperangle3-presentangle6;
    if(error1>0)
    {
      analogWrite (EN1,150);
      digitalWrite(IN1,LOW);
      digitalWrite(IN2,HIGH);
      delay(5);
      digitalWrite(IN1,HIGH);
      digitalWrite(IN2,HIGH);
      delay(5);
    }
    if(error1<0)
    {
      analogWrite (EN1,150);
      digitalWrite(IN1,HIGH);
      digitalWrite(IN2,LOW);
      delay(5);
      digitalWrite(IN1,HIGH);
```

```
     digitalWrite(IN2,HIGH);
     delay(5);
   }
   if(error2>0)
   {
     analogWrite (EN2,150);
     digitalWrite(IN3,LOW);
     digitalWrite(IN4,HIGH);
     delay(5);
     digitalWrite(IN3,HIGH);
     digitalWrite(IN4,HIGH);
     delay(5);
   }
   if(error2<0)
   {
     analogWrite (EN2,150);
     digitalWrite(IN3,HIGH);
     digitalWrite(IN4,LOW);
     delay(5);
     digitalWrite(IN3,HIGH);
     digitalWrite(IN4,HIGH);
     delay(5);
   }
     if(error3>0)
   {
     analogWrite (EN3,150);
     digitalWrite(IN5,LOW);
     digitalWrite(IN6,HIGH);
     delay(5);
     digitalWrite(IN5,HIGH);
     digitalWrite(IN6,HIGH);
     delay(5);
   }
   if(error3<0)
   {
     analogWrite (EN3,150);
     digitalWrite(IN5,HIGH);
     digitalWrite(IN6,LOW);
     delay(5);
     digitalWrite(IN5,HIGH);
     digitalWrite(IN6,HIGH);
     delay(5);
   }
   myservo1.write(gripperangle1);
   myservo2.write(gripperangle2);
   myservo3.write(gripperangle3);
}
```

ARM MANIPULATORS

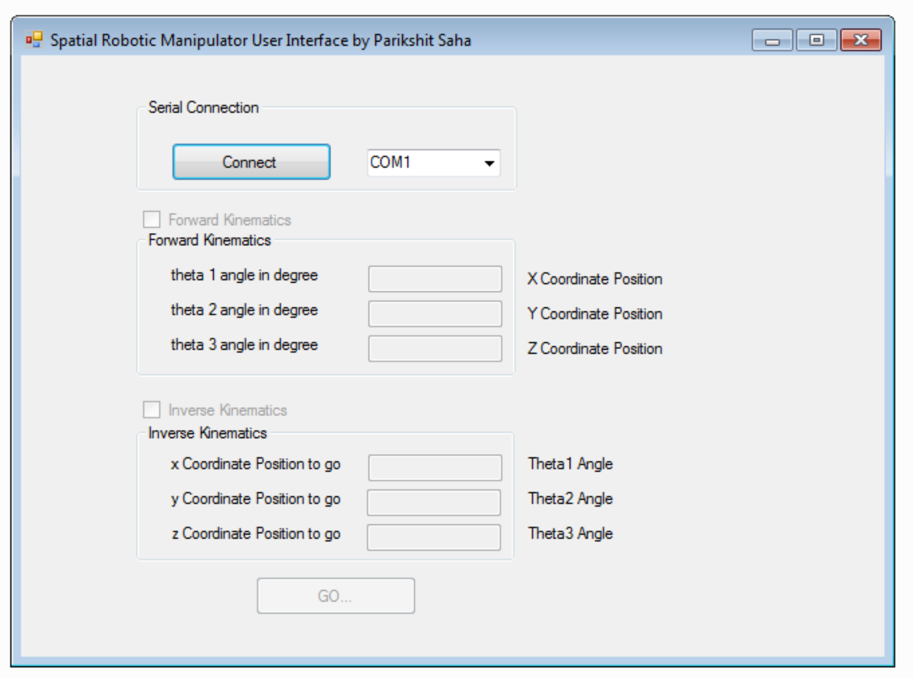
```
#include <Servo.h>
Servo myservo1,myservo2;
void setup() {
  Serial.begin(9600);
  myservo1.attach(2);
  myservo2.attach(4);
}

void loop() {
  if (Serial.available() > 0)
  {
    String theta1degree = Serial.readStringUntil(',');
    Serial.read();
    String theta2degree = Serial.readStringUntil('\0');
    int theta1 = theta1degree.toInt();
    int theta2 = theta2degree.toInt();
    Serial.print("theta1 is ");
    Serial.println(theta1);
    Serial.print("theta2 is ");
    Serial.println(theta2);
    myservo1.write(theta1);
    myservo2.write(theta2);
  }

}
```



2 DOF Robotic Manipulator User Interface designed by Parikshit Saha

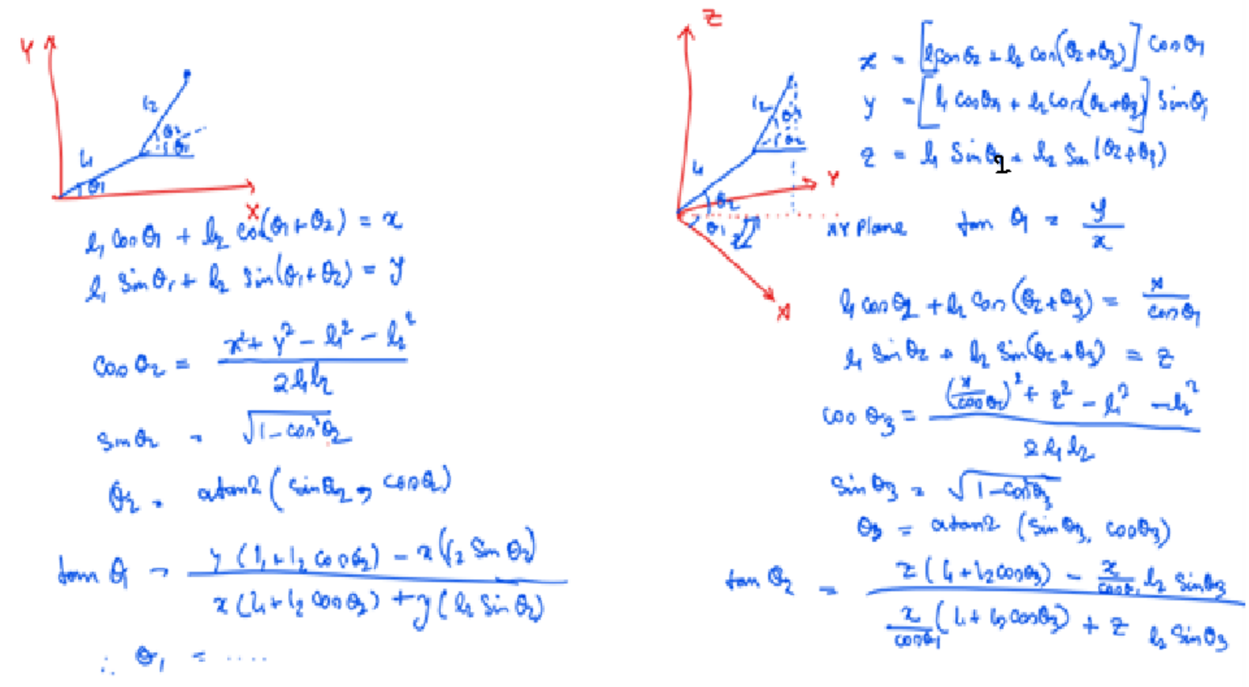
ARM MANIPULATORS

```
#include <Servo.h>
Servo myservo1, myservo2, myservo3;

void setup() {
  myservo1.attach(5);
  myservo2.attach(6);
  myservo3.attach(7);
  Serial.begin(9600);

}

void loop() {
  if (Serial.available() > 0)
  {
    String angle1 = Serial.readStringUntil(',');
    Serial.read();
    String angle2 = Serial.readStringUntil(',');
    Serial.read();
    String angle3 = Serial.readStringUntil('\0');
    int angle1int = angle1.toInt();
    int angle2int = angle2.toInt();
    int angle3int = angle3.toInt();
    Serial.println(angle1int);
    Serial.println(angle2int);
    Serial.println(angle3int);
    myservo1.write(angle1int);
    myservo2.write(angle2int);
    myservo3.write(angle3int);
  }
}
```



ARM MANIPULATORS

Calculation for Forward & Inverse Kinematics



$$l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) = x$$

$$l_1 \sin\theta_1 + l_2 \sin(\theta_1+\theta_2) = y$$

$$\cos\theta_2 = \frac{x^2+y^2-l_1^2-l_2^2}{2l_1l_2}$$

$$\sin\theta_2 = \sqrt{1-\cos^2\theta_2}$$

$$\theta_2 = \operatorname{atan2}(\sin\theta_2, \cos\theta_2)$$

$$\tan\theta_1 = \frac{y(l_1+l_2\cos\theta_2) - x(l_2\sin\theta_2)}{x(l_1+l_2\cos\theta_2)+y(l_2\sin\theta_2)}$$

$$\therefore \theta_1 = \dots$$

$$x = [l_1\cos\theta_2 + l_2\cos(\theta_2+\theta_3)]\cos\theta_1$$

$$y = [l_1\cos\theta_2 + l_2\cos(\theta_2+\theta_3)]\sin\theta_1$$

$$z = l_1\sin\theta_2 + l_2\sin(\theta_2+\theta_3)$$

$$\tan\theta_1 = \frac{y}{x}$$

$$l_1\cos\theta_2 + l_2\cos(\theta_2+\theta_3) = \frac{x}{\cos\theta_1}$$

$$l_1\sin\theta_2 + l_2\sin(\theta_2+\theta_3) = z$$

$$\cos\theta_3 = \frac{\left(\frac{x}{\cos\theta_1}\right)^2 + z^2 - l_1^2 - l_2^2}{2l_1l_2}$$

$$\sin\theta_3 = \sqrt{1-\cos^2\theta_3}$$

$$\theta_3 = \operatorname{atan2}(\sin\theta_3, \cos\theta_3)$$

$$\tan\theta_2 = \frac{z(l_1+l_2\cos\theta_3) - \frac{x}{\cos\theta_1} l_2\sin\theta_3}{\frac{x}{\cos\theta_1}(l_1+l_2\cos\theta_3) + z\, l_2\sin\theta_3}$$

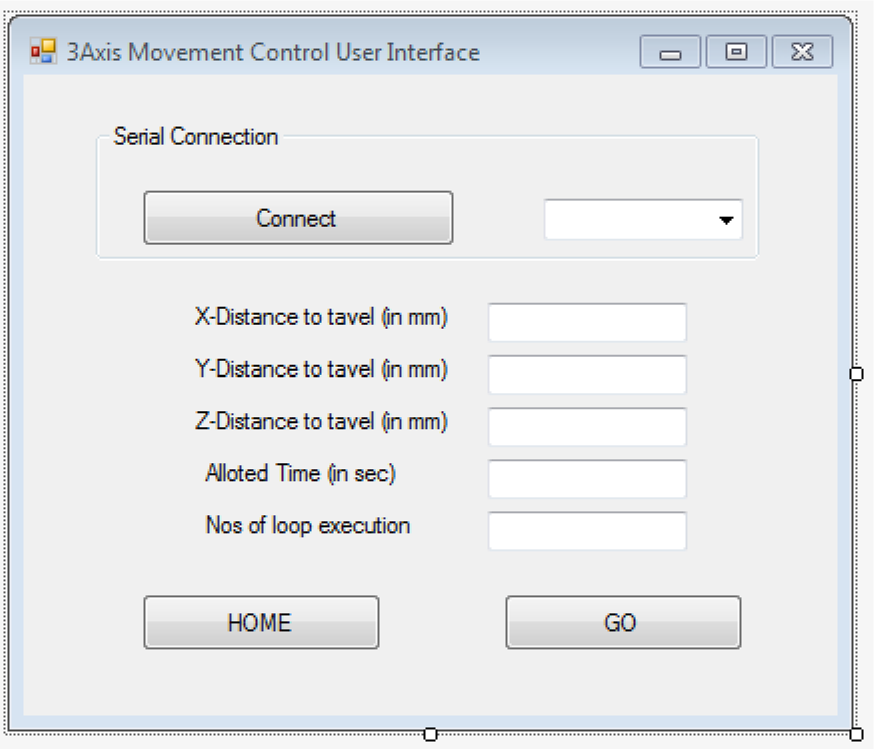
These mathematical formulas are hidden under the GUI and only directly the angles needed to move the servos are directly feed to the Arduino through serial UART Communication.

CODE FORMAT FOR CREATING THE GUI:-

We have used C# language & using visual studio made the above User Interfaces.

ARM MANIPULATORS

A Sample GUI for ARDUINO Interfacing & It’s C# code is given below. Once anyone is familiar with C# programming with Visual studio, it’ll be clear to them.


3Axis Movement Control User Interface
Serial Connection
Connect
X-Distance to tavel (in mm)
Y-Distance to tavel (in mm)
Z-Distance to tavel (in mm)
Alloted Time (in sec)
Nos of loop execution
HOME
GO

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;

namespace userinterface
{
    public partial class userinterface : Form
    {
        bool isConnected = false;
        String[] ports;
        SerialPort port;
        public userinterface()
        {
            InitializeComponent();
            disableControls();
            getAvailableComPorts();

            foreach (string port in ports)
            {
                comboBox1.Items.Add(port);
                Console.WriteLine(port);
                if (ports[0] != null)
                {
                    comboBox1.SelectedItem = ports[0];
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (!isConnected)
            {
                connectToArduino();
            }
            else
            {
                disconnectFromArduino();
            }
        }
        void getAvailableComPorts()
        {
            ports = SerialPort.GetPortNames();
        }

        private void connectToArduino()
        {
            isConnected = true;
            string selectedPort =
comboBox1.GetItemText(comboBox1.SelectedItem);
            port = new SerialPort(selectedPort, 9600, Parity.None, 8,
StopBits.One);
            port.Open();
            button1.Text = "Disconnect";
            enableControls();
        }

        private void disconnectFromArduino()
        {
            isConnected = false;;
            port.Close();
            button1.Text = "Connect";
            disableControls();
        }
        private void enableControls()
        {
            textBox1.Enabled = true;
            textBox3.Enabled = true;
            textBox4.Enabled = true;
            textBox2.Enabled = true;
            textBox5.Enabled = true;
            button2.Enabled = true;
            button3.Enabled = true;
        }

        private void disableControls()
        {
            textBox1.Enabled = false;
            textBox3.Enabled = false;
            textBox4.Enabled = false;
            textBox2.Enabled = false;
            textBox5.Enabled = false;
            button2.Enabled = false;
            button3.Enabled = false;
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void label2_Click(object sender, EventArgs e)
        {

        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }


        private void comboBox1_SelectedIndexChanged(object sender, EventArgs
e)
        {

        }

        private void groupBox1_Enter(object sender, EventArgs e)
        {

        }

        private void textBox1_TextChanged_1(object sender, EventArgs e)
        {

        }

        private void userinterface_Load(object sender, EventArgs e)
        {

        }

        private void Label7_Click(object sender, EventArgs e)
        {

        }

        private void Label5_Click(object sender, EventArgs e)
        {

        }
        private void Label6_Click(object sender, EventArgs e)
        {

        }

        private void TextBox6_TextChanged(object sender, EventArgs e)
        {

        }


        private void button2_Click(object sender, EventArgs e)
        {
            if (isConnected)
            {
                port.Write(textBox2.Text + "," + textBox1.Text + "," +
textBox3.Text + "," + textBox4.Text + "," +textBox5.Text);
            }
        }

        private void label5_Click_1(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)
        {
            if (isConnected)
            {
                port.Write("HOME");
            }
        }
    }
}
```