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

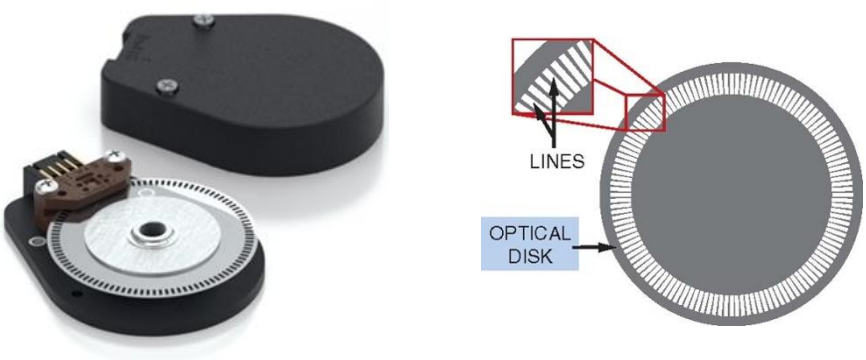


Figure 1 : Incremental and Quadrature Encoder

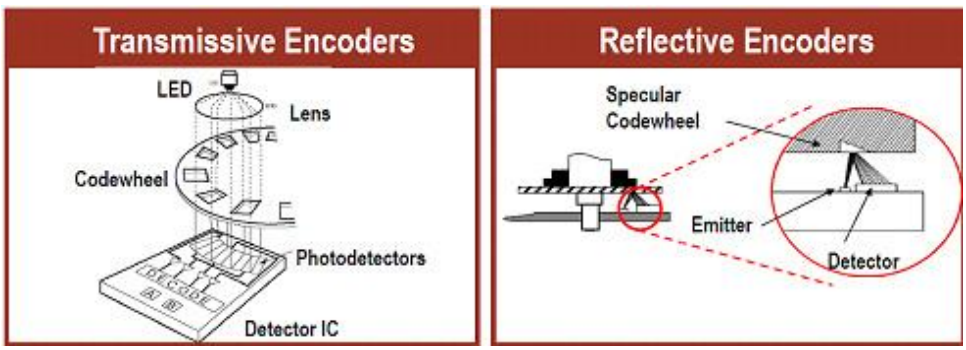


Figure 2: Types of Encoders

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?

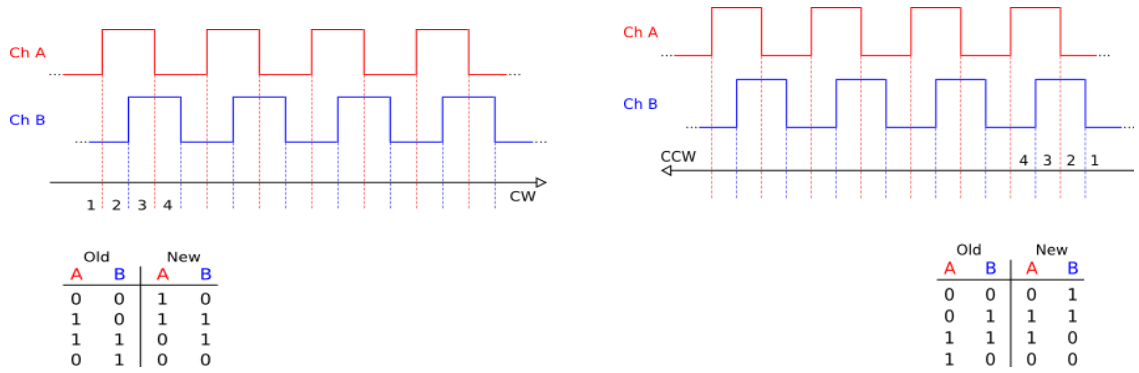


Figure 3: Two square waves in quadrature.

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.

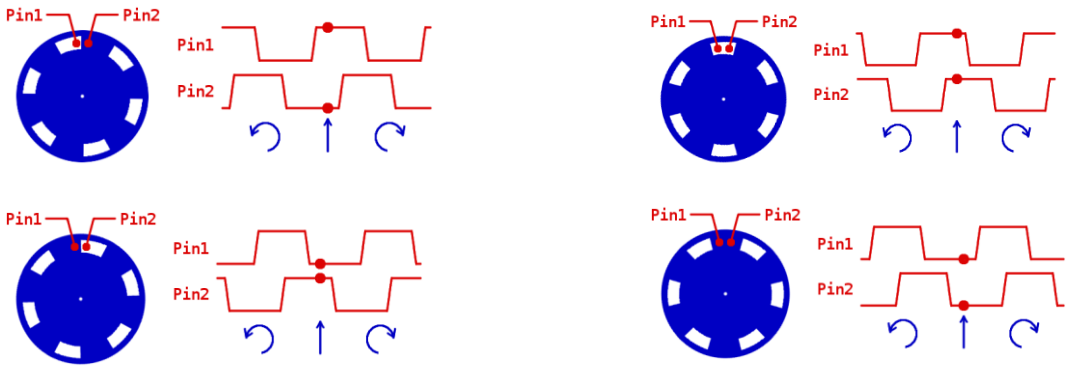


Figure 4: Rotary encoder, with corresponding channel A/B signal states shown on the right

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

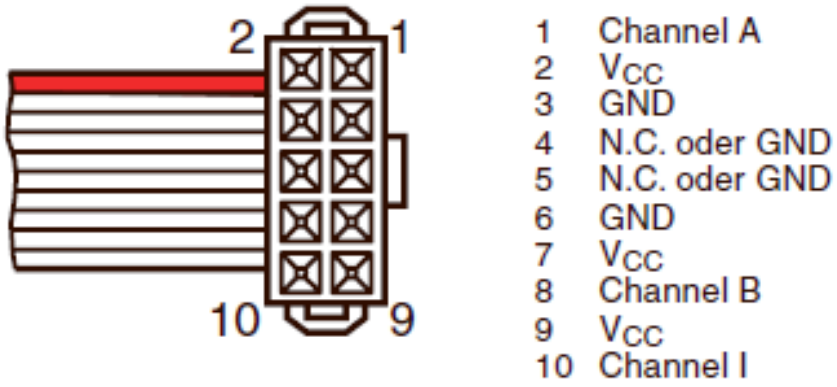


***Figure 5**:* HEDS-6540 Encoder Connector Pin Connections

Calculation:

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

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

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

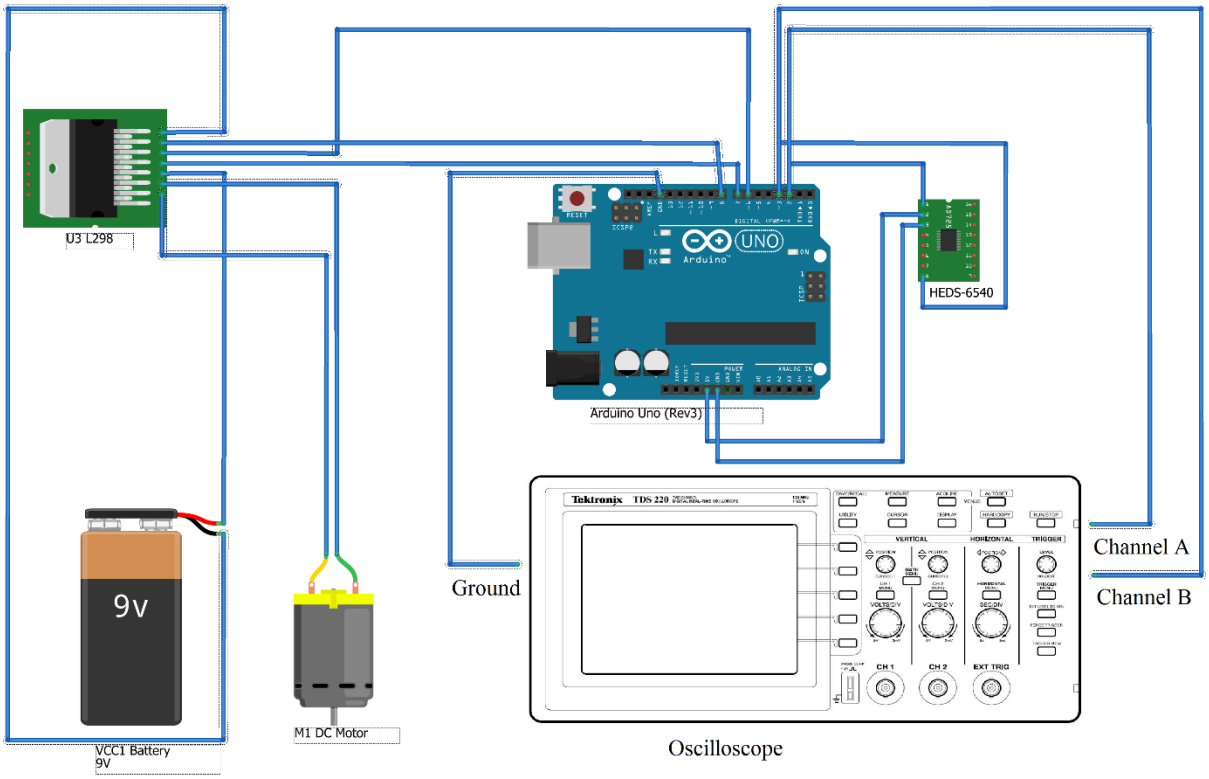


Figure 6 : Schematic diagram of circuit

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);
      }
}
}
}
```

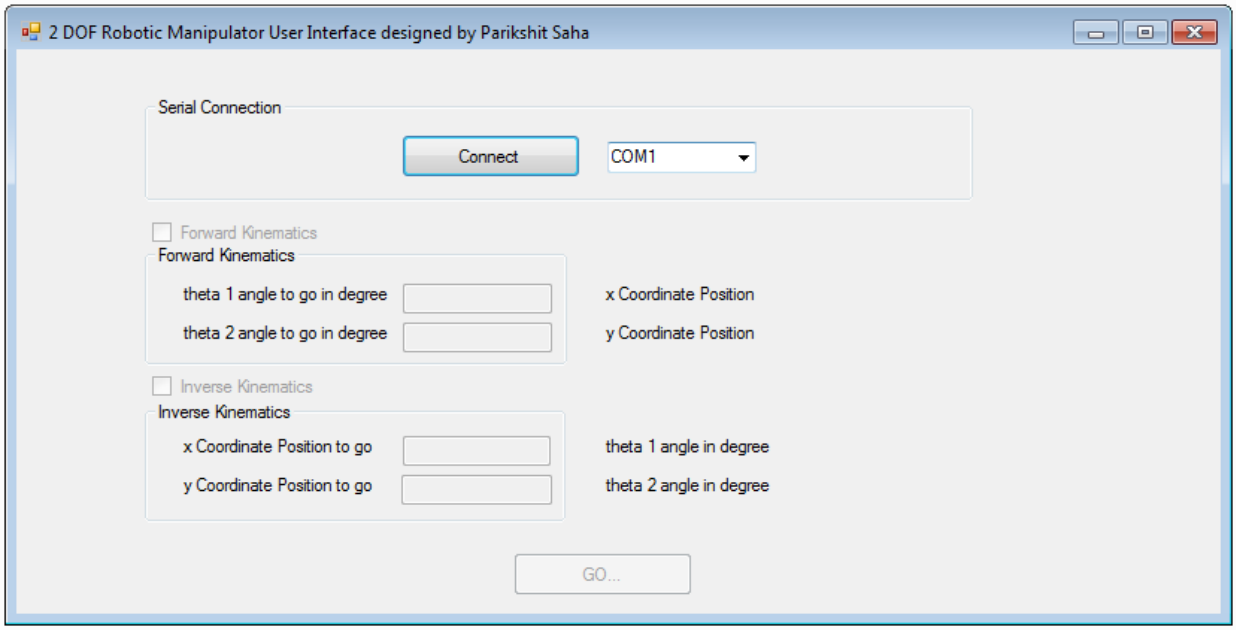
Study the given DC motor plus incremental encoder circuit. Go through the program of microcontroller and understand how it used to activate motor for various applications.

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.

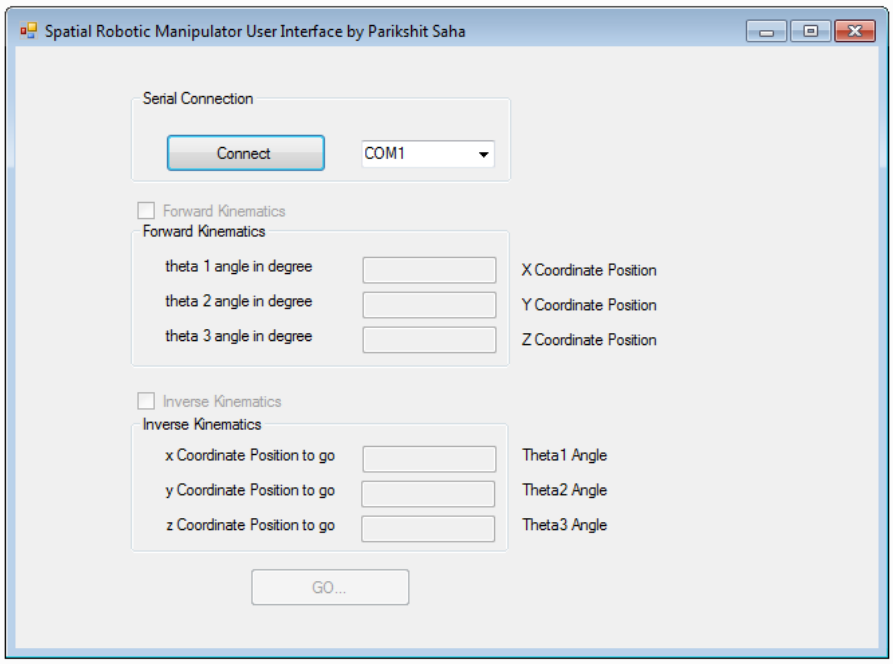
ARM MANIPULATORS

```
#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);
  }

}
```



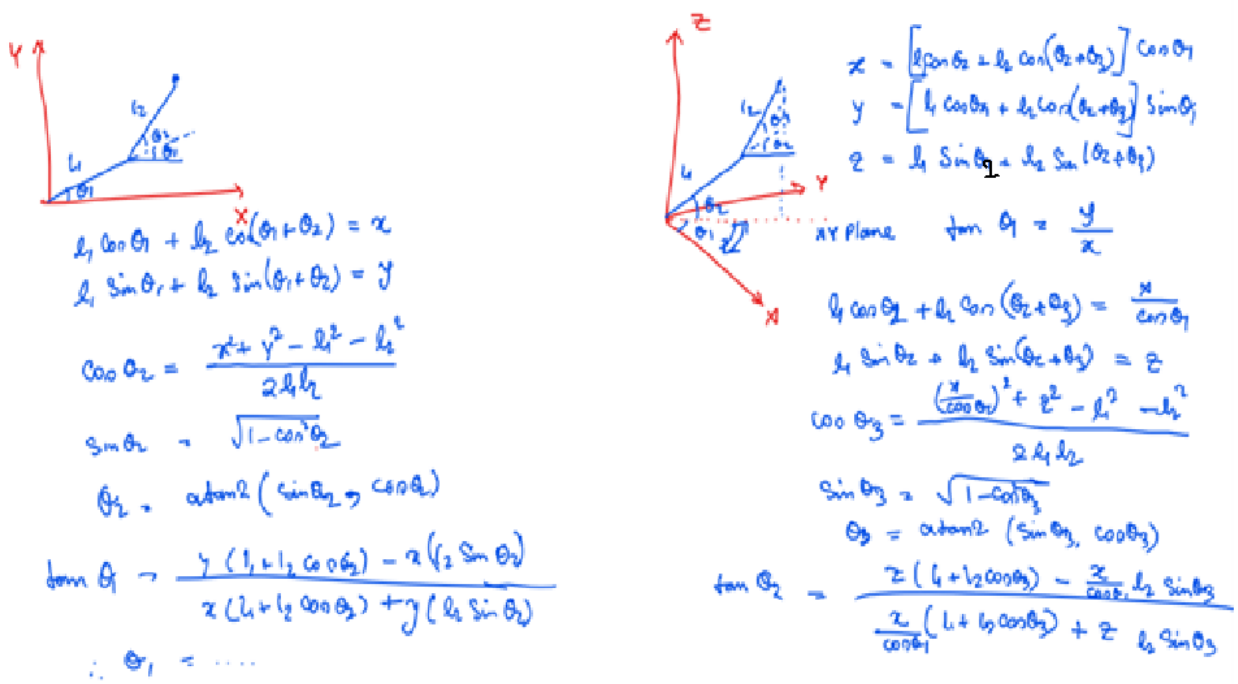
ARM MANIPULATORS

```
#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



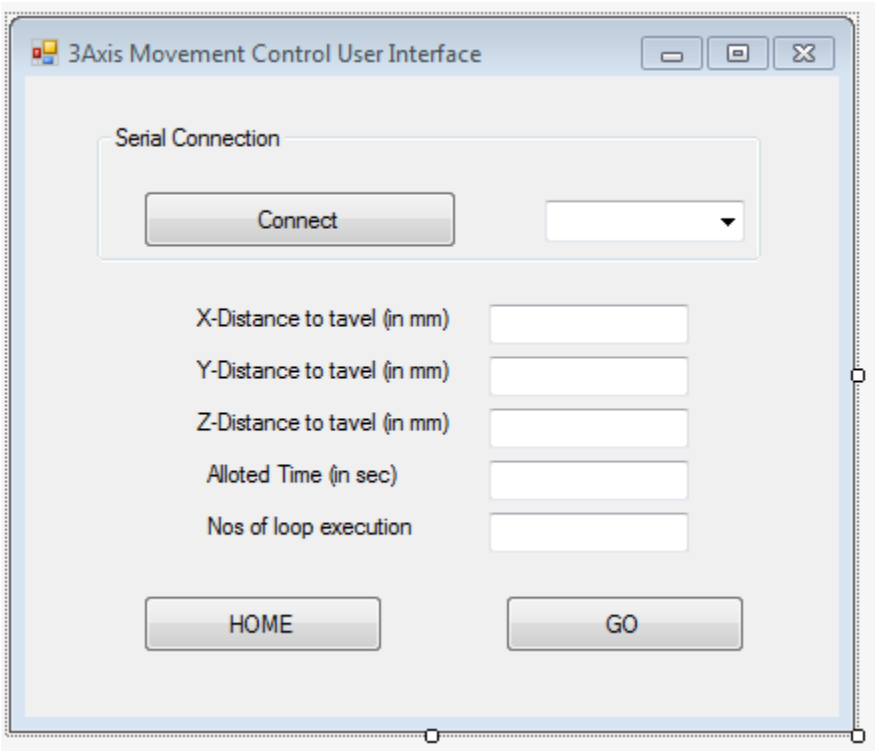
These mathematical formulas are hidden under the GUI and only directly the angles needed to move the servos are directly feed to the Arduino through serial UART Communication.

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.



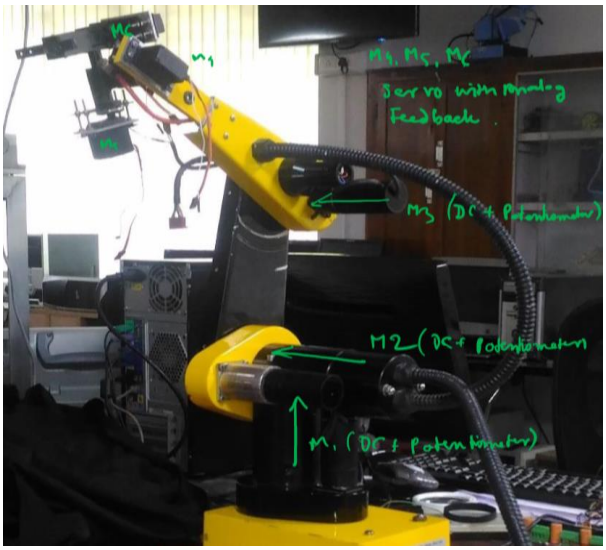

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;

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");
            }
        }
    }
}
```

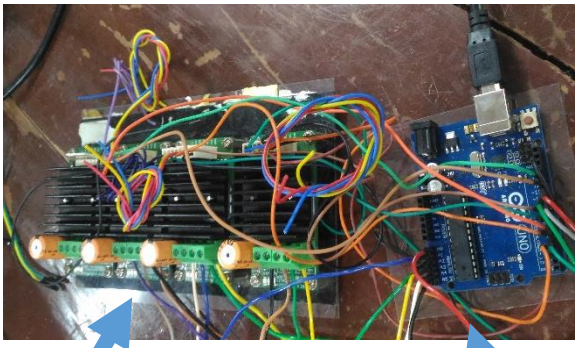
ACTIVE ROBOTIC ARM BLOCK DIAGRAM DESCRIBING WORKING PRINCIPLE



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



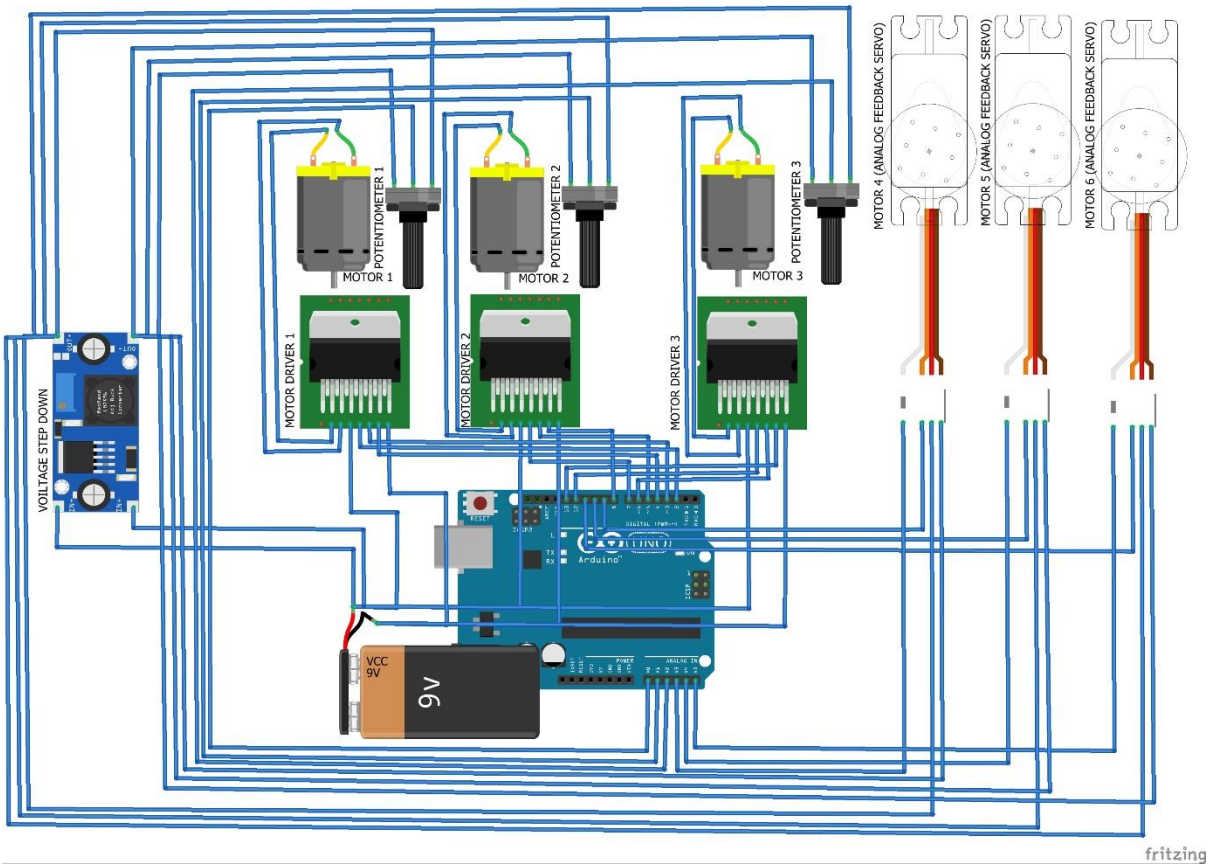
POWER SUPPLY CONNECTION FOR FEEDING IN MOTOR THROUGH DRIVER CIRCUIT

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);
```

ACTIVE ROBOTIC ARM BLOCK DIAGRAM DESCRIBING WORKING PRINCIPLE

```
    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);
}
```

IR based Line Follower:

- Based on the condition on the 3 array line follower IR sensor, the motor is controlled by ARDUINO board through a motor driver (LD298).
- Thus the mobile robot can be made to follow a user defined path by rotating the right and left motors either in the same direction or in different directions as per the condition on the 3 sensors (i.e., Right, Center & Left sensors

IR Module:

Working Principle

IR Sensors work by using a specific light sensor to detect a select light wavelength in the Infra-Red (IR) spectrum.

By using an LED which produces light at the same wavelength as what the sensor is looking for, you can look at the intensity of the received light

When an object is close to the sensor, the light from the LED bounces off the object and into the light sensor. This results in a large jump in the intensity, which can be detected using a threshold.

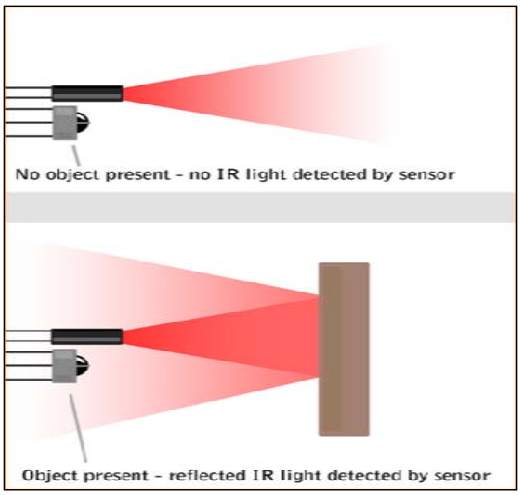


Fig1. IR Emission and detection process

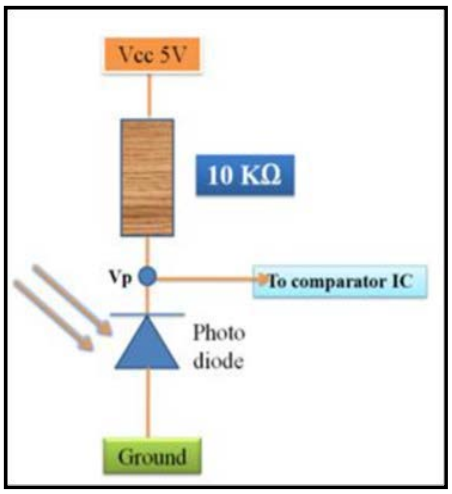


Fig.2 Resistance change detection

If we have white surface it reflects the light and will be sensed by the receiver, similarly if we have black surface it absorbs the light and receiver cannot sense light.

Photo diode has property that if IR light fall on it, its electrical resistance comes down. For sensing the change in resistance voltage divider circuit is used as shown in Fig 9.

Sensor Line Following Array

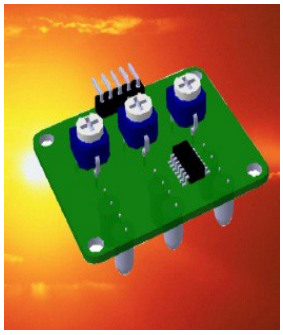


Fig. 3 Three sensor array

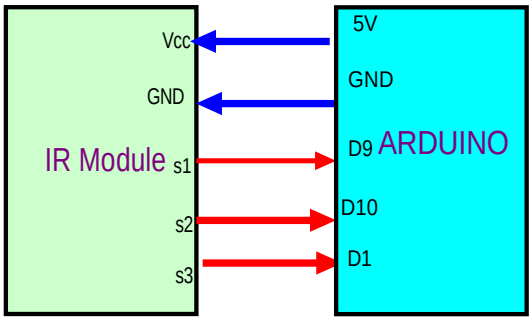


Fig. 4 Typical Interface with ARDUINO

- This is an infrared based sensor array which can be used in basic line following and grid navigation robots.
- The array has 3 individual sensors placed next to each other.
- Each sensor has its own digital output (High or Low) and can sense the presence of a line and indicate it with a 5V logic output.

Features

- Uses 3 sensors for good resolution and simplicity
- Great useful in building basic line following and grid navigating robots
- Input Voltage: 5V DC
- Comes with easy to use digital outputs that can be connected directly to microcontrollers
- The array has mounting holes of 3mm diameter for easy mounting.

➢ The digital outputs from the sensor are directly supplied to the ARDUINO so that based on the conditions on the three sensors, output devices, such as motor can be controlled.

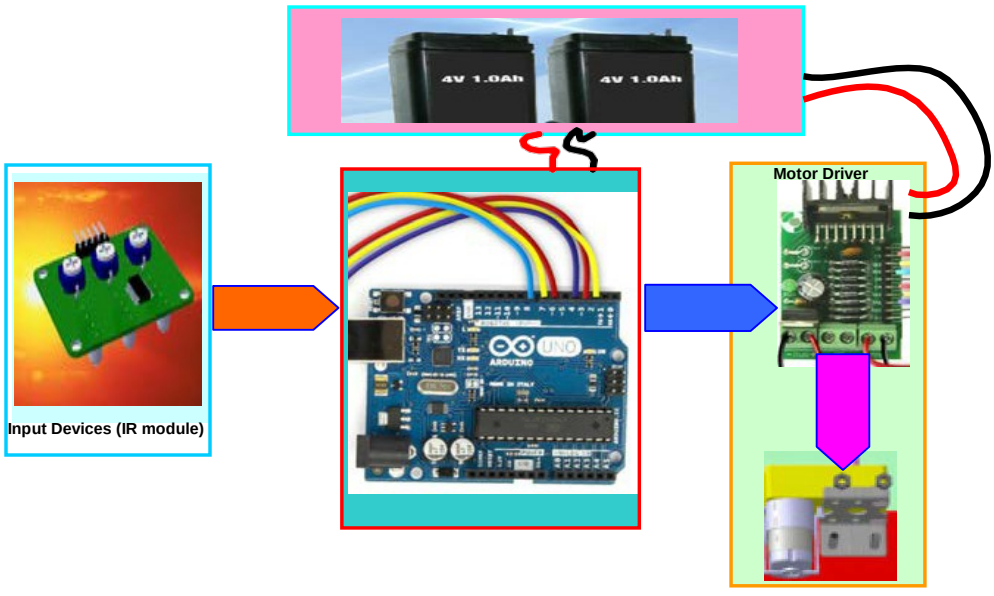


Fig. 5 General block Diagram

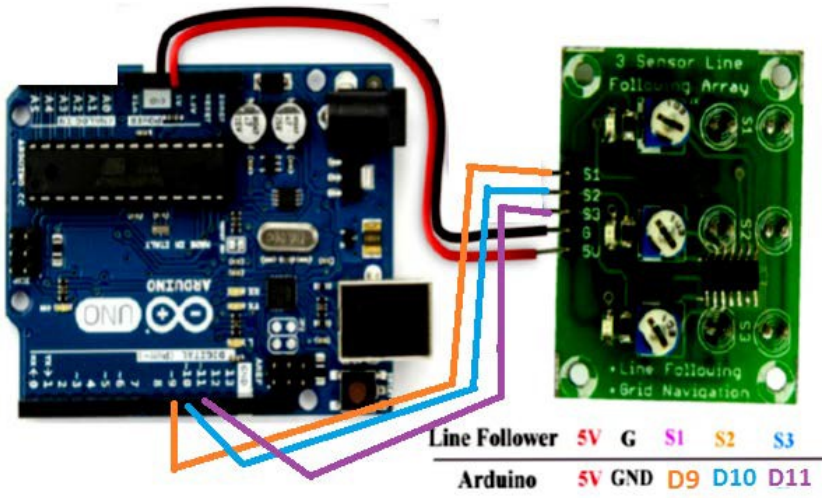
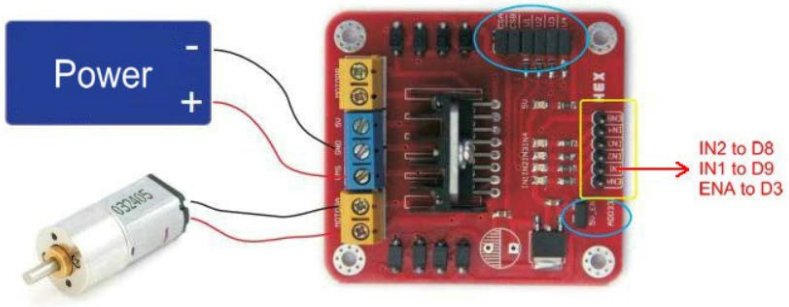


Fig.6 IR Module connection with Arduino

Connection of Line Follower with Arduino

Line follower module	Arduino
S1	9
S2	10
S3	11
GND	GND
5 V	5 V

Motor Connection:-



Sample program:

```
int IN1=8;
int IN2=9;
int ENA=3;
void setup()
{
 pinMode(IN1,OUTPUT);
 pinMode(IN2,OUTPUT);
}
void loop()
{
 analogWrite(ENA, 200);// motor speed
 digitalWrite(IN1,LOW);// rotate forward
 digitalWrite(IN2,HIGH);
 delay(2000);
 digitalWrite(IN1,HIGH);// rotate reverse
 digitalWrite(IN2,LOW);
 delay(2000);
}
```

Fig.7 Motor Control set up

- You should be able to change motor speed and direction of motor at this stage

Quick Connection of Motors

User setup for motor driver through Arduino UNO Board

- Connect battery negative (-ve) to the negative (-ve) pin of motor driver.
- Connect battery positive (+ve) to the positive (+ve) pin of motor driver.
- Motor A(MA)-> left motor
- Motor B(MB)-> right motor
- Connect motor to motor driver

-ve	+ve	Down	Up	Down	Up
PWR		MB		MA	

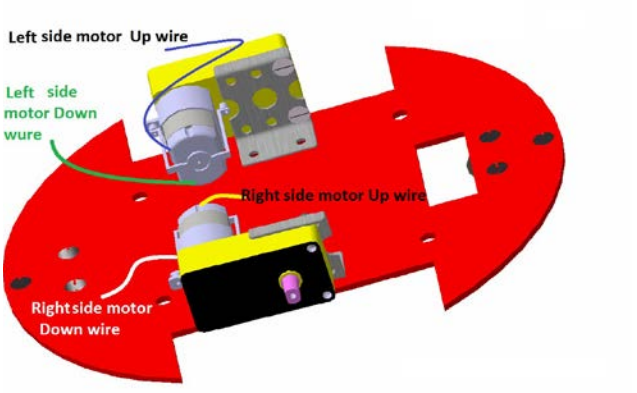


Fig. 8 Wheel Motor Positions

- Connect

Motor driver pin	to	Arduino digital port
IA1		4
IA2		2
EA		3
IB1		8
IB2		7
EB		6

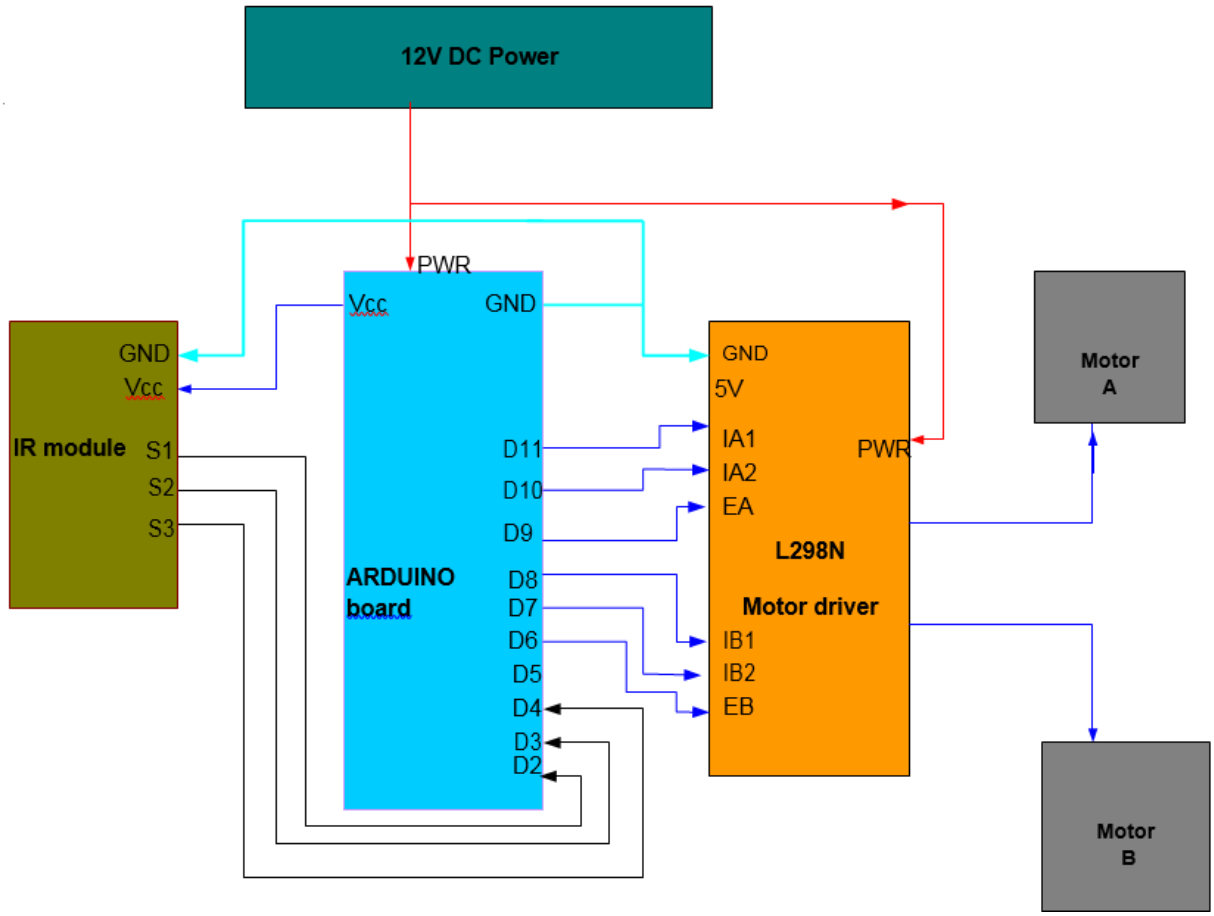


Fig. 9 Overall Block Diagram

Arduino Code:

```
  int val1,val2,val3;
  int m0_enable=3,m1_dir1=2,m1_dir2=4;//motor1 3pin pwm //Left side motor
  int m1_enable=6,m0_dir1=7,m0_dir2=8;//motor2 6 pin pwm// right side motor

  int spid=240;//motor speed
  int S1_pin = 9, S2_pin = 10, S3_pin = 11;
  int caliVal = 1;
  int left=0,right=0;
 void setup() {

Serial.begin(9600); // Start serial communication at 9600 bps

  pinMode(m0_enable, OUTPUT); // Set pin as OUTPUT
  pinMode(m0_dir1, OUTPUT);
  pinMode(m0_dir2, OUTPUT);

  pinMode(m1_enable, OUTPUT);
  pinMode(m1_dir1, OUTPUT);
  pinMode(m1_dir2, OUTPUT);
  analogWrite(m1_enable, spid);
  analogWrite(m0_enable, spid);

  pinMode(S1_pin, INPUT);
  pinMode(S2_pin, INPUT);
  pinMode(S3_pin, INPUT);
```

```
}

void loop() {
///S3 S2 S1
  //front
  //^
  //|
  //v
  //back

val1 = digitalRead(S1_pin);    // read the input pin
val2 = digitalRead(S2_pin);    // read the input pin
val3 = digitalRead(S3_pin);    // read the input pin



if(val1>=caliVal & val2<caliVal & val3>=caliVal){//forward 101
 //if(val2>=caliVal){

 digitalWrite(m0_dir1, HIGH);
 digitalWrite(m0_dir2, LOW);
 digitalWrite(m1_dir1, HIGH);
 digitalWrite(m1_dir2, LOW);

 }
 else if(val1>=caliVal & val2<caliVal & val3<caliVal){//Left 100

 digitalWrite(m0_dir1, HIGH);
 digitalWrite(m0_dir2, LOW);
 digitalWrite(m1_dir1, LOW);
 digitalWrite(m1_dir2, LOW);
 left =1;
 right =0;
 }
 else if(val1>=caliVal & val2>=caliVal & val3<caliVal){//Left 110
 digitalWrite(m0_dir1, HIGH);
 digitalWrite(m0_dir2, LOW);
 digitalWrite(m1_dir1, LOW);
 digitalWrite(m1_dir2, LOW);

 left =1;
 right =0;
 }
 else if(val1<caliVal & val2<caliVal & val3>=caliVal){//Right 001
 digitalWrite(m0_dir1, LOW);
 digitalWrite(m0_dir2, LOW);
 digitalWrite(m1_dir1, HIGH);
 digitalWrite(m1_dir2, LOW);
 left =0;
 right =1;
 }
 else if(val1<caliVal & val2>=caliVal & val3>=caliVal){//Right 011
 digitalWrite(m0_dir1, LOW);
 digitalWrite(m0_dir2, LOW);
 digitalWrite(m1_dir1, HIGH);
 digitalWrite(m1_dir2, LOW);
 left =0;
 right =1;
 }
```

```
 else if(val1>=caliVal & val2>=caliVal & val3>=caliVal){ //111 when there is
no path
   if(left==1){
 digitalWrite(m0_dir1, HIGH);
 digitalWrite(m0_dir2, LOW);
 digitalWrite(m1_dir1, LOW);
 digitalWrite(m1_dir2, LOW);

  left =1;
 right =0;
   }if(right==1){
 digitalWrite(m0_dir1, LOW);
 digitalWrite(m0_dir2, LOW);
 digitalWrite(m1_dir1, HIGH);
 digitalWrite(m1_dir2, LOW);
  left =0;
 right =1;
   }
   if(left!=1 & right!=1){
   digitalWrite(m0_dir1, HIGH);
 digitalWrite(m0_dir2, LOW);
 digitalWrite(m1_dir1, HIGH);
 digitalWrite(m1_dir2, LOW);
   }

 }else{
 digitalWrite(m0_dir1, HIGH);
 digitalWrite(m0_dir2, LOW);
 digitalWrite(m1_dir1, HIGH);
 digitalWrite(m1_dir2, LOW);

     }

 }
```

Study the line follower robot and its circuit. Go through the microcontroller code on your computer screen and understand how it is works.

Run the program and see the robot movement. Now do the following things

1. Tell the various strategies fo the speed control of the motor?
2. Tell the logic for searching the path?
3. How you will guid the robot to select between two black lines?
4. How you will change the turning radius of robot?
5. What is the effect of line thickness on robot movement?
6. What is the effect of position of the sensors.

Ultrasonic Sensor based Obstacle Avoidance:

- Here distance of object from the mobile robot is measured by Ultrasonic sensor.
- Depending up on the logic we set, the robot can avoid the object detected either turning to the right or to the left of the detected object.
- In this experiment, we will be learning how to
  - set the Ultrasonic sensor for operation
  - display sensor reading on serial monitor
  - be able to control motors
- The overall block diagram and sample program is provided bellow.

Ultrasonic Sensor:

An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves which vibrates at a frequency above the range of human hearing.

An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity.

High-frequency sound waves reflect from boundaries to produce distinct echo patterns.

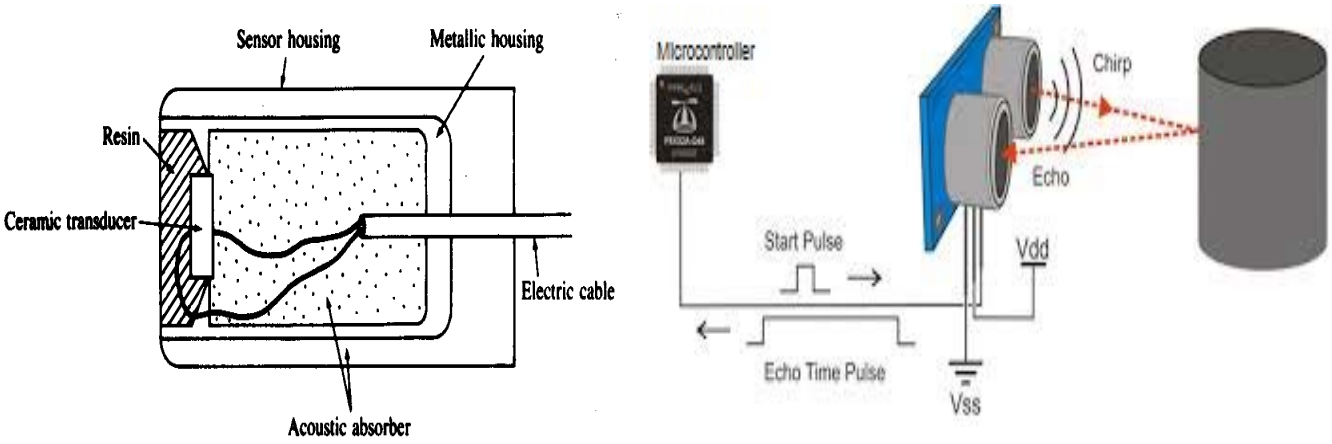


Fig. 1 Internal construction of Ultrasonic sensor and its working principal

Ultrasonic Ranging Module HC - SR04

Product features:

- Ultrasonic ranging module HC - SR04 provides *2cm - 400cm* non-contact measurement function, the ranging accuracy can reach to 3mm. The module includes ultrasonic transmitter, receiver and control circuit.
- The basic principle of work:
  - Using IO ,trigger for at least 10us high level signal,
  - The Module automatically sends eight cycles of *40 kHz* and makes the ECHO

pin HIGH and it detects whether there is a pulse signal back.

- IF there is a signal back, the ECHO pin is made to be LOW.
- Thus the time duration of high level ECHO pulse is the measure of the distance of the object detected.

Test distance = [(ECHO high level duration)×(velocity of sound / 2)]

Test Distance = (ECHO pulse duration) / (58) cm

Fig. 2 Ultrasonic Sensor

- ***Wire connecting direct as following:***
  - 5V Supply
  - Trigger Pulse Input
  - Echo Pulse Output
  - 0V Ground

- ***Timing diagram and Typical Interface with Arduino***

  - The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The period of the Echo pulse is then used for distance calculation

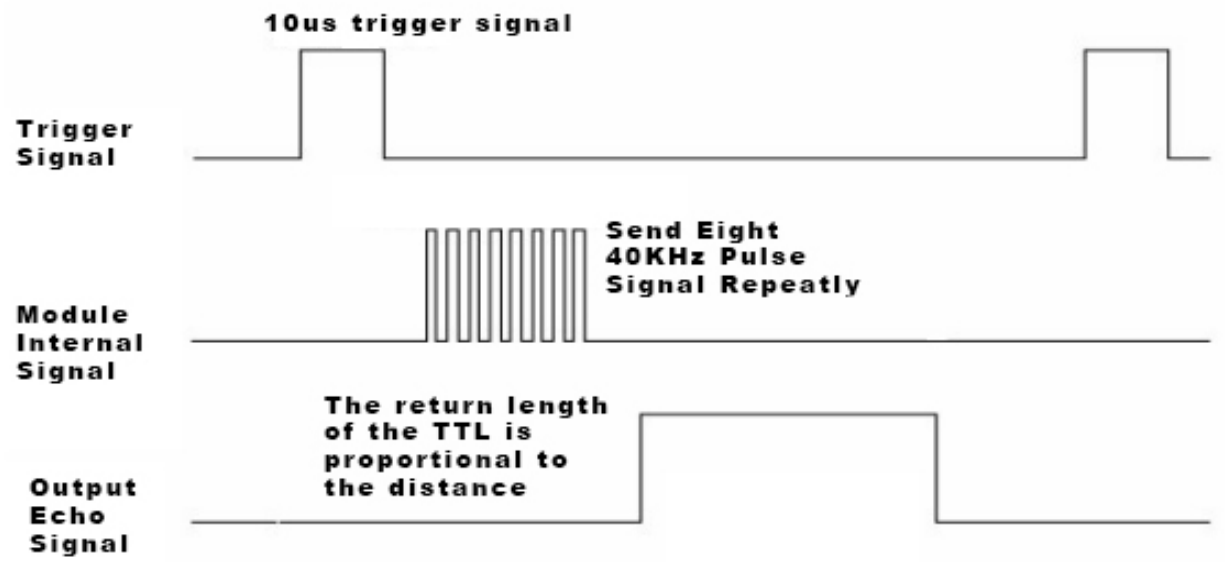


Fig.3 Timing Diagram

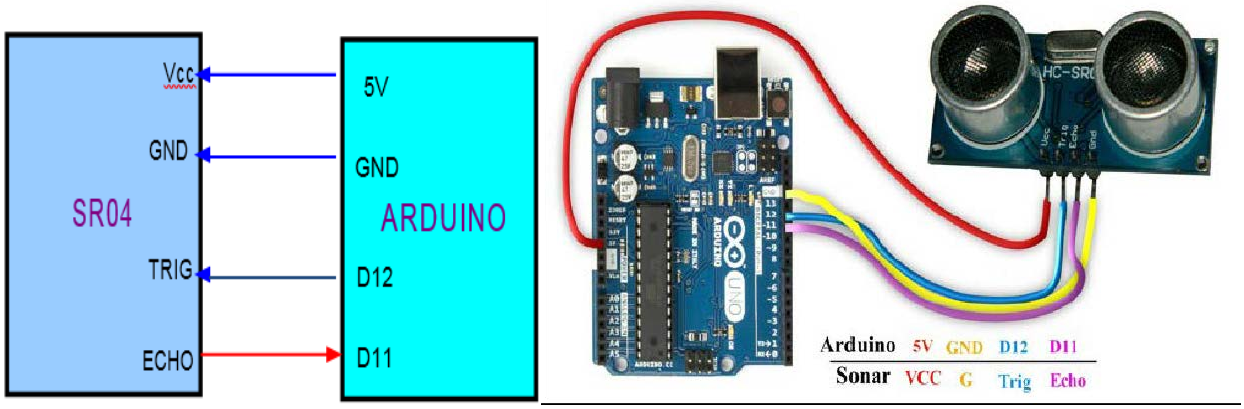


Fig.4 Typical Interface with ARDUINO

Servo Motor:

It is just made up of simple motor which run through **servo mechanism**. If motor is used is DC powered then it is called DC servo motor, and if it is AC powered motor then it is called AC servo motor. We can get a very high torque servo motor in a small and light weight packages.

Servo Mechanism

It consists of three parts:

1. Controlled device
2. Output sensor
3. Feedback system

It is a closed loop system where it uses positive feedback system to control motion and final position of the shaft. Here the device is controlled by a feedback signal generated by comparing output signal and reference input signal.

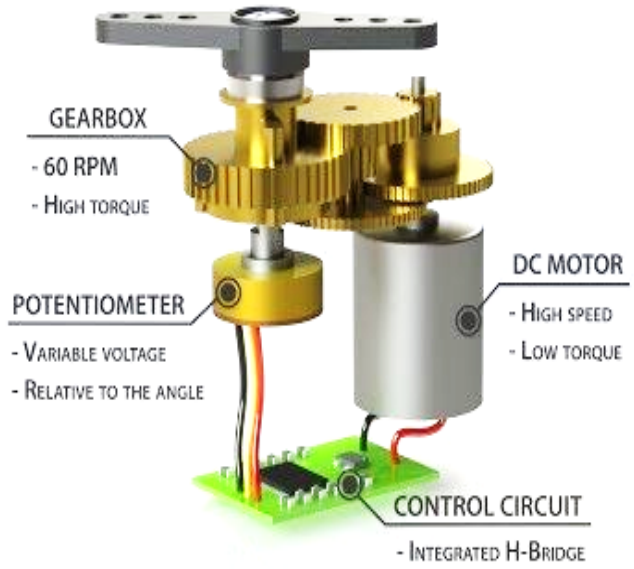


Fig.5 Servo motor internal structure

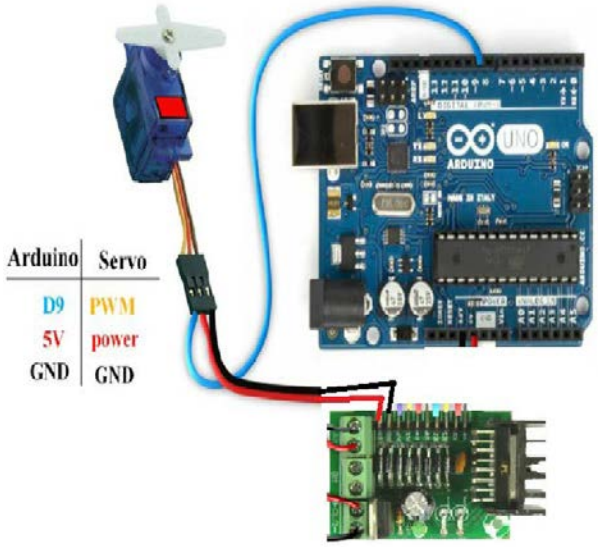
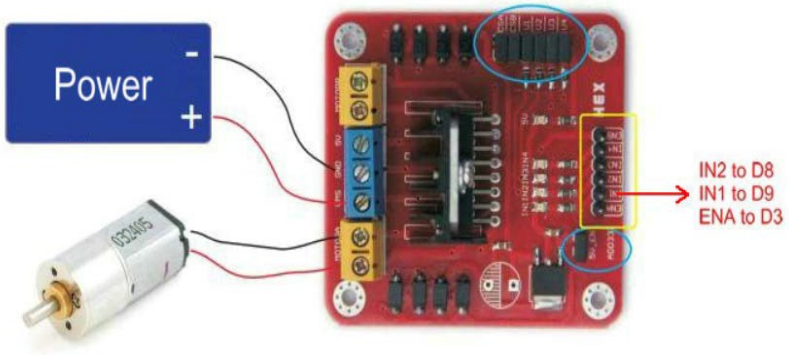


Fig.6 Connection of Servo with microcontroller

Wheel Motor Connections:-

- You should be able to change motor speed and direction of motor at this stage.



Sample program:

```
int IN1=8;
int IN2=9;
int ENA=3;
void setup()
{
 pinMode(IN1,OUTPUT);
 pinMode(IN2,OUTPUT);
}
void loop()
{
 analogWrite(ENA, 200);// motor speed
 digitalWrite(IN1,LOW);// rotate forward
 digitalWrite(IN2,HIGH);
 delay(2000);
 digitalWrite(IN1,HIGH);// rotate reverse
 digitalWrite(IN2,LOW);
 delay(2000);
}
```

Fig.7 Motor Control set up

Quick Connection of Wheel Motors

User setup for motor driver through Arduino UNO Board

- Connect battery negative (-ve) to the negative (-ve) pin of motor driver.
- Connect battery positive (+ve) to the positive (+ve) pin of motor driver.
- Motor A(MA)-> left motor
- Motor B(MB)-> right motor

-ve	+ve	Down	Up	Down	Up
PWR		MB		MA	

- Connect motor to motor driver

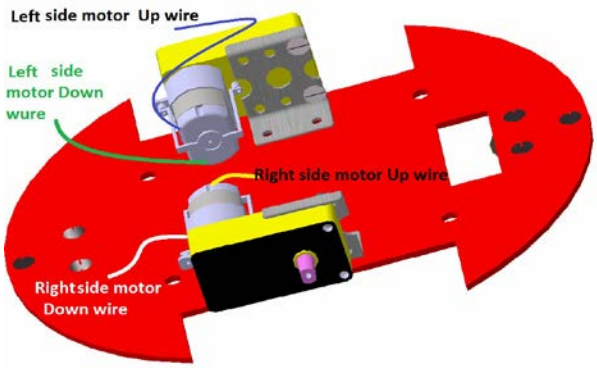


Fig. 8 Wheel Motor Positions

Motor driver pin	to	Arduino digital port
IA1		4
IA2		2
EA		3
IB1		8
IB2		7
EB		6

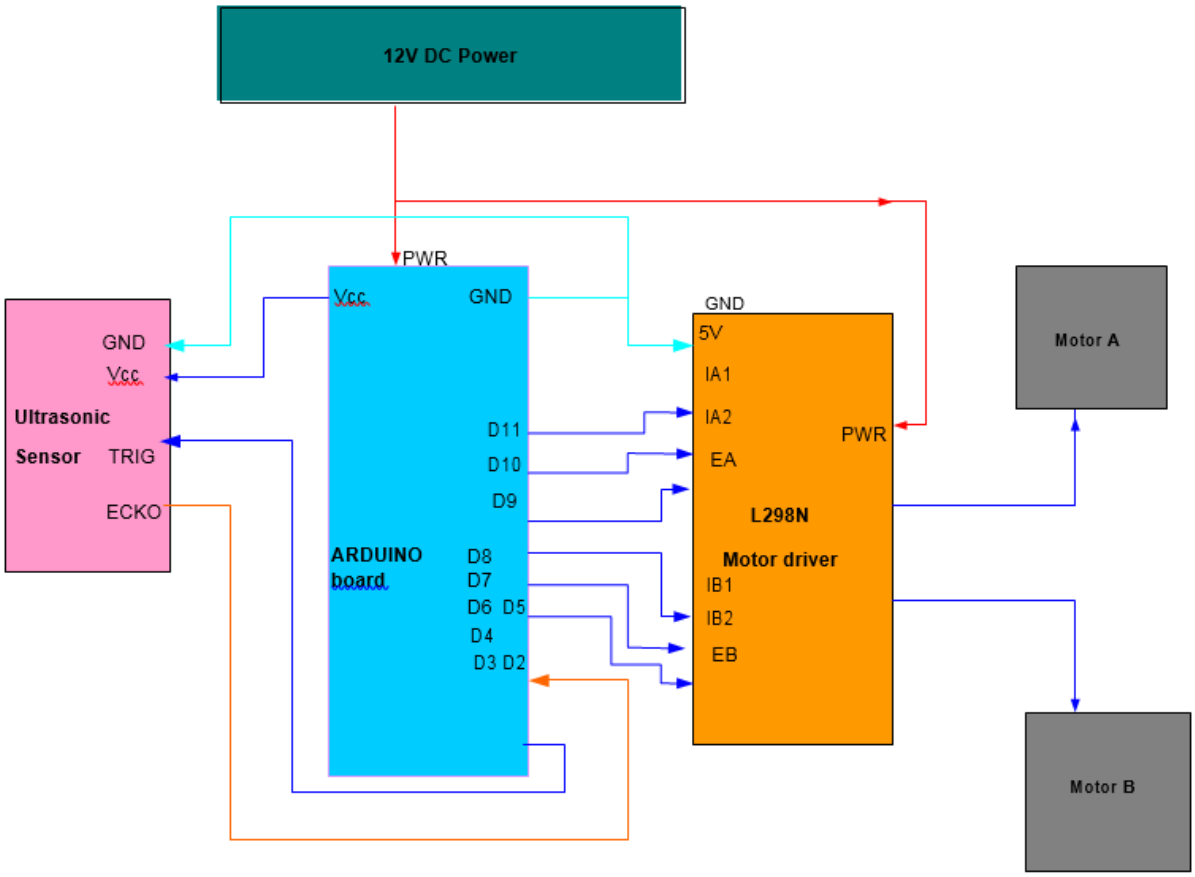


Fig. 9 Block diagram of the overall system

Arduino code

```
const int ECHOPIN = 11;
const int TRIGPIN = 12;
int m0_enable=3,m1_dir1=2,m1_dir2=4;//motor1 3pin pwm //Left side motor
int m1_enable=6,m0_dir1=8,m0_dir2=7;//motor2 6 pin pwm// right side motor
int spid=225;//motor speed
void setup() {
  Serial.begin(9600);
  pinMode(m0_enable, OUTPUT); // Set pin as OUTPUT
  pinMode(m0_dir1, OUTPUT);
  pinMode(m0_dir2, OUTPUT);
  pinMode(m1_enable, OUTPUT);
  pinMode(m1_dir1, OUTPUT);
  pinMode(m1_dir2, OUTPUT);
  analogWrite(m1_enable, spid);
  analogWrite(m0_enable, spid);
  pinMode(ECHOPIN,INPUT);
  pinMode(TRIGPIN,OUTPUT);
}

void loop() {
  digitalWrite( TRIGPIN, LOW);
  delay(2);
  digitalWrite(TRIGPIN, HIGH);
  delay(10);
```

```
  digitalWrite( TRIGPIN, LOW);
  // distance calculation
    float distance = pulseIn(ECHOPIN, HIGH);
  distance = (distance)/(58);
  Serial.print("distance in cm");
  Serial.println(distance);
  delay(200);
  // the distance can be used to control output devices such as motor
  if(distance>10)
    {
   digitalWrite(m0_dir1, HIGH); // forward
   digitalWrite(m0_dir2, LOW);
   digitalWrite(m1_dir1, HIGH);
   digitalWrite(m1_dir2, LOW);
    }
  else if (distance<10)
   {
   int dir = random(10,20);

   digitalWrite(m0_dir1, LOW); // stop for a while
   digitalWrite(m0_dir2, LOW);
   digitalWrite(m1_dir1, LOW);
   digitalWrite(m1_dir2, LOW);
   delay(200);

   if (dir >=15){
   digitalWrite(m0_dir1, HIGH);
   digitalWrite(m0_dir2, LOW);
   digitalWrite(m1_dir1, LOW);
   digitalWrite(m1_dir2, HIGH);
   }
   else{
   digitalWrite(m0_dir1, LOW);
   digitalWrite(m0_dir2,HIGH);
   digitalWrite(m1_dir1, HIGH);
   digitalWrite(m1_dir2, LOW);

   }
   delay(200);
   }
}
```

Distance Map Generation with a Sonar Sensor on an Arduino (Range Scanning):

It uses the propagation of sound to detect objects. Depending on the time that the signal takes to come back to the sonar, it can detect the range or distance of an object and its orientation. It also detects the strength of a signal to determine how much time it took to be picked up by the receiver.

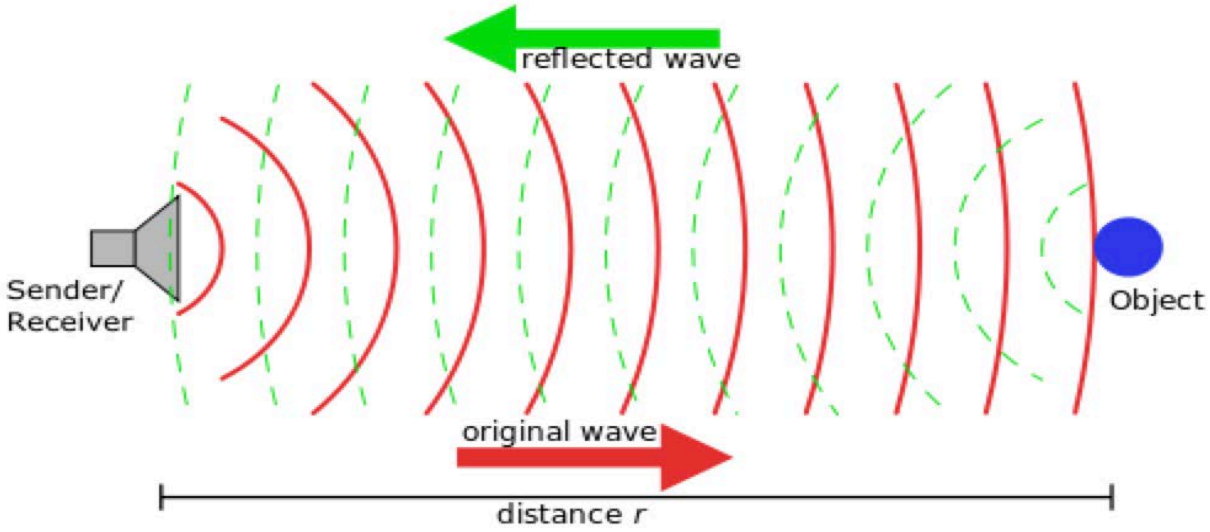


Fig. 10 Working Principal of Sonar

We will be using an Arduino Uno as our microprocessor to be able to read distance detected by the sonar. The sonar that we are using is the Ultrasonic ranging module HC - SR04

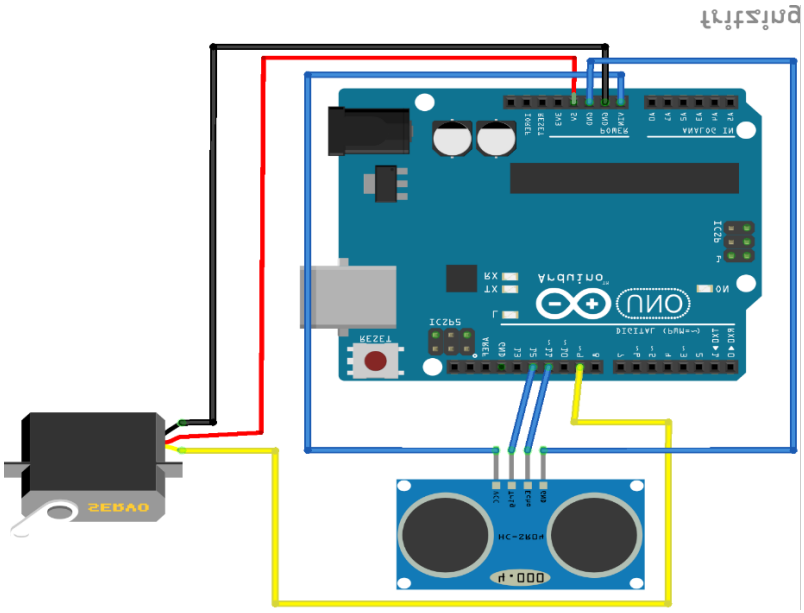


Fig. 11 Block diagram of the Arduino connections

Arduino code

```
// Includes the Servo library
#include <Servo.h>

// Defines Tirg and Echo pins of the Ultrasonic Sensor
const int trigPin = 12;
const int echoPin = 11;
// Variables for the duration and the distance
long duration;
int distance;

Servo myServo; // Creates a servo object for controlling the servo motor

void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600);
  myServo.attach(9); // Defines on which pin is the servo motor attached
}
void loop() {
  // rotates the servo motor from 15 to 165 degrees
  for(int i=10;i<=170;i++){
  myServo.write(i);
  delay(20);
  distance = calculateDistance();// Calls a function for calculating the
distance measured by the Ultrasonic sensor for each degree

  Serial.print(i); // Sends the current degree into the Serial Port
  Serial.print(","); // Sends addition character right next to the previous
value needed later in the Processing IDE for indexing
  Serial.print(duration); // Sends the distance value into the Serial Port
  Serial.print("."); // Sends addition character right next to the previous
value needed later in the Processing IDE for indexing
  }
  // Repeats the previous lines from 165 to 15 degrees
  for(int i=165;i>15;i--){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
}
// Function for calculating the distance measured by the Ultrasonic sensor
int calculateDistance(){

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound
wave travel time in microseconds
  distance= duration*0.034/2;
  return distance;
}
```

Processing code

```
import processing.serial.*; // imports library for serial communication
import java.awt.event.KeyEvent; // imports library for reading the data from
the serial port
import java.io.IOException;

Serial myPort; // defines Object Serial
// defubes variables
String angle="";
String distance="";
String data="";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;

void setup() {

 size (1440,800); // ***CHANGE THIS TO YOUR SCREEN RESOLUTION***
 smooth();
 myPort = new Serial(this,"COM6", 9600); // starts the serial communication
 myPort.bufferUntil('.'); // reads the data from the serial port up to the
character '.'. So actually it reads this: angle,distance.
 //orcFont = loadFont("OCRAExtended-30.vlw");
}

void draw() {

  fill(98,245,31);
  //textFont(orcFont);
  // simulating motion blur and slow fade of the moving line
  noStroke();
  fill(0,4);
  rect(0, 0, width, height-height*0.065);

  fill(98,245,31); // green color
  // calls the functions for drawing the radar
  drawRadar();
  drawLine();
  drawObject();
  drawText();
}

void serialEvent (Serial myPort) { // starts reading data from the Serial
Port
  // reads the data from the Serial Port up to the character '.' and puts it
into the String variable "data".
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);

  index1 = data.indexOf(","); // find the character ',' and puts it into the
variable "index1"
  angle= data.substring(0, index1); // read the data from position "0" to
position of the variable index1 or thats the value of the angle the Arduino
Board sent into the Serial Port
  distance= data.substring(index1+1, data.length()); // read the data from
position "index1" to the end of the data pr thats the value of the distance
```

```
  // converts the String variables into Integer
  iAngle = int(angle);
  iDistance = int(distance);
}

void drawRadar() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to
new location
  noFill();
  strokeWeight(2);
  stroke(98,245,31);
  // draws the arc lines
  arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI);
  arc(0,0,(width-width*0.27),(width-width*0.27),PI,TWO_PI);
  arc(0,0,(width-width*0.479),(width-width*0.479),PI,TWO_PI);
  arc(0,0,(width-width*0.687),(width-width*0.687),PI,TWO_PI);
  // draws the angle lines
  line(-width/2,0,width/2,0);
  line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
  line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
  line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
  line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
  line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));
  line((-width/2)*cos(radians(30)),0,width/2,0);
  popMatrix();
}

void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to
new location
  strokeWeight(9);
  stroke(255,10,10); // red color
  pixsDistance = iDistance*((height-height*0.1666)*0.025); // covers the
distance from the sensor from cm to pixels
  // limiting the range to 40 cms
  if(iDistance<40){
    // draws the object according to the angle and the distance
  line(pixsDistance*cos(radians(iAngle)),-
pixsDistance*sin(radians(iAngle)),(width-width*0.505)*cos(radians(iAngle)),-
(width-width*0.505)*sin(radians(iAngle)));
  }
  popMatrix();
}

void drawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30,250,60);
  translate(width/2,height-height*0.074); // moves the starting coordinats to
new location
  line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-
height*0.12)*sin(radians(iAngle))); // draws the line according to the angle
  popMatrix();
}

void drawText() { // draws the texts on the screen
```

```
  pushMatrix();
  if(iDistance>40) {
  noObject = "Out of Range";
  }
  else {
  noObject = "In Range";
  }
  fill(0,0,0);
  noStroke();
  rect(0, height-height*0.0648, width, height);
  fill(98,245,31);
  textSize(25);

  text("10cm",width-width*0.3854,height-height*0.0833);
  text("20cm",width-width*0.281,height-height*0.0833);
  text("30cm",width-width*0.177,height-height*0.0833);
  text("40cm",width-width*0.0729,height-height*0.0833);
  textSize(20);
  text("Object: " + noObject, width-width*0.875, height-height*0.0277);
  text("Angle: " + iAngle +" °", width-width*0.48, height-height*0.0277);
  text("Distance: ", width-width*0.27, height-height*0.0277);
  if(iDistance<40) {
  text("        " + iDistance +" cm", width-width*0.225, height-
height*0.0277);
  }
  textSize(20);
  fill(98,245,60);
  translate((width-width*0.4994)+width/2*cos(radians(30)),(height-
height*0.0907)-width/2*sin(radians(30)));
  rotate(-radians(-60));
  text("30°",0,0);
  resetMatrix();
  translate((width-width*0.503)+width/2*cos(radians(60)),(height-
height*0.0888)-width/2*sin(radians(60)));
  rotate(-radians(-30));
  text("60°",0,0);
  resetMatrix();
  translate((width-width*0.507)+width/2*cos(radians(90)),(height-
height*0.0833)-width/2*sin(radians(90)));
  rotate(radians(0));
  text("90°",0,0);
  resetMatrix();
  translate(width-width*0.513+width/2*cos(radians(120)),(height-
height*0.07129)-width/2*sin(radians(120)));
  rotate(radians(-30));
  text("120°",0,0);
  resetMatrix();
  translate((width-width*0.5104)+width/2*cos(radians(150)),(height-
height*0.0574)-width/2*sin(radians(150)));
  rotate(radians(-60));
  text("150°",0,0);
  popMatrix();
}
```

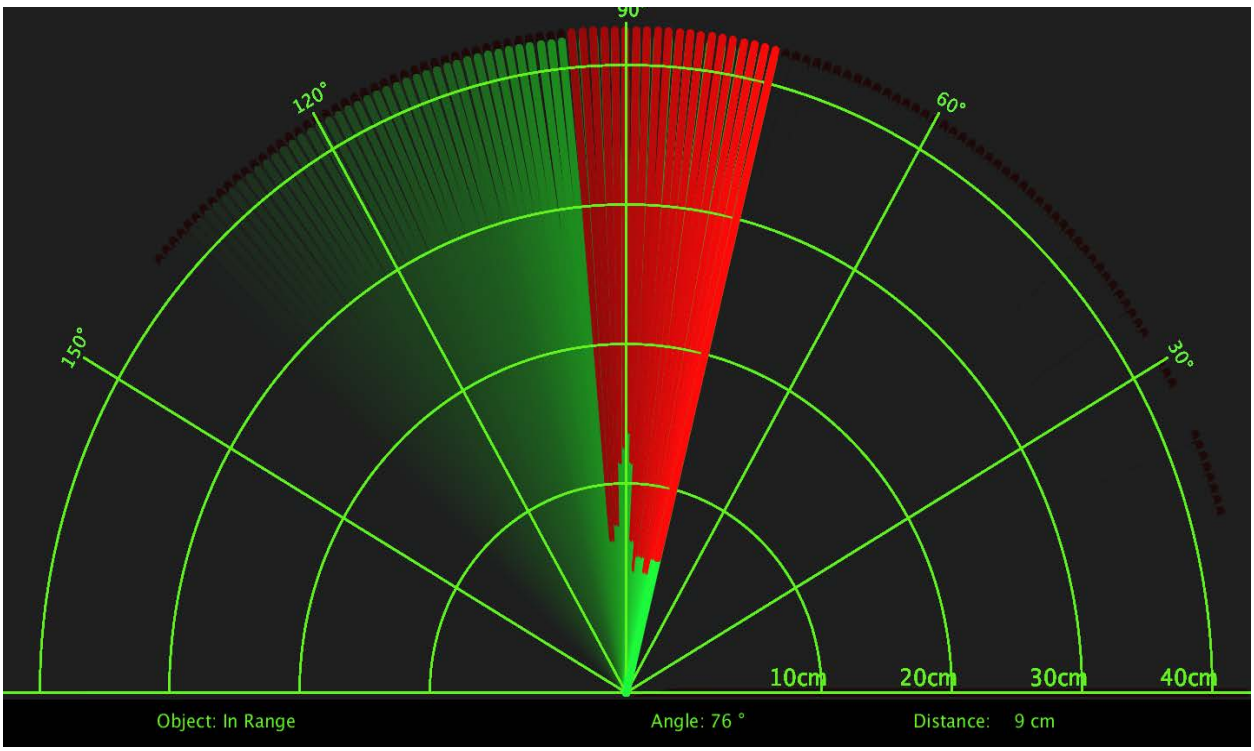


Fig. 12 Sonar Scanning Image

Obstacle avoidance

Study the obstacle avoiding robot and its circuit. Go through the microcontroller code on your computer screen and understand how it is works.

Run the program and see the robot movement. Now do the following things

1. Tell the logic for avoiding the obstacle.
2. Change the speed of robot and see how it avoids the obstacles.
3. Set various range of avoidance distance and see how robot behaves.

Sonar

1. Vary scanning angles on sonar and see its effect on object detection.
2. Change in the speed of servo motor while scanning and see the reading in sonar. Tell your observations.
3. Place various objects in front of sonar and see the reading in the sonar screen. Tell your observations.

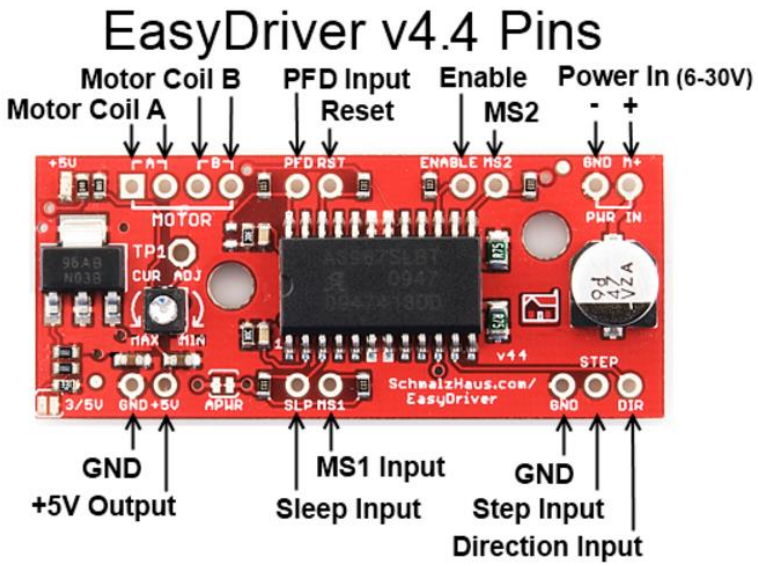
STEPPER MOTOR CONTROL WITH ARDUINO

Basic Arduino setup with Easy Driver

Connect the motor's four wires to the Easy Driver (note the proper coil connections), connect a power supply of 12V is to the Power In pins, and connect the Arduino's GND, pin 8 and pin 9 to the Easy Driver.

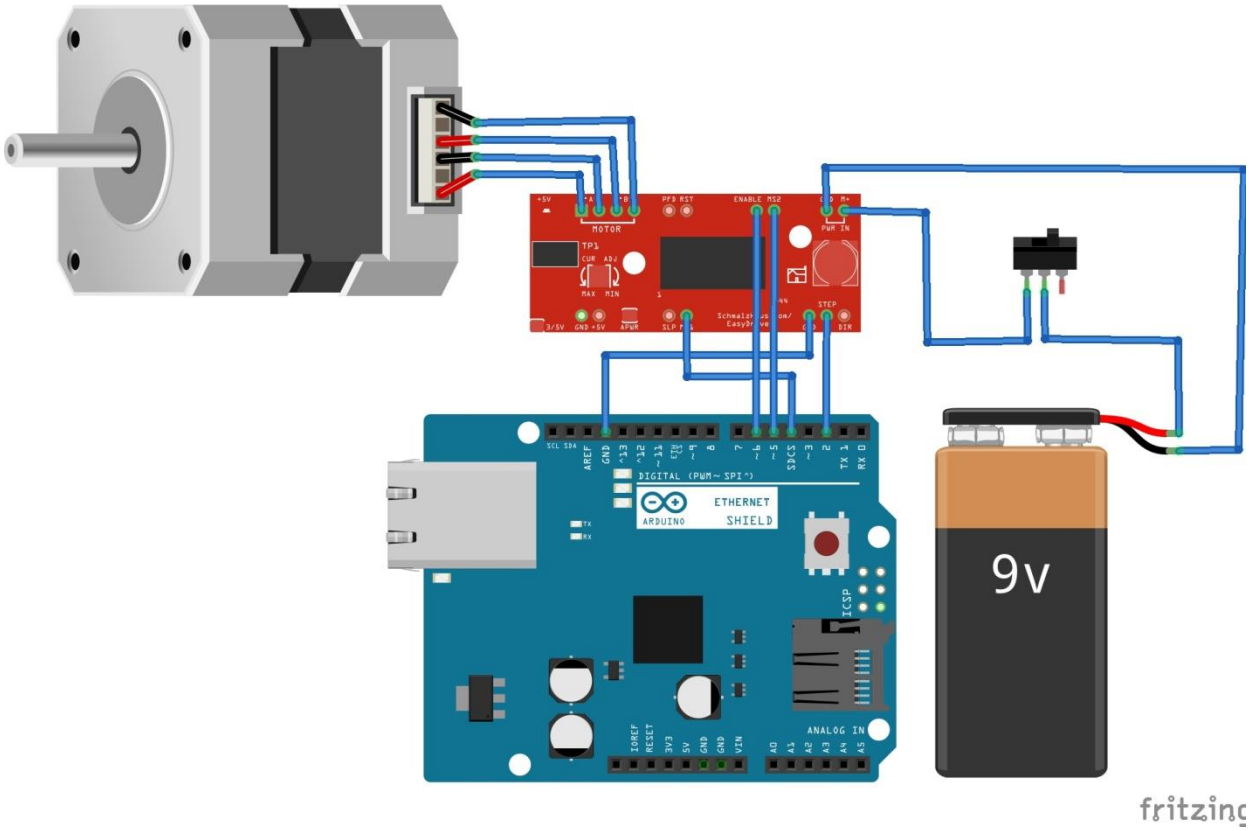
Features of Easy Driver (V44A3967)

- ±750 mA, 30 V output rating
- Satlington® sink drivers
- Automatic current-decay mode detection/selection
- 3.0 to 5.5 V logic supply voltage range
- Mixed, fast, and slow current-decay modes
- Internal UVLO and thermal shutdown circuitry
- Crossover-current protection



- We use pin 9 as the STEP control and pin 8 as the DIRECTION control to the Easy Driver.
- Each time the "digitalWrite(9, HIGH);" call is executed, the stepper motor will move 1/8th of a full step
- So if your motor is 1.8 degrees per step, there will be 200 full steps per revolution, or 1600 microsteps per revolution.
- Thus for 18degrees, we have 1600*18/360=80microsteps.

STEPPER MOTOR CONTROL WITH ARDUINO



ARDUINO CODE

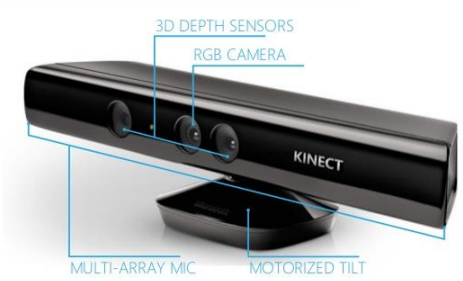
```
//Declare pin functions on Redboard
#define stp 2
#define dir 3
#define MS1 4
#define MS2 5
#define EN  6
//Declare variables for functions
char user_input;
int x;

void setup() {
  pinMode(stp, OUTPUT);
  pinMode(dir, OUTPUT);
  pinMode(MS1, OUTPUT);
  pinMode(MS2, OUTPUT);
  pinMode(EN, OUTPUT);
  Serial.begin(9600); //Open Serial connection for debugging
  Serial.println("Welcone to Robotic Workshop ");
  Serial.println("Begin motor control");
  Serial.println();
  //Print function list for user selection
  Serial.println("Enter number for control option:");
  Serial.println("1. Turn in Forward direction at default microstep mode.");
  Serial.println("2. Turn in Reverse direction at default microstep mode.");
  Serial.println();
}
```

STEPPER MOTOR CONTROL WITH ARDUINO

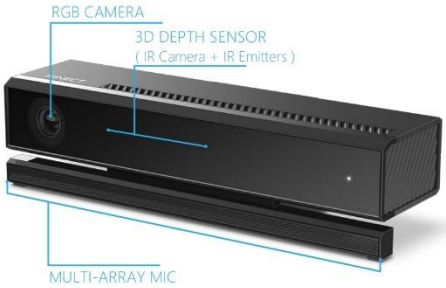
```
//Main loop
void loop() {
  while(Serial.available()){
      user_input = Serial.read(); //Read user input and trigger appropriate
function
      digitalWrite(EN, LOW); //Pull enable pin low to allow motor control
      if (user_input =='1')
      {
         StepForwardDefault();
      }
      else if(user_input =='2')
      {
        ReverseStepDefault();
      }
      else
      {
        Serial.println("Invalid option entered.");
        Serial.println("Enter new option");
        }
  }
}

//Default microstep mode function-1
void StepForwardDefault()
{
  Serial.println("Moving forward at default step mode.");
  digitalWrite(dir, LOW); //Pull direction pin low to move "forward"
  for(x= 1; x<50; x++)  //Loop the forward stepping enough times for motion
to be visible
  {
    digitalWrite(stp,HIGH); //Trigger one step forward
    delay(50);
    digitalWrite(stp,LOW); //Pull step pin low so it can be triggered again
    delay(50);
  }
  Serial.println("Enter new option");
  Serial.println();
}

//Reverse default microstep mode function-2
void ReverseStepDefault()
{
  Serial.println("Moving in reverse at default step mode.");
  digitalWrite(dir, HIGH); //Pull direction pin high to move in "reverse"
  for(x= 1; x<50; x++)  //Loop the stepping enough times for motion to be
visible
  {
    digitalWrite(stp,HIGH); //Trigger one step
    delay(50);
    digitalWrite(stp,LOW); //Pull step pin low so it can be triggered again
    delay(50);
  }
  Serial.println("Enter new option");
  Serial.println();
}
```

Skeleton Tracking Using Kinect v2

Kinect for Windows v1 Sensor



Kinect for Windows v2 Sensor



Kinect for Windows v2 Sensor

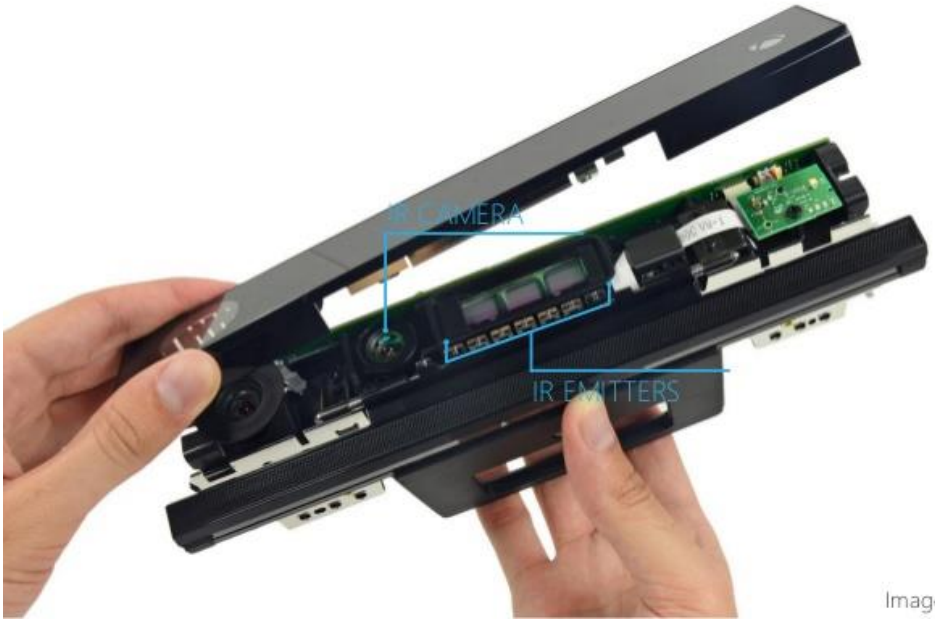


Image by iFixit

System / Software Requirements

OS *	Windows 8, 8.1, Embedded 8, Embedded 8.1 (x64)
CPU	Intel Core i7 3.1GHz (or higher)
RAM	4GB (or more)
GPU *	DirectX 11 supported
USB *	USB 3.0 (Intel or Renesas Host Controller)
Compiler *	Visual Studio 2012, 2013 (Supported Express)
Language	Native (C++), Managed (C#,VB.NET), WinRT (C#,HTML)
Other	Unity Pro (Add-in), Cinder, openFrameworks (wrapper)

Skeleton Tracking Using Kinect v2

Specifications

	Kinect for Windows v1	Kinect for Windows v2
Color	640 × 480 @ 30fps	1920 × 1080 @ 30fps
Depth	320 × 240 @ 30fps	512 × 424 @ 30fps
Sensor	Structured Light (PrimeSense Light Coding)	Time of Flight (ToF)
Range	0.8~4.0 m	0.5~4.5 m
Angle of View Horizontal / Vertical	57 / 43 degree	70 / 60 degree
Microphone Array	○	○

Specifications

	Kinect for Windows v1	Kinect for Windows v2
BodyIndex	6 people	6 people
Body	2 people	6 people
Joint	20 joint/people	25 joint/people
Hand State	Open / Closed	Open / Closed / Lasso
Gesture	×	○
Face	○	◎
Speech / Beamforming	○	○

Skeleton Tracking Using Kinect v2

Basic Features

- Color
  - 1920×1080@30fps / 15fps (Lighting Condition)
  - RGBA, YUV, BGRA, Bayer, YUY2

Basic Features

- Depth
  - 512×424@30fps
  - 500~4500[mm]
  - ToF (Time of Flight)

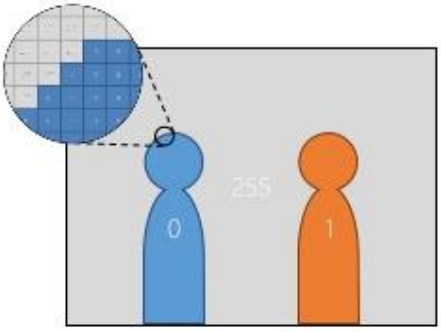
Basic Features

- Infrared / LongExposureInfrared
  - 512×424@30fps
  - 16bit (higher 8 bits)

Skeleton Tracking Using Kinect v2

Basic Features

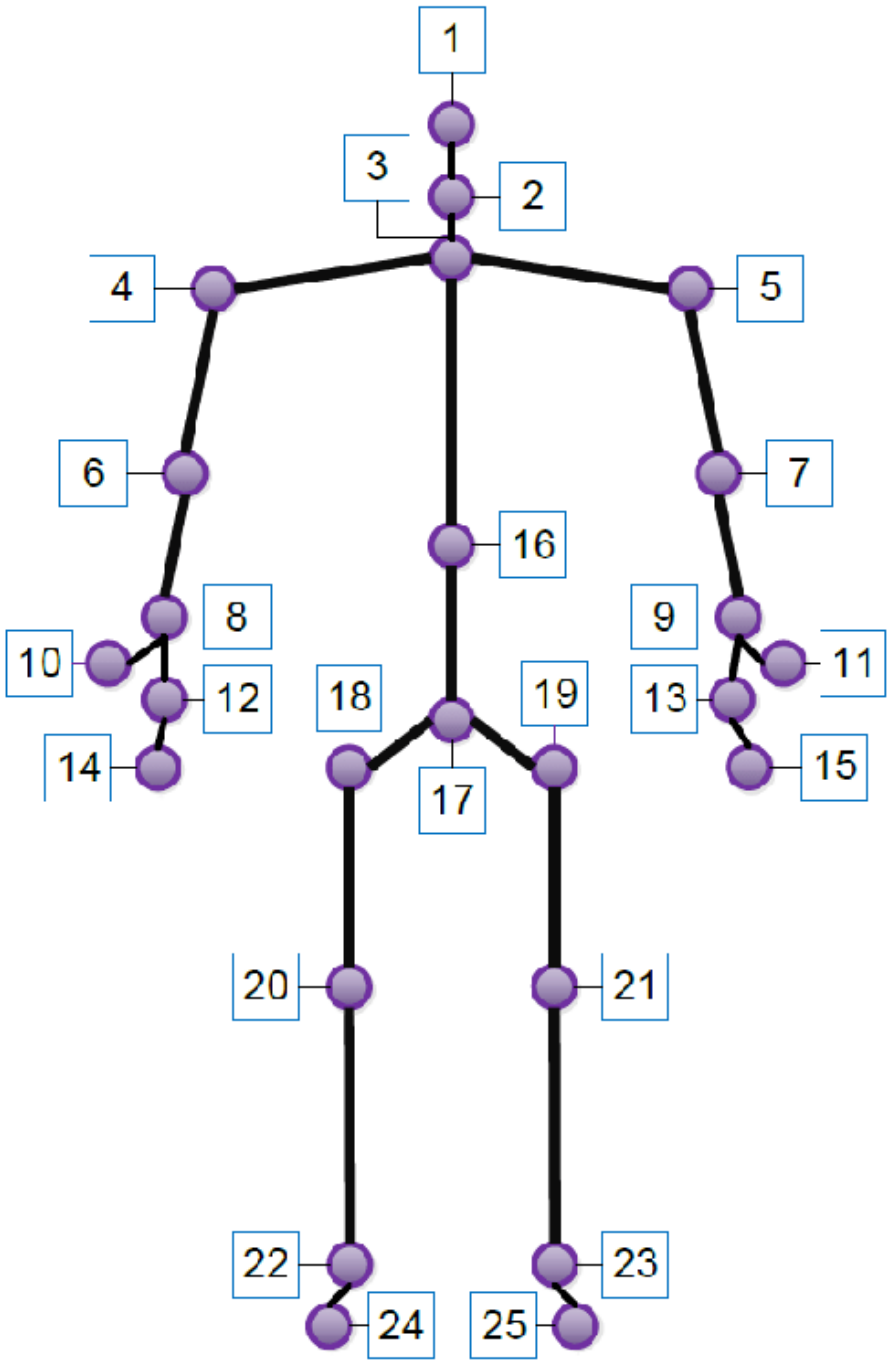
- BodyIndex
  - 512×424@30fps
  - 6 people
  - Body Area : 0~5, Other Area : 255 (5 < Index)



Basic Features

- Body
  - 6 people
  - 25 joint / people (Add Tip, Thumb, Neck)
  - Orientation (Quaternion)
  - Hand Type (Right, Left), Hand State (Open, Closed, Lasso), Lean (-1.0f~1.0f)

Skeleton Tracking Using Kinect v2



1. Head 2. Neck 3. SpineShoulder 4. ShoulderLeft 5. ShoulderRight	6. ElbowLeft 7. ElbowRight 8. WristLeft 9. WristRight 10. ThumbLeft	11. ThumbRight 12. HandLeft 13. HandRight 14. HandTipLeft 15. HandTipRight

16. SpineMid 17. SpineBase 18. HipLeft 19. HipRight 20. KneeLeft	21. KneeRight 22. AnkleLeft 23. AnkleRight 24. FootLeft 25. FootRight

Skeleton Tracking Using Kinect v2

Demo 1:- Skeleton detection and overlaying it on colour image

CODE : –

```
% Create color and depth kinect videoinput objects.
colorVid = videoinput('kinect', 1)
depthVid = videoinput('kinect', 2)
% Look at the device-specific properties on the depth source device,
% which is the depth sensor on the Kinect V2.
% Set 'EnableBodyTracking' to on, so that the depth sensor will
% return body tracking metadata along with the depth frame.
depthSource = getselectedsource(depthVid);
depthSource.EnableBodyTracking = 'on';
% Acquire 100 color and depth frames.
framesPerTrig = 100;
colorVid.FramesPerTrigger = framesPerTrig;
depthVid.FramesPerTrigger = framesPerTrig;
% Start the depth and color acquisition objects.
% This begins acquisition, but does not start logging of acquired data.
pause(5);
start([depthVid colorVid]);
% Get images and metadata from the color and depth device objects.
[colorImg] = getdata(colorVid);
[~, ~, metadata] = getdata(depthVid);
% These are the order of joints returned by the kinect adaptor.
%     SpineBase = 1;
%     SpineMid = 2;
%     Neck = 3;
%     Head = 4;
%     ShoulderLeft = 5;
%     ElbowLeft = 6;
%     WristLeft = 7;
%     HandLeft = 8;
%     ShoulderRight = 9;
%     ElbowRight = 10;
%     WristRight = 11;
%     HandRight = 12;
%     HipLeft = 13;
%     KneeLeft = 14;
%     AnkleLeft = 15;
%     FootLeft = 16;
%     HipRight = 17;
%     KneeRight = 18;
```

Skeleton Tracking Using Kinect v2

```
%     AnkleRight = 19;
%     FootRight = 20;
%     SpineShoulder = 21;
%     HandTipLeft = 22;
%     ThumbLeft = 23;
%     HandTipRight = 24;
%     ThumbRight = 25;
% Create skeleton connection map to link the joints.
SkeletonConnectionMap = [ [4 3];  % Neck
                          [3 21]; % Head
                          [21 2]; % Right Leg
                          [2 1];
                          [21 9];
                          [9 10];  % Hip
                          [10 11];
                          [11 12]; % Left Leg
                          [12 24];
                          [12 25];
                          [21 5];  % Spine
                          [5 6];
                          [6 7];   % Left Hand
                          [7 8];
                          [8 22];
                          [8 23];
                          [1 17];
                          [17 18];
                          [18 19];  % Right Hand
                          [19 20];
                          [1 13];
                          [13 14];
                          [14 15];
                          [15 16];
                        ];

% Extract the 90th frame and tracked body information.
lastFrame = framesPerTrig-10;
lastframeMetadata = metadata(lastFrame);
% Find the indexes of the tracked bodies.
anyBodiesTracked = any(lastframeMetadata.IsBodyTracked ~= 0);
trackedBodies = find(lastframeMetadata.IsBodyTracked);
% Find number of Skeletons tracked.
nBodies = length(trackedBodies);
% Get the joint indices of the tracked bodies with respect to the color
% image.
colorJointIndices = lastframeMetadata.ColorJointIndices(:, :, trackedBodies);
% Extract the 90th color frame.
lastColorImage = colorImg(:, :, :, lastFrame);
% Marker colors for up to 6 bodies.
colors = ['r';'g';'b';'c';'y';'m'];
% Display the RGB image.
imshow(lastColorImage);
% Overlay the skeleton on this RGB frame.
for i = 1:24
     for body = 1:nBodies
         X1 = [colorJointIndices(SkeletonConnectionMap(i,1),1,body)
colorJointIndices(SkeletonConnectionMap(i,2),1,body)];
         Y1 = [colorJointIndices(SkeletonConnectionMap(i,1),2,body)
colorJointIndices(SkeletonConnectionMap(i,2),2,body)];
         line(X1,Y1, 'LineWidth', 1.5, 'LineStyle', '-', 'Marker', '+', 'Color',
colors(body));
     end

    hold on;
 end
 hold off;
```

Skeleton Tracking Using Kinect v2

Demo 2:- Skeleton mapping on real-time depth streaming video

CODE : –

```
imaqreset;
colorVid = videoinput('kinect', 1);
depthVid = videoinput('kinect', 2);
triggerconfig (depthVid,'manual');
framesPerTrig = 1;
depthVid.FramesPerTrigger=framesPerTrig;
depthVid.TriggerRepeat=inf;
src = getselectedsource(depthVid);
src.EnableBodyTracking = 'on';
start(depthVid);
himg = figure;
SkeletonConnectionMap = [ [4 3];  % Neck
                          [3 21]; % Head
                          [21 2]; % Right Leg
                          [2 1];
                          [21 9];
                          [9 10];  % Hip
                          [10 11];
                          [11 12]; % Left Leg
                          [12 24];
                          [12 25];
                          [21 5];  % Spine
                          [5 6];
                          [6 7];   % Left Hand
                          [7 8];
                          [8 22];
                          [8 23];
                          [1 17];
                          [17 18];
                          [18 19];  % Right Hand
                          [19 20];
                          [1 13];
                          [13 14];
                          [14 15];
                          [15 16];
                        ];
while ishandle(himg);
trigger (depthVid);
```

Skeleton Tracking Using Kinect v2

```
[depthMap, ts, depthMetaData] = getdata (depthVid);
anyBodiesTracked = any(depthMetaData.IsBodyTracked ~= 0);
trackedBodies = find(depthMetaData.IsBodyTracked);
nBodies = length(trackedBodies);
colors = ['r';'g';'b';'c';'y';'m'];
imshow (depthMap, [0 4096]);
if  sum(depthMetaData.IsBodyTracked) >0
skeletonJoints = depthMetaData.DepthJointIndices (:,:,depthMetaData.IsBodyTracked);
hold on;
for i = 1:24
for body = 1:nBodies
X1 = [skeletonJoints(SkeletonConnectionMap(i,1),1,body);
skeletonJoints(SkeletonConnectionMap(i,2),1,body)];
Y1 = [skeletonJoints(SkeletonConnectionMap(i,1),2,body),
skeletonJoints(SkeletonConnectionMap(i,2),2,body)];
line(X1,Y1, 'LineWidth', 2, 'LineStyle', '-' , 'Marker', '+', 'Color',
colors(body));
end
end
hold off;
end
end
stop(depthVid);
```

Demo 3:- Skeleton detection on real-time colour streaming video & recording of it for future analysis

CODE : –

```
clc
clear
imaqreset
delete(imaqfind)
pause(2)
% Create color and depth kinect videoinput objects.
colorVid = videoinput('kinect', 1)
depthVid = videoinput('kinect', 2)

% Look at the device-specific properties on the depth source device,
% which is the depth sensor on the Kinect V2.
% Set 'EnableBodyTracking' to on, so that the depth sensor will
% return body tracking metadata along with the depth frame.
depthSource = getselectedsource(depthVid);
depthSource.EnableBodyTracking = 'on';

% Acquire 100 color and depth frames.
```

Skeleton Tracking Using Kinect v2

```
framesPerTrig = 200;
colorVid.FramesPerTrigger = framesPerTrig;
depthVid.FramesPerTrigger = framesPerTrig;

% Start the depth and color acquisition objects.
% This begins acquisition, but does not start logging of acquired data.
pause(5);
start([depthVid colorVid]);

% Get images and metadata from the color and depth device objects.
[colorImg] = getdata(colorVid);
[~, ~, metadata] = getdata(depthVid);

% Create skeleton connection map to link the joints.
SkeletonConnectionMap = [ [4 3];  % Neck
                          [3 21]; % Head
                          [21 2]; % Right Leg
                          [2 1];
                          [21 9];
                          [9 10];  % Hip
                          [10 11];
                          [11 12]; % Left Leg
                          [12 24];
                          [12 25];
                          [21 5];  % Spine
                          [5 6];
                          [6 7];   % Left Hand
                          [7 8];
                          [8 22];
                          [8 23];
                          [1 17];
                          [17 18];
                          [18 19];  % Right Hand
                          [19 20];
                          [1 13];
                          [13 14];
                          [14 15];
                          [15 16];
                        ];


trackingIndex = [];
trackingIncrement = 1;
for i=1:framesPerTrig
    if(sum(metadata(i).BodyTrackingID) ~= 0)
        trackingIndex(trackingIncrement)=i;
        trackingIncrement=trackingIncrement+1;
    else
        disp("yup")
    end
end
metadata = metadata(trackingIndex);
colorImg = colorImg(:,:,:,trackingIndex);
colors = ['r';'g';'b';'c';'y';'m'];
hfig = figure('units','normalized','outerposition',[0 0 1 1])
% % Overlay the skeleton on this RGB frame.
for j=1:size(metadata)
    anyBodiesTracked = any(metadata(j).IsBodyTracked ~= 0);
trackedBodies = find(metadata(j).IsBodyTracked);
nBodies = length(trackedBodies);
colorJointIndices = metadata(j).ColorJointIndices(:, :, trackedBodies);
```

Skeleton Tracking Using Kinect v2

```
imshow(colorImg(:,:,:,j));
for i = 1:24
     for body = 1:nBodies
         X1 = [colorJointIndices(SkeletonConnectionMap(i,1),1,body)
colorJointIndices(SkeletonConnectionMap(i,2),1,body)];
         Y1 = [colorJointIndices(SkeletonConnectionMap(i,1),2,body)
colorJointIndices(SkeletonConnectionMap(i,2),2,body)];
         line(X1,Y1, 'LineWidth', 1.5, 'LineStyle', '-', 'Marker', '+',
'Color', colors(body));
     end
    hold on;
end
drawnow
tmp = getframe(hfig);
 colorImg2(:,:,:,j) = imresize(tmp.cdata,[1280 1920]);
 hold off;
end
 mov=immovie(colorImg2);
 implay(mov);
```

EXPORTING VIDEO FILE

```
>> implay(mov);
>> myVideo = VideoWriter('myfile.avi');
>> open(myVideo);
>> writeVideo(myVideo, mov);
>> close(myVideo);
```

Applications:-

1. Pose Estimation
2. Gait Analysis
3. Animation Characters' real-time motion/movement control
4. Real-time Interactive Robotic Control
5. Virtual Dance trainer platform with scoring based on how much perfect is learned
6. Gesture based interactive platform

..... and so many

Experiment (Controlling of robotic arm with skeleton data):-

Step i.	Create colour & depth video streaming object
Step ii.	Check the available data within created colour & video object
Step iii.	Filter out detected only one skeleton data based on depth distance
Step iv.	Detect the joint points & calculate the joint angle
Step v.	Map & feed the angle to the used motor through microcontroller