# PANIMALAR INSTITUTE OF TECHNOLOGY

**JAISAKTHI EDUCATIONAL TRUST**
**(Affiliated to Anna University, Chennai) Bangalore Trunk Road, Varadharajapuram**
**Poonamallee, Chennai – 600 123**

## DEPARTMENT OF
## ARTIFICIAL INTELLIGENCE & DATA SCIENCE

**AD8705 ARTIFICIAL INTELLIGENCE & ROBOTICS LABORATORY**

**VII SEMESTER - IV YEAR**

**LAB MANUAL**

**ACADEMIC YEAR 2023-24**

## TABLE OF CONTENTS



## LIST OF THE EXPERIMENTS

# VISION AND MISSION OF THE INSTITUTE

## VISION

An Institution of Excellence by imparting quality education and serve as a perennial source of technical manpower with dynamic professionalism and entrepreneurship having social responsibility for the progress of the society and nation

## MISSION

Panimalar Institute of Technology will strive to emerge as an Institution of Excellence in the country by

- Providing state-of-the-art infrastructure facilities for designing and developing solutions for engineering problems.
- Imparting quality education and training through qualified, experienced and committed members of the faculty.
- Inculcating high moral values in the minds of the Students and transforming them into a well-rounded personality.
- Establishing Industry Institute interaction to make students ready for the industrial environment.
- Promoting research based projects/activities in the emerging areas of Engineering & Technology.

# VISION AND MISSION OF THE DEPARTMENT

## VISION

To Cultivate the Computer Science and Business System with excellence in Computer Science Technology and managing business opportunities with adequate training in skill development via research, innovation, entrepreneur development , communication development and business mind that empowers budding engineers to establish a business thoughtfulness in computer technology.

## MISSION

- To equip qualified engineers by providing great skill development opportunities.
- To educate innovators through entrepreneurial and communication skills.
- To generate new engineers who have innovative business ideas.
- To foster a business mindset by embracing technological innovation.
- To encourage industry-institute collaboration in furthering computer technology

## PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

**PEO 1 :** To provide graduates with the proficiency to utilize the fundamental knowledge of basic sciences, mathematics, Artificial Intelligence, data science and statistics to build systems that require management and analysis of large volume of data.

**PEO 2 :** To enrich graduates with necessary technical skills to pursue pioneering research in the field of AI and Data Science and create disruptive and sustainable solutions for the welfare of ecosystems.

**PEO 3 :** To enable graduates to think logically, pursue lifelong learning and collaborate with an ethical attitude in a multidisciplinary team.

**PEO 4 :** To enable the graduates to design and model AI based solutions to critical problem domains in the real world.

**PEO 5 :** To enrich the innovative thoughts and creative ideas of the graduates for effective contribution towards economy building.

## PROGRAM OUTCOMES OF THE DEPARTMENT

Engineering Graduates will be able to:

**PO 1 :** **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems

**PO 2 :** **Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO 3 :** **Design/Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO 4 :** **Conduct Investigations of Complex Problems:** Use research-based knowledge and research methods, including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO 5 :** **Modern Tool Usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling of complex engineering activities with an understanding of the limitations.

**PO 6 :** **The Engineer and Society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**PO 7 :** **Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO 8 :** **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**PO 9 :** **Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO 10 :** **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO 11 :** **Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO 12 :** **Life-long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## PROGRAM SPECIFIC OUTCOMES OF THE DEPARTMENT

**PSO 1 :** Graduates should be able to evolve AI based efficient domain specific processes for effective decision making in several domains such as business and governance domains.

**PSO 2 :** Graduates should be able to arrive at actionable Fore sight, Insight , hind sight from data for solving business and engineering problems.

**PSO 3 :** Graduates should be able to create, select and apply the theoretical knowledge of AI and Data Analytics along with practical industrial tools and techniques to manage and solve wicked societal problems.

**PSO 4 :** Graduates should be capable of developing data analytics and data visualization skills, skills pertaining to knowledge acquisition, knowledge representation and knowledge engineering, and hence capable of coordinating complex projects

**PSO 5 :** Graduates should be able to carry out fundamental research to cater the critical needs of the society through cutting edge technologies of AI.

## SYLLABUS

## ARTIFICIAL INTELLIGENCE &ROBOTICS LABORATORY

| L | T | P | C |
|---|---|---|---|
| 3 | 0 | 2 | 4 |

**COURSE OBJECTIVES:**

1. To study the Robot Locomotion and types of robots.
2. To explore the kinematic models and constraints
3. To Learn sensors of robots and image processing for robotics.
4. To understand the methods for mobile robot Localization
5. To study the Path planning and Navigation of Robots.

**PRACTICAL EXERCISES:**

1. Line tracing bot
2. Gesture controlled bot
3. 4(Four) DOF Robotic Arm
4. Home Security System using NodeMCU
5. RF Controlled or WiFi controlled Navigation bot
6. Pick and place bot with Object Detetction
7. Wall Following bot
8. Maze solving Robot
9. Forward and reverse kinematics based experiment using open source platforms
10. Computer Visio based robotic tasks execution

**COURSE OUTCOMES:**

**At the end of this course, the students will be able to:**

CO1: Explain the types of Robots

CO2: Narrate the kinematics of Robots

CO3: Implement image processing algorithms

CO4: Devise Localization algorithms

CO5: Devise Path planning methods for navigation

# TABLE OF CONTENTS

## Ex.No:1

## LINE TRACING BOT

## Aim

To execute the line tracing robot using Arduino IDE.

## Hardware Requirement:

1. Arduino Uno
2. 1x IR Sensor Module
3. 1x L293D Motor Driver IC
4. Robotic Platform / Chassis
5. Breadboard / PCB ( It's up to you )
6. 2x DC Motors
7. 1x Castor Wheel
8. 2x Wheels
9. 1x 12V Battery
10. 1x 9V Battery
11. Jumper Cables

## Software Requirement:

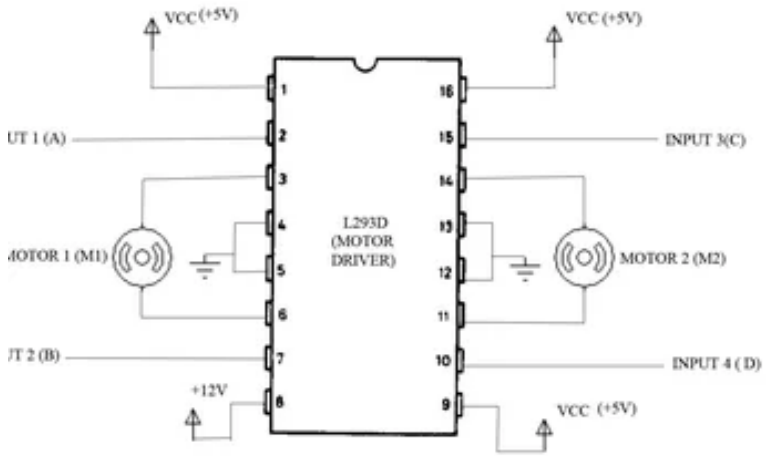
1. Arduino IDE

## Procedure:

Make the connections as follows:

1. ARDUINO VCC - SENSOR MODULE VCC
2. ARDUINO GND - SENSOR MODULE GND
3. ARDUINO DIGITAL PIN 12 - SENSOR MODULE S1
4. ARDUINO DIGITAL PIN 11 - SENSOR MODULE S2
5. ARDUINO DIGITAL PIN 10 - SENSOR MODULE S3
6. ARDUINO DIGITAL PIN 9 - SENSOR MODULE S4
7. ARDUINO DIGITAL PIN 8 - SENSOR MODULE S5

## Diagram:




## Program Code:

```
#define lmotorf 4 //Motor A1
#define lmotorb 5 //Motor A2
#define rmotorf 6 //Motor B1
#define rmotorb 7 //Motor B2

//HIGH white
//LOW black

void setup() {
pinMode(lmotorf,OUTPUT);       //Just connect all th pin to the gpio of the board
pinMode(rmotorf,OUTPUT);      //Output pins pinmode, These are the output pins
pinMode(lmotorb,OUTPUT);      //And These connect to the
pinMode(rmotorb,OUTPUT);     //Board L293D motor Driver
pinMode(8,INPUT);
pinMode(9,INPUT);
pinMode(10,INPUT);
pinMode(11,INPUT);
pinMode(12,INPUT);
pinMode(13,INPUT);
}
```

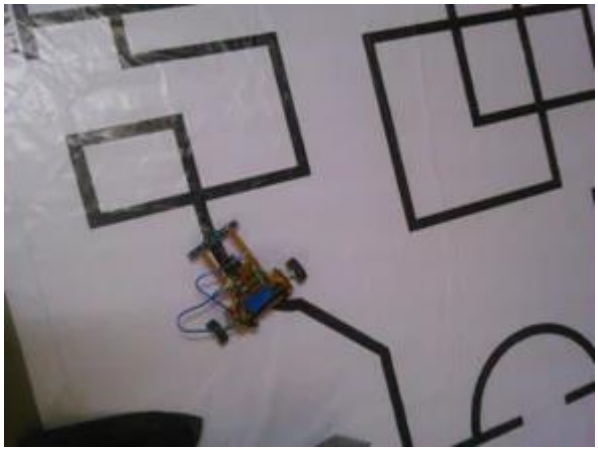
```
void loop() {

int sensor1=digitalRead(8);//sensor1
int sensor2=digitalRead(9);//sensor2
int sensor3=digitalRead(10);//sensor3
int sensor4=digitalRead(11);//sensor4
int sensor5=digitalRead(12);//sensor5

if((sensor2==LOW)&&(sensor3==LOW)&&(sensor4==LOW)) //When ALL ARE on WHITE line
{

digitalWrite(lmotorf,HIGH);
digitalWrite(rmotorf,HIGH);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}
else if((sensor2==HIGH)&&(sensor3==LOW)&&(sensor4==HIGH))

{
digitalWrite(lmotorf,HIGH);
digitalWrite(rmotorf,HIGH);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}
else if((sensor2==LOW)&&(sensor3==HIGH)&&(sensor4==LOW))

{
digitalWrite(lmotorf,HIGH);
digitalWrite(rmotorf,HIGH);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}
else
if((sensor1==HIGH)&&(sensor5==LOW)&&(sensor2==LOW)&&(sensor3==HIGH)&&(sensor4==LOW))//1
LEFT

{
digitalWrite(lmotorf,HIGH);
digitalWrite(rmotorf,LOW);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}

else
if((sensor1==LOW)&&(sensor5==HIGH)&&(sensor2==LOW)&&(sensor3==HIGH)&&(sensor4==HIGH))//1
LEFT

{
digitalWrite(lmotorf,HIGH);
digitalWrite(rmotorf,LOW);
digitalWrite(lmotorb,LOW);
```

```
digitalWrite(rmotorb,LOW);
}
else
if((sensor1==HIGH)&&(sensor5==LOW)&&(sensor2==LOW)&&(sensor3==LOW)&&(sensor4==LOW))//1
LEFT

{
digitalWrite(lmotorf,HIGH);
digitalWrite(rmotorf,LOW);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}

else
if((sensor1==LOW)&&(sensor5==HIGH)&&(sensor2==HIGH)&&(sensor3==HIGH)&&(sensor4==HIGH))//N
ULL

{
digitalWrite(lmotorf,HIGH);
digitalWrite(rmotorf,LOW);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}
else if((sensor2==LOW)&&(sensor3==LOW)&&(sensor4==HIGH))//1 LEFT

{
digitalWrite(lmotorf,HIGH);
digitalWrite(rmotorf,LOW);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}
else if((sensor2==LOW)&&(sensor3==HIGH)&&(sensor4==HIGH))//1 LEFT

{
digitalWrite(lmotorf,HIGH);
digitalWrite(rmotorf,LOW);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}

else if((sensor2==HIGH)&&(sensor3==LOW)&&(sensor4==LOW))//FOR RIGHT

{
digitalWrite(lmotorf,LOW);
digitalWrite(rmotorf,HIGH);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}

else if((sensor2==HIGH)&&(sensor3==HIGH)&&(sensor4==LOW))//FOR RIGHT
```

```
{
digitalWrite(lmotorf,LOW);
digitalWrite(rmotorf,HIGH);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}
else
if((sensor1==HIGH)&&(sensor5==LOW)&&(sensor2==HIGH)&&(sensor3==HIGH)&&(sensor4==HIGH))//1
LEFT
{
digitalWrite(lmotorf,LOW);
digitalWrite(rmotorf,HIGH);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}
else
if((sensor1==HIGH)&&(sensor5==LOW)&&(sensor2==HIGH)&&(sensor3==HIGH)&&(sensor4==LOW))//1
LEFT
{
digitalWrite(lmotorf,LOW);
digitalWrite(rmotorf,HIGH);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}

else if((sensor1==LOW)&&(sensor5==HIGH)&&(sensor2==LOW)&&(sensor3==HIGH)&&(sensor4==LOW))
{
digitalWrite(lmotorf,LOW);
digitalWrite(rmotorf,HIGH);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}
else if((sensor1==HIGH)&&(sensor5==LOW)&&(sensor2==HIGH)&&(sensor3==HIGH)&&(sensor4==HIGH))
{
digitalWrite(lmotorf,LOW);
digitalWrite(rmotorf,HIGH);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}

else if((sensor2==HIGH)&&(sensor3==HIGH)&&(sensor4==HIGH))//FOR RIGHT
{
digitalWrite(lmotorf,HIGH);
digitalWrite(rmotorf,HIGH);
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}
else
{
digitalWrite(lmotorf,LOW);
digitalWrite(rmotorf,LOW);
```

```
digitalWrite(lmotorb,LOW);
digitalWrite(rmotorb,LOW);
}

}
```

## Output:



## Result:

Thus the execution of line tracing bot is done successfully.

## EX. NO: 2

# GESTURE CONTROLLED BOT

## AIM:

To execute the Gesture controlled bot using Arduino IDE.

## Hardware Requirement:

1. Arduino Uno (2)
2. NRF24L01 (2)
3. MPU6050DC Motor (2)
4. L293D Motor Driver Module
5. Battery

## Software Requirement:

1. Arduino IDE

## Procedure:

**Transmitter Side Program**

In this program, Arduino reads the data from MPU6050 and sends it to nRF 24L01 transmitter.

1. Begin the program by adding the required library files.

2. Then define the variables for MPU6050 Gyroscope and Accelerometer data. Here only Accelerometer data will be used.

3. Define the Radio pipe addresses for the communication and nRF transmitters CN and CSN pins.

4. Inside the void setup() function, begin the serial monitor. And also initialize the wire and radio communication. radio.setDataRate is used to set the data transmission rate.

5. Read the MPU6050 sensor data. Here we are only using X and Y direction accelerometer data.

6. Finally, transmit the sensor data using the radio.write function.

**Receiver Side Program**

1. As usual, start the program by including the required library files.

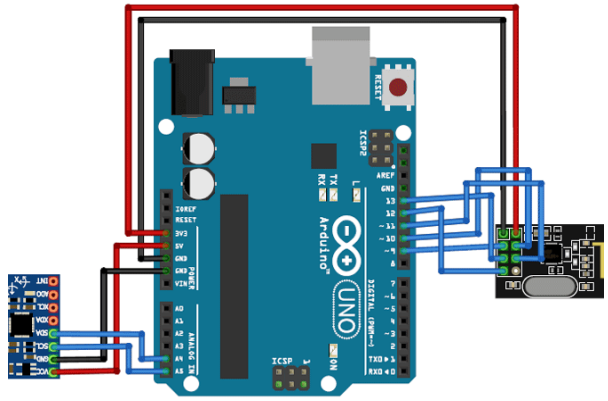
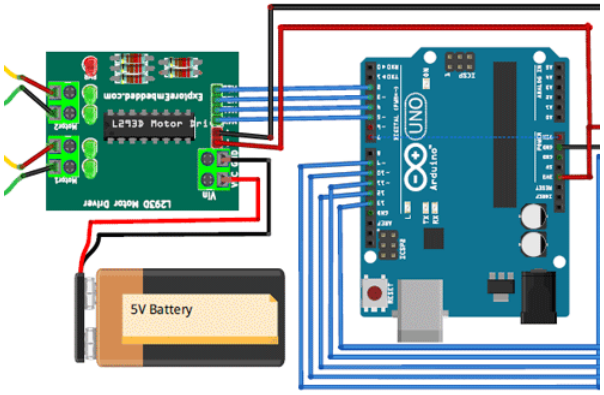
2. Define the Radio pipe addresses for the communication and nRF transmitters CN and CSN pins.

3. Define the left and right DC motor pins.

4. Now check if the radio is available or not. If it is, then read the data.

5. Now compare the received data and drive the motors according to the conditions.

## Diagram:

| Transmitter Pinout Diagram | Receiver Pinout Diagram |
|---|---|
|  |   |

## Program Code:

**Transmitter Side Program**

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include "Wire.h"
#include "I2Cdev.h"
#include "MPU6050.h"

//Define variables for Gyroscope and Accelerometer data

MPU6050 mpu;
int16_t ax, ay, az;
int16_t gx, gy, gz;

const uint64_t pipeOut = 0xE8E8F0F0E1LL;

RF24 radio(9, 10); //  CN and CSN  pins of nrf

struct MyData {
  byte X;
  byte Y;
```

```
};

MyData data;

void setup()
{
  Serial.begin(9600);
  Wire.begin();
  mpu.initialize();

  radio.begin();
  radio.setAutoAck(false);
  radio.setDataRate(RF24_250KBPS);
  radio.openWritingPipe(pipeOut);
}

void loop()
{
  mpu.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

  data.X = map(ax, -17000, 17000, 0, 255 ); //Send X axis data
  data.Y = map(ay, -17000, 17000, 0, 255);  //Send Y axis data

  delay(50);
  radio.write(&data, sizeof(MyData));

  Serial.print("Axis X = ");
  Serial.print(data.X);
  Serial.print("  ");
  Serial.print("Axis X = ");
  Serial.println(data.X);
}
```

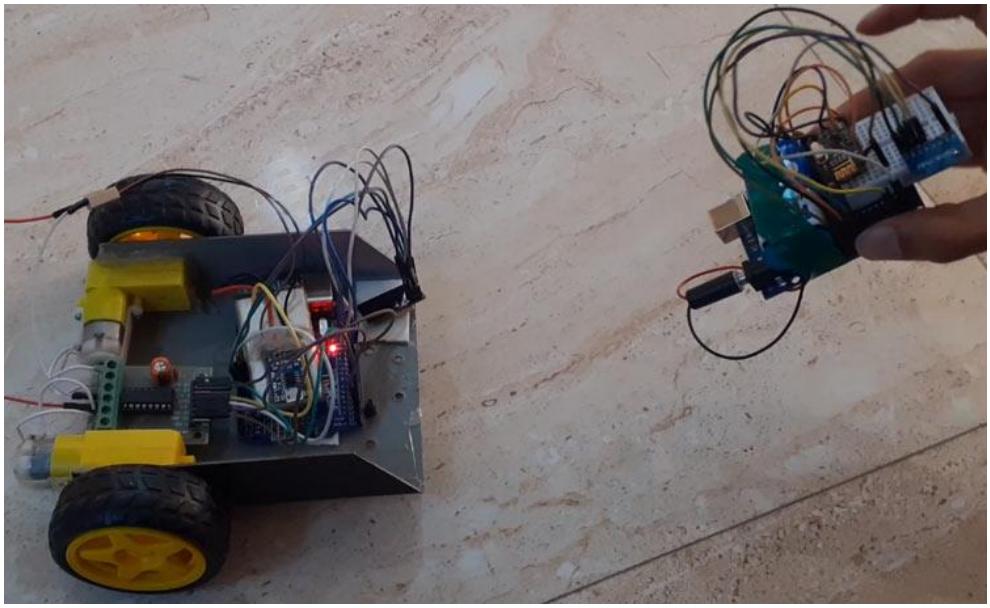
**Receiver Side Program**

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
const uint64_t pipeIn = 0xE8E8F0F0E1LL;

RF24 radio(9, 10);

// Left & Right motor pins
const int IN1 = 2;
const int IN2 = 3;
const int IN3 = 4;
const int IN4 = 5;

struct MyData {
  byte X;
  byte Y;
};
```

```
MyData data;

void setup()
{
  Serial.begin(9600);                    //MyData is the data sent and received by the RF
  radio.begin();                         //Analyzes the Data received from the transmitter
  radio.setAutoAck(false);               //MPU6050 acknowledgementConfig
  radio.setDataRate(RF24_250KBPS);       //Acknowledgement of  the Data ReadRate
  radio.openReadingPipe(1, pipeIn);      //Transmitter OpenReadingPipe
  radio.startListening();                //Handshake Completed
}                                        //And Starts Listening

void recvData()
{
  if ( radio.available() ) {
    radio.read(&data, sizeof(MyData));


    if (data.Y < 80) { //Reverse
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, HIGH);

    }

    if (data.Y > 145) {//forward
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, HIGH);
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
    }

    if (data.X > 155) {//right turn
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, HIGH);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, HIGH);
    }

    if (data.X < 80) {//left turn
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
    }

    if (data.X > 100 && data.X < 170 && data.Y > 80 && data.Y < 130) { //stop car
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, LOW);
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, LOW);
```

```
    }
  }
}

void loop()
{
  recvData();
  Serial.print("X: ");
  Serial.print(data.X);
  Serial.print("    ");
  Serial.print("Y: ");
  Serial.print(data.Y);
  Serial.print("\n");
}
```

## Output:



## Result:

Thus the execution of the Gesture controlled bot is done successfully.

EX. NO: 3

# 4(FOUR) DOF ROBOTIC ARM

## Aim:

To execute the 4(Four) DOF Robotic Arm using Arduino IDE.

## Hardware Requirement:

1. Arduino Uno
2. Breadboard / PCB ( It's up to you )
3. 1x 12V Battery
4. 1x 9V Battery
5. Jumper Cables
6. 4x Servos

## Software Requirement:

1. Arduino IDE

## Procedure:

1. Assemble the robotic arm according to the manufacturer's instructions. Ensure that all the parts are securely connected and that the arm can move freely.

2. Connect the servo motors to the arm. Servo motors are used to control the movement of the arm and are usually connected to the arm's joints. You will need to follow the manufacturer's instructions for connecting the servo motors, which may involve soldering wires or using connectors.

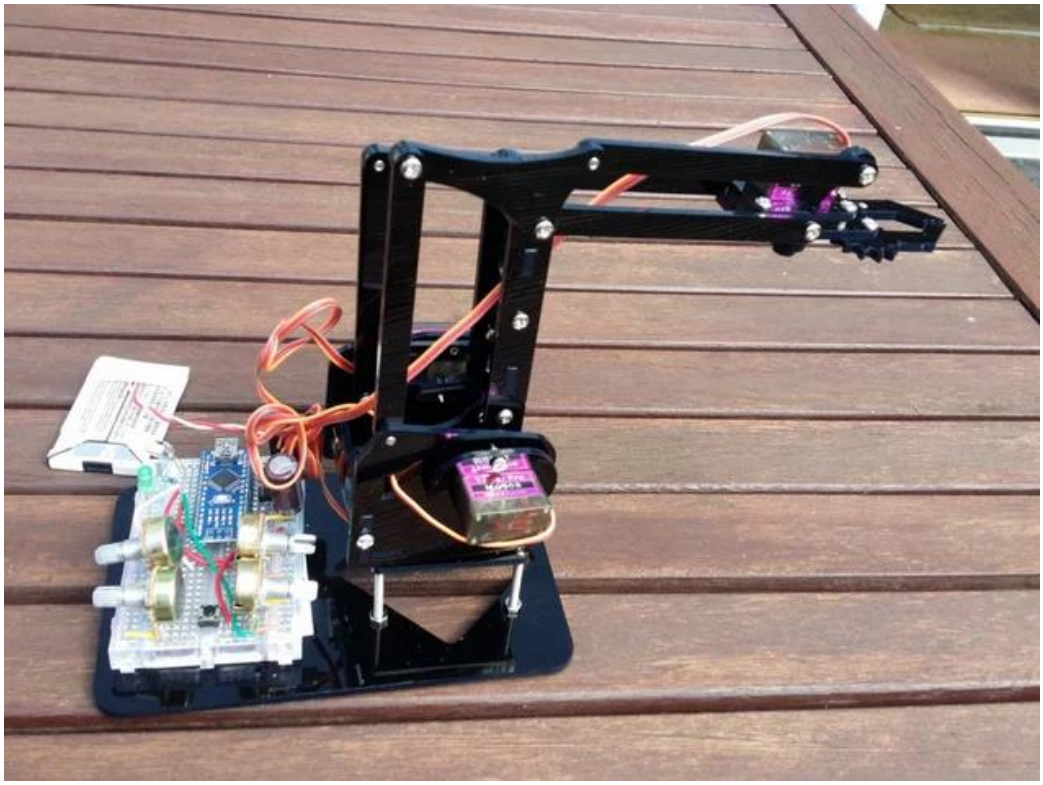
3. Connect the servo motors to a microcontroller or motor driver. A microcontroller or motor driver is used to control the servo motors. You will need to connect the signal wires from each servo motor to the appropriate pins on the microcontroller or motor driver.

4. Program the microcontroller or motor driver to control the servo motors. This will involve writing code to control each servo motor's movement based on the arm's desired movement.

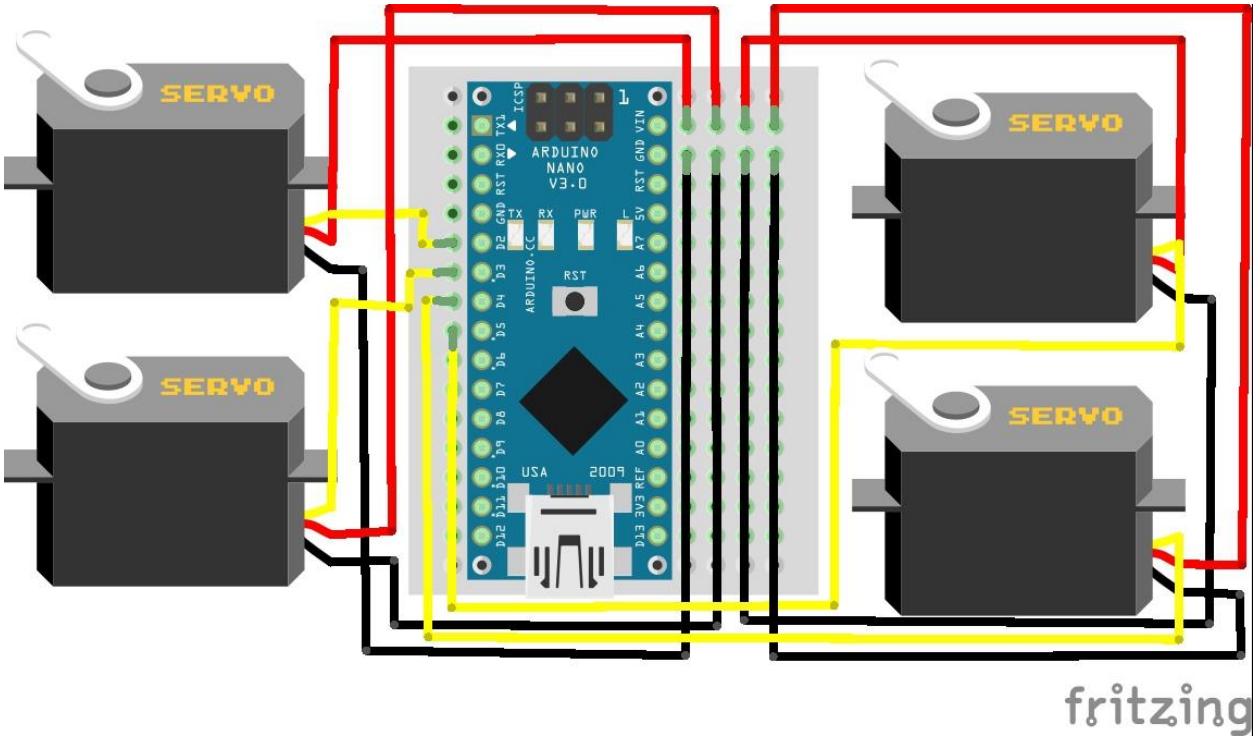
5. Connect the microcontroller or motor driver to a computer or other control device. This will allow you to send commands to the arm and control its movement.

6. Test the robotic arm to ensure that it is working properly. You can do this by sending simple commands to the arm and observing its movement.

## Diagram:



**Pinout Diagram**

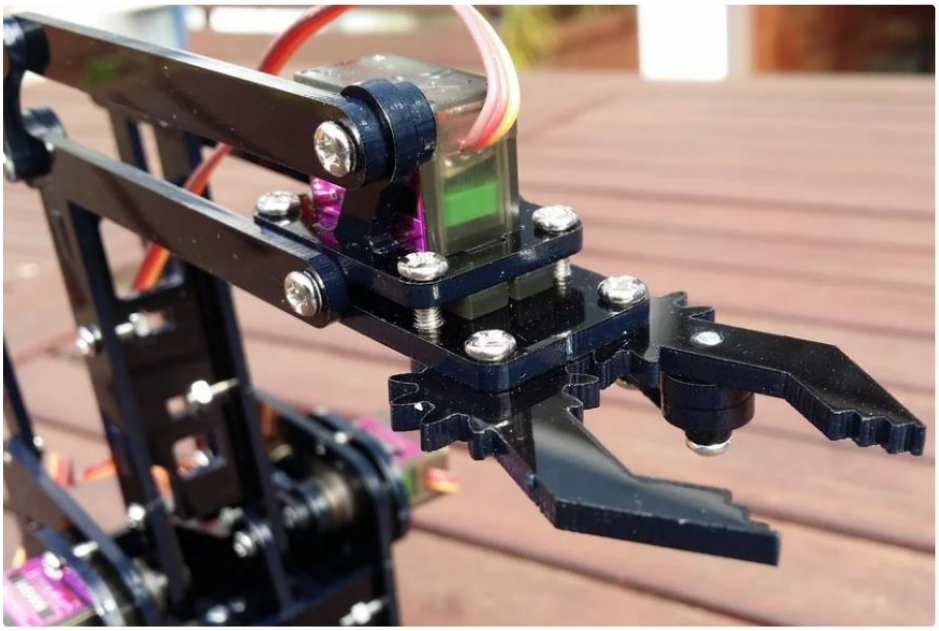


## Program Code:

```
#include <Servo.h>

// Define the servo motor pins
const int basePin = 2;
const int shoulderPin = 3;
const int elbowPin = 4;
const int wristPin = 5;

// Define the servo motor objects
Servo base;
Servo shoulder;
Servo elbow;
Servo wrist;

// Define the initial servo motor angles
int baseAngle = 90;
int shoulderAngle = 90;
int elbowAngle = 90;
int wristAngle = 90;

void setup() {
  // Attach the servo motors to the pins
  base.attach(basePin);
  shoulder.attach(shoulderPin);
  elbow.attach(elbowPin);
  wrist.attach(wristPin);

  // Set the initial servo motor angles
  base.write(baseAngle);
  shoulder.write(shoulderAngle);
  elbow.write(elbowAngle);
  wrist.write(wristAngle);
}

void loop() {
  // Move the base servo motor
  baseAngle += 1;
  if (baseAngle > 180) {
    baseAngle = 0;
  }
  base.write(baseAngle);
  delay(10);

  // Move the shoulder servo motor
  shoulderAngle += 1;
  if (shoulderAngle > 180) {
```

```
    shoulderAngle = 0;
  }
  shoulder.write(shoulderAngle);
  delay(10);

  // Move the elbow servo motor
  elbowAngle += 1;
  if (elbowAngle > 180) {
    elbowAngle = 0;
  }
  elbow.write(elbowAngle);
  delay(10);

  // Move the wrist servo motor
  wristAngle += 1;
  if (wristAngle > 180) {
    wristAngle = 0;
  }
  wrist.write(wristAngle);
  delay(10);
}
```

## Output:



## Result:

Thus the execution of 4(Four) DOF Robotic Arm is done successfully.

## EX. NO: 4

# HOME SECURITY SYSTEM USING NODEMCU

## AIM:

To execute the Home security system using Node MCU using Arduino IDE.

## Basic Requirements:

1. Active Wi-Fi with Internet Connectivity

## Hardware Requirement:

1. NodeMCU "ESP8266"
2. HC-sr501 PIR motion sensor
3. Jumper wires
4. A breadboard - if needed
5. ThingSpeak and Twitter account - if you want to be notified (tweeted)
6. Project enclosure box and power USB cord if you plan to actually use it

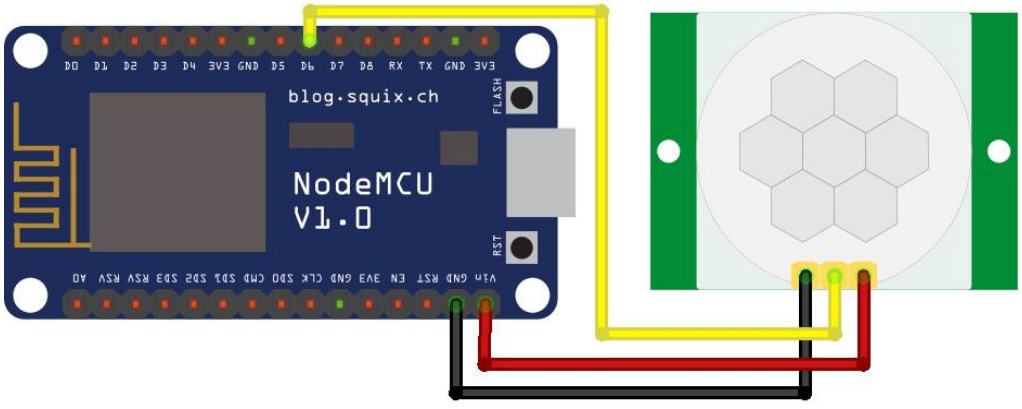
## Software Requirement:

1. Arduino IDE

## Procedure:

1. Connect our Motion Sensor to a NodeMCU
2. Create a ThingSpeak Account and create a Channel
3. Create a ThingSpeak React and Link it to your Twitter Account
4. Download the Arduino Code and make a few changes to it
5. Install the ThingSpeak Library in the Arduino IDE
6. Upload the code

## Pinout Diagram:

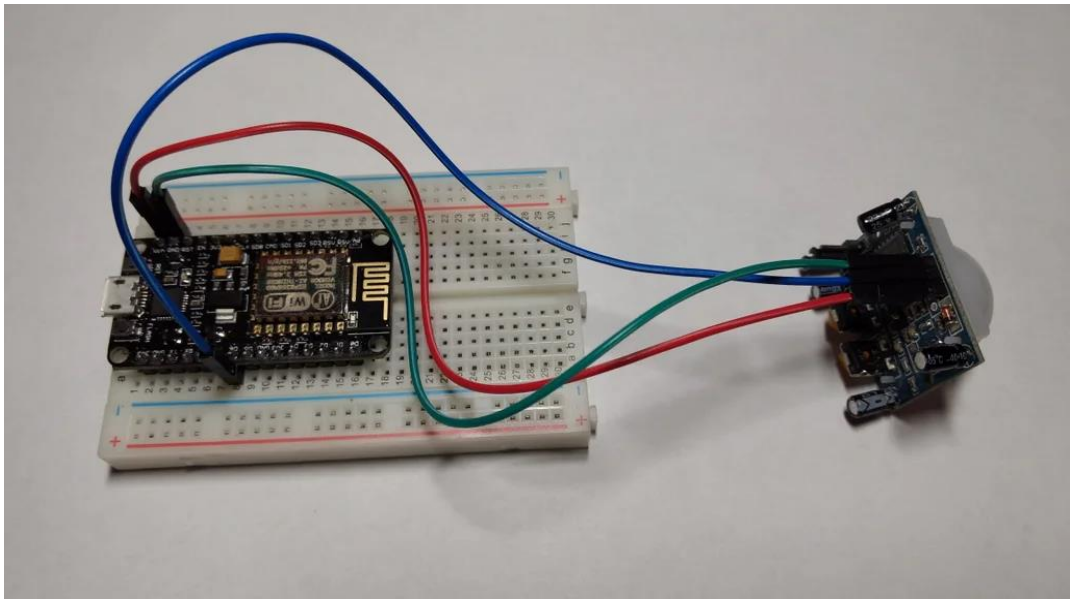


## Program Code:

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <ESP8266HTTPClient.h>

const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";   //Wi-Fi Password

const char* emailFrom = "your_email@gmail.com";
const char* emailFromPassword = "your_email_password";
const char* emailTo = "recipient_email@example.com";
const char* smtpServer = "smtp.gmail.com";
const int smtpPort = 465;

const int pirPin = 2;

WiFiClientSecure client;
HTTPClient http;

void setup() {
  Serial.begin(115200);
  pinMode(pirPin, INPUT);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("WiFi connected");
}

void loop() {
  if (digitalRead(pirPin) == HIGH) {
    Serial.println("Motion detected");

    client.setInsecure();
    if (client.connect(smtpServer, smtpPort)) {
      Serial.println("Connected to SMTP server");
      client.println("EHLO example.com");
      client.println("AUTH LOGIN");
      client.println(base64::encode(emailFrom));
      client.println(base64::encode(emailFromPassword));
      client.println("MAIL FROM:<" + String(emailFrom) + ">");
      client.println("RCPT TO:<" + String(emailTo) + ">");
      client.println("DATA");
      client.println("Subject: Motion detected");
      client.println("Motion was detected at home");
      client.println(".");
      client.println("QUIT");
      Serial.println("Email sent");
```

```
    } else {
      Serial.println("Could not connect to SMTP server");
    }
    client.stop();

    delay(5000);
  }
}
```

## Output:



## Result:

Thus the execution of the Home security system using Node MCU is done successfully.

# EX. NO: 5
# RF CONTROLLED OR WIFI CONTROLLED NAVIGATION BOT

## AIM:

To execute the RF Controlled or WiFi-controlled Navigation bot using Arduino IDE.

## Hardware Requirement:

**The Controller Board :-**

1. Any model/clone Arduino board.
2.The 433MHz OR the 315MHz module (both work the same way) Tx and Rx pair ( Link here)
3. A breadboard for the controller pad (Not needed if you're using a computer to control the bot).
4.Push button switches X 5 (Again not needed if you're using a computer to control the bot).
5. Connecting wires.
6. 9V battery to power the Arduino on the Tx end.
7.A 17cm long breadboard wire for the Antenna for the Tx and Rx.

**The Receiver Board :-**

1. Any Arduino model/clone board.
2. The Rx part of the Tx/Rx pair (Link in Step 2 above).
3. Breadboard and connecting wires.
4.12V battery pack to power Motors.(In the video I have used a 12V AC to DC adapter)
5. 100uF Electrolytic Caps X 4.
6. L293D motor driver chip X 2 (Each chip controls 2 motors bi-directionally...So if you want to make a 2-wheeled bot you will require only 1 L293D chip).
7. 0.1uF ceramic caps X 12 (3 for each motor)....for eliminating noise between the controller and motors.
8. A 9V battery for powering the Arduino on the Rx end.

## Software Requirement:

1. Arduino IDE

## Procedure:

These are the connections between the Arduino and the Rx module.

| Rx | Arduino |
|---|---|
| Vcc | 5v pin |
| Gnd | Gnd pin |
| Data | Digital Pin 2 |

You will also need to connect the two L293D motor driver chips to the Arduino at the receiver.( Pin diagram of L293D given in the pics) Make sure you connect the chip properly...the end with the notch is the front end.

First L293D Arduino

Enable1 ----------> Digital Pin 05
Enable2 ----------> Digital Pin 06
Input 1 ----------> Digital Pin 07
Input 2 ----------> Digital Pin 08
Input 3 ----------> Digital Pin 09
Input 4 ----------> Digital Pin 10

Second L293D Arduino

Enable1 ----------> Digital Pin 03
Enable2 ----------> Digital Pin 11
Input 1 ----------> Digital Pin 04
Input 2 ----------> Digital Pin 12
Input 3 ----------> Digital Pin 14(Analog 0)
Input 4 ----------> Digital Pin 15(Analog 1)

**Additional connections :-**

(1)For the L293D motor drivers :-
Pin 8 of both the L293D chip go to the poitive of the 12V supply....this pin gives the power to the motors.

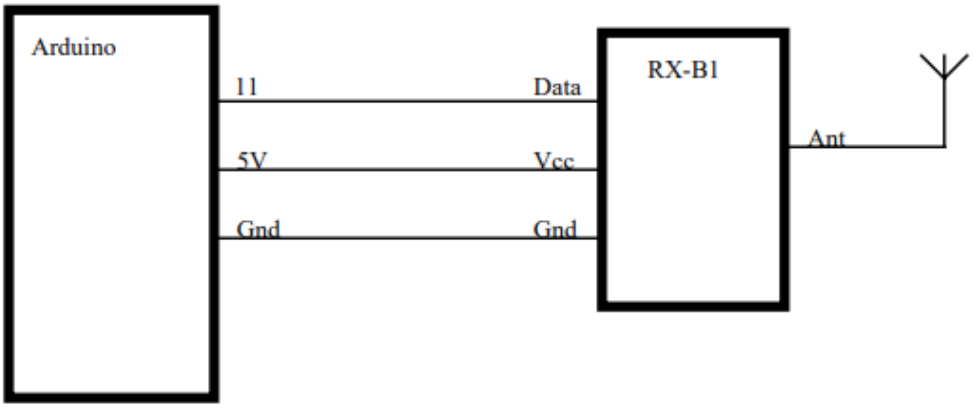
Pins 4,5,12,13 of both chips go to ground.

Pins 3 and 6 are connected to the leads of first motor.....and pins 11,14 go to the leads of the second motor.(Same for the second L293D as well but for third and fourth motor).

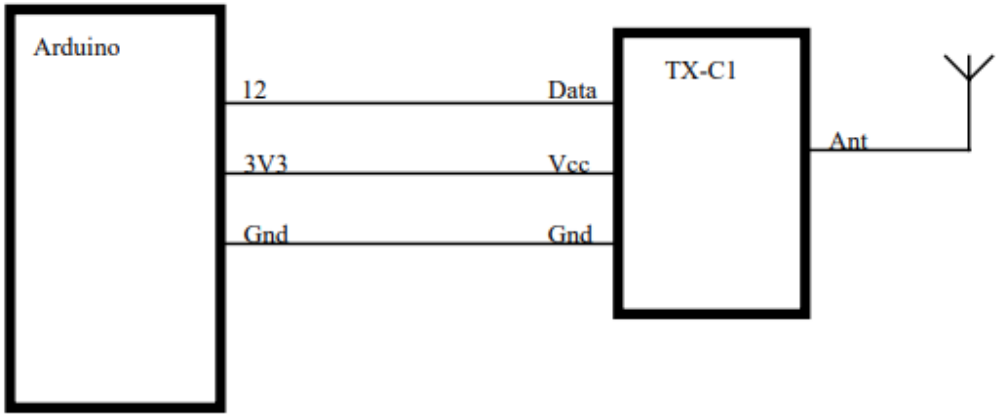
Pin 16 of both chips is connected to the Arduino 5v supply.....this pin powers the chip.

## Pinout Diagram:

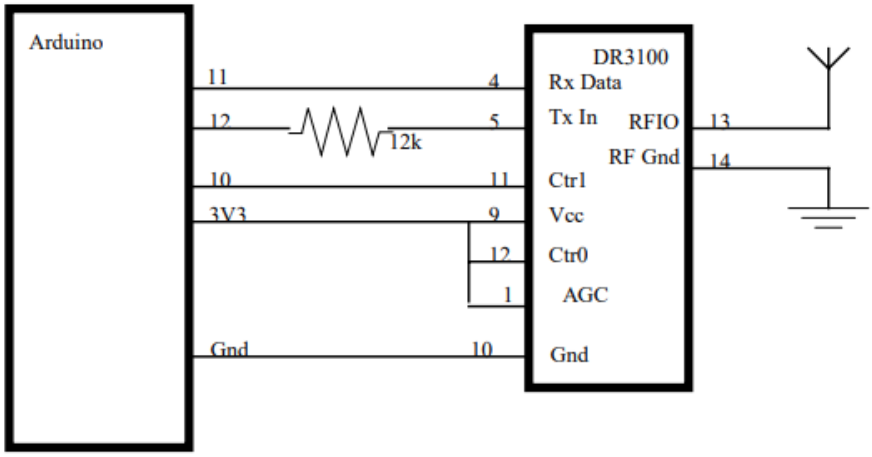
**1.RX-B1 receiver**



**2.TX-C1 transmitter**




**1. DR3100 transceiver**




## Program Code:

```
// TRANSMITTER CODE

#include <VirtualWire.h>

void setup()
{
   Serial.begin(9600);      // Debugging only
   Serial.println("setup");

   // Initialise the IO and ISR

   vw_setup(2000);         // Bits per sec
      vw_set_tx_pin(3); //Transmitter Data Pin to Digital Pin 3
```

```
}//close setup

void loop()
{

 if (Serial.available() > 0)
 {
   int sendByte = Serial.read();//Starts reading data from the serial monitor once condition is met


 switch (sendByte){

  case 'f':  //if the controller type f
 {

  char *msg2 = "f";
  digitalWrite(13, true); // Flash a light to show transmitting
  vw_send((uint8_t *)msg2, strlen(msg2));//send byte to the receiver
  vw_wait_tx(); // Wait until the whole message is gone
  digitalWrite(13, false);
  break;

 }
  case 'b':  //if controller types b
  {
 char *msg2 = "b";
 digitalWrite(13, true); // Flash a light to show transmitting
 vw_send((uint8_t *)msg2, strlen(msg2));//send byte to the receiver
 vw_wait_tx(); // Wait until the whole message is gone
 digitalWrite(13, false);
 break;
  }

  case 's': //if controller types s
  {
 char *msg2 = "s";
 digitalWrite(13, true); // Flash a light to show transmitting
 vw_send((uint8_t *)msg2, strlen(msg2));//send byte to the receiver
 vw_wait_tx(); // Wait until the whole message is gone
 digitalWrite(13, false);
 break;
  }

  case 'l': //if controller types l
 {
 char *msg2 = "l";
 digitalWrite(13, true); // Flash a light to show transmitting
 vw_send((uint8_t *)msg2, strlen(msg2));//send byte to the receiver
 vw_wait_tx(); // Wait until the whole message is gone
 digitalWrite(13, false);
 break;
 }
```

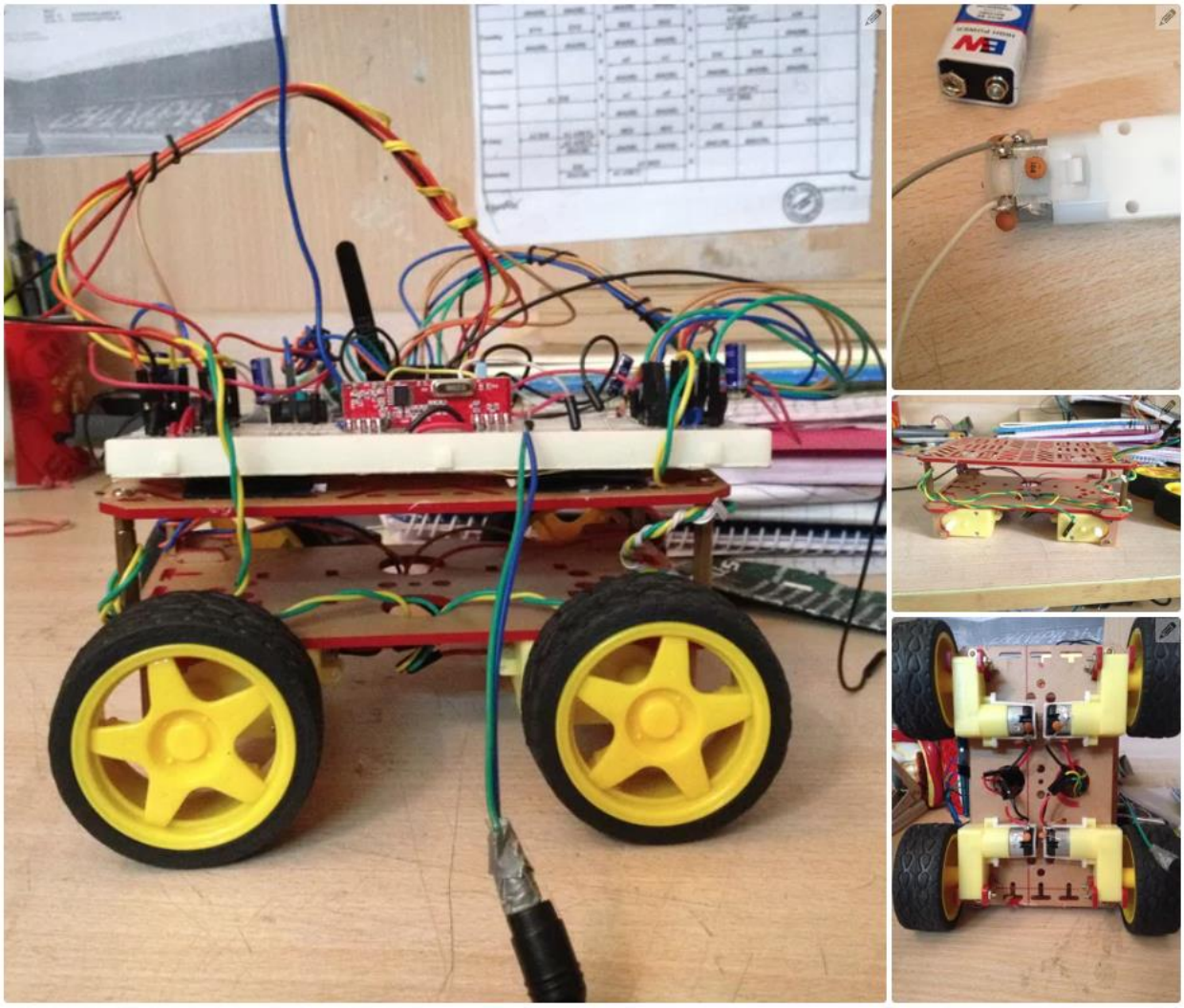
```
    case 'r': //if controller types r
    {
   char *msg2 = "r";
   digitalWrite(13, true); // Flash a light to show transmitting
   vw_send((uint8_t *)msg2, strlen(msg2));//send byte to the receiver
   vw_wait_tx(); // Wait until the whole message is gone
   digitalWrite(13, false);
   break;
    }
    default://if any other value is entered
    {

      Serial.println("wrong entry");//print wrong entry in the serial monitor
    }
   }// close switch-case
  }//close if
 }//close loop

 // End Of Code

// RECEIVER CODE

#include <VirtualWire.h>

//declaring pin nos to FIRST L293D
int en1 = 5;
 int en2 = 6;
 int in1 = 7;
 int in2 = 8;
 int in3 = 9;
 int in4 = 10;
 //declaring pin nos to SECOND L293D
 int en1o2 = 3;
 int en2o2 = 11;
 int in1o2 = 4;
 int in2o2 = 12;
 int in3o2 = 14;
 int in4o2 = 15;
 int motorPin[] = {3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15};//array for storing pin nos

void setup()
{
   Serial.begin(9600);    // Debugging only
   Serial.println("setup");

   // Initialise the IO and ISR
   vw_setup(2000);         // Bits per sec
   vw_set_rx_pin(2);    //Rx Data pin to Digital  Pin 2
   vw_rx_start();       // Start the receiver PLL running

   for (int i = 0; i < 12; i++)
   {
```

```
        pinMode(motorPin[i], OUTPUT);
   }
/*
This is basically what the above for loop does :-
pinMode(en1, OUTPUT);
pinMode(en2, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
pinMode(in3, OUTPUT);
pinMode(in4, OUTPUT);
pinMode(en1o2, OUTPUT);
pinMode(en2o2, OUTPUT);
pinMode(in1o2, OUTPUT);
pinMode(in2o2, OUTPUT);
pinMode(in3o2, OUTPUT);
pinMode(in4o2, OUTPUT);
*/
}//close setup

void loop()
{
   uint8_t buf[VW_MAX_MESSAGE_LEN];
   uint8_t buflen = VW_MAX_MESSAGE_LEN;

   if (vw_get_message(buf, &buflen)) // Non-blocking
   {
         int i;

      digitalWrite(13, true); // Flash a light to show received good message
         // Message with a good checksum received, dump it.
         for (i = 0; i < buflen; i++)
         {
          Serial.print(buf[i]);//print received command
      if(buf[i] == '1')//if button 1 is pressed.... i.e.forward buton
     {
      forward();//go forward
      Serial.println(" = forward");
     }
   if(buf[i] == '2')//if button 2 is pressed.... i.e.back buton
     {
      backward();//go backward
      Serial.println(" = backward");
     }
      if(buf[i] == '3')//if button 3 is pressed.... i.e.left buton
     {
      left();//go left
      Serial.println(" = left");
     }
      if(buf[i] == '4')//if button 4 is pressed.... i.e.right buton
     {
      right();//go right
      Serial.println(" = right");
```

```
      }
       if(buf[i] == '5')//if button 5 is pressed.... i.e.stop buton
       {
        stopMotor();//stop/brake
        Serial.println(" = stopped");
       }
              Serial.print(" ");
            }
            Serial.println("");
       digitalWrite(13, false);

   }//close if
}//close loop

void forward()
{
  digitalWrite(en1,HIGH);
  digitalWrite(en2,HIGH);
  digitalWrite(en1o2,HIGH);
  digitalWrite(en2o2,HIGH);
  digitalWrite(in1,HIGH);
  digitalWrite(in1o2,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in2o2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in3o2,LOW);
  digitalWrite(in4,HIGH);
  digitalWrite(in4o2,HIGH);
  }


void backward()
{
  digitalWrite(en1,HIGH);
  digitalWrite(en2,HIGH);
  digitalWrite(en1o2,HIGH);
  digitalWrite(en2o2,HIGH);
  digitalWrite(in1,LOW);
  digitalWrite(in1o2,LOW);
  digitalWrite(in2,HIGH);
  digitalWrite(in2o2,HIGH);
  digitalWrite(in3,HIGH);
  digitalWrite(in3o2,HIGH);
  digitalWrite(in4,LOW);
  digitalWrite(in4o2,LOW);

}

void left()
{
  digitalWrite(en1,HIGH);
  digitalWrite(en2,HIGH);
```

```
  digitalWrite(en1o2,HIGH);
  digitalWrite(en2o2,HIGH);
  digitalWrite(in1,LOW);
  digitalWrite(in1o2,LOW);
  digitalWrite(in2,LOW);
  digitalWrite(in2o2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in3o2,LOW);
  digitalWrite(in4,HIGH);
  digitalWrite(in4o2,HIGH);

}

void right()
{
  digitalWrite(en1,HIGH);
  digitalWrite(en2,HIGH);
  digitalWrite(en1o2,HIGH);
  digitalWrite(en2o2,HIGH);
  digitalWrite(in1,HIGH);
  digitalWrite(in1o2,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in2o2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in3o2,LOW);
  digitalWrite(in4,LOW);
  digitalWrite(in4o2,LOW);

}

void stopMotor()
{
  digitalWrite(en1,HIGH);
  digitalWrite(en2,HIGH);
  digitalWrite(en1o2,HIGH);
  digitalWrite(en2o2,HIGH);
  digitalWrite(in1,LOW);
  digitalWrite(in1o2,LOW);
  digitalWrite(in2,LOW);
  digitalWrite(in2o2,LOW);
  digitalWrite(in3,LOW);
  digitalWrite(in3o2,LOW);
  digitalWrite(in4,LOW);
  digitalWrite(in4o2,LOW);
}

 //End Of Code
```

## Output:



## Result:

Thus the execution of the RF Controlled or WiFi controlled Navigation bot is done successfully.

**EX. NO: 6**

# PICK AND PLACE ROBOT WITH OBJECT DETECTION

**AIM:**

To execute the Pick and place robot with Object Detection using Arduino IDE.

**Hardware Requirement:**

1. Robotic arm kit(which includes 4 geared motors, 4 wheels with track belts and arm gripper)
2. Arduino Uno R3 board (or any other arduino board)
3. HC05 or HC06 Bluetooth module
4. 2 X L293D motor driver IC
5. 12V Battery
6. Android Smartphone or Tablet
7. Android App and Arduino code

**Software Requirement:**

Arduino IDE

**Procedure:**

1. Gather all the necessary components for your robot, including a microcontroller (such as an Arduino), motor drivers, distance sensors (such as ultrasonic or infrared sensors), and motors.

2. Connect the motor drivers to the microcontroller. Depending on the type of motor driver you're using, you may need to connect power, ground, and control pins.

3. Connect the distance sensors to the microcontroller. Again, depending on the type of sensor you're using, you may need to connect power, ground, and signal pins.

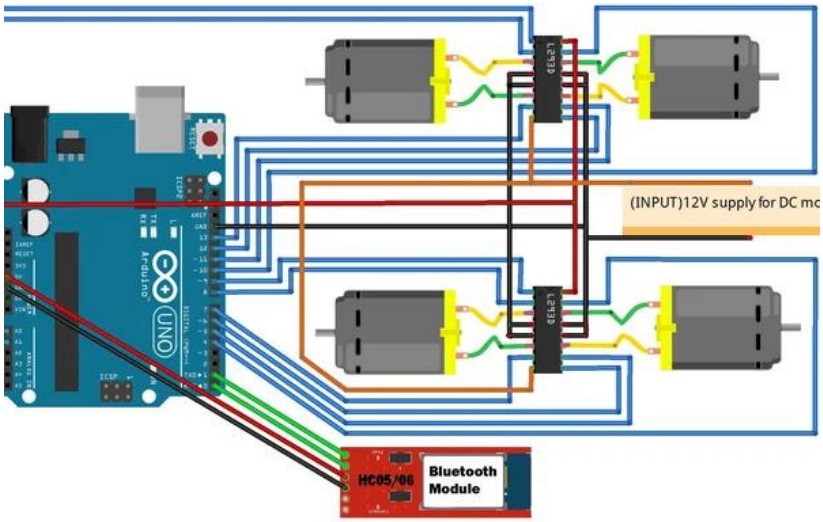
4. Connect the motors to the motor drivers.

5. Write code for your microcontroller that reads the distance sensors and controls the motors to keep the robot at a constant distance from the wall. This may involve using a PID controller or a simple proportional controller.

6. Test your robot and adjust your code as necessary.

7. Once you're satisfied with the performance of your robot, you can add additional features like obstacle avoidance or line following.

**Pinout Diagram:**




**Program Code:**

```
int motor1Pin1 = 8; // pin 2 on L293D IC
int motor1Pin2 = 7; // pin 7 on L293D IC
int enable1Pin = 9; // pin 1 on L293D IC
int motor2Pin1 = 5; // pin 10 on L293D IC
int motor2Pin2 = 4; // pin 15 on L293D IC
int enable2Pin = 6; // pin 9 on L293D IC
int motor3Pin1 = 14; // pin 2 on L293D IC
int motor3Pin2 = 13; // pin 7 on L293D IC
int enable3Pin = 15; // pin 1 on L293D IC
int motor4Pin1 = 11; // pin 10 on L293D IC
int motor4Pin2 = 10; // pin 15 on L293D IC
int enable4Pin = 12; // pin 9 on L293D IC
int state;
int flag=0;        //makes sure that the serial only prints once the state

void setup() {
   pinMode(motor1Pin1, OUTPUT),pinMode(motor1Pin2, OUTPUT),pinMode(enable1Pin, OUTPUT);
   pinMode(motor2Pin1, OUTPUT),pinMode(motor2Pin2, OUTPUT),pinMode(enable2Pin, OUTPUT);
   pinMode(motor3Pin1, OUTPUT),pinMode(motor3Pin2, OUTPUT),pinMode(enable3Pin, OUTPUT);
   pinMode(motor4Pin1, OUTPUT),pinMode(motor4Pin2, OUTPUT),pinMode(enable4Pin, OUTPUT);
   Stop();
   Serial.begin(9600);
}

void loop() {
   if(Serial.available() > 0){
     state = Serial.read();
     flag=0;
```

```
    }

    if (state == 'F'){
      Forward();
      if(flag == 0)Serial.println("Go Forward"),flag=1;
    }

    else if (state == 'B'){
      Reverse();
      if(flag == 0)Serial.println("Go Reverse"),flag=1;
    }

    else if (state == 'L'){
      Left();
      if(flag == 0)Serial.println("Go Left"),flag=1;
    }

    else if (state == 'R'){
      Right();
      if(flag == 0)Serial.println("Go Right"),flag=1;
    }

    else if (state == 'S'){
      Stop();
      if(flag == 0)Serial.println("STOP!"),flag=1;
    }

    else if (state == 'P'){
      Pick();
      if(flag == 0)Serial.println("Pick"),flag=1;
    }

    else if (state == 'Q'){
      Place();
      if(flag == 0)Serial.println("Place"),flag=1;
    }

    else if (state == 'H'){
      Hold();
      if(flag == 0)Serial.println("Hold"),flag=1;
    }

    else if (state == 'I'){
      Free();
      if(flag == 0)Serial.println("Unhold"),flag=1;
    }

  }

  void Forward(){
    digitalWrite(motor1Pin1, LOW);
    digitalWrite(motor1Pin2, HIGH);
```

```
  digitalWrite(motor2Pin1, HIGH);
  digitalWrite(motor2Pin2, LOW);
  digitalWrite(enable1Pin, HIGH);
  digitalWrite(enable2Pin, HIGH);
}

void Reverse(){
  digitalWrite(motor1Pin1, HIGH);
  digitalWrite(motor1Pin2, LOW);
  digitalWrite(motor2Pin1, LOW);
  digitalWrite(motor2Pin2, HIGH);
  digitalWrite(enable1Pin, HIGH);
  digitalWrite(enable2Pin, HIGH);
}

void Left(){
  digitalWrite(motor1Pin1, HIGH);
  digitalWrite(motor1Pin2, LOW);
  digitalWrite(motor2Pin1, HIGH);
  digitalWrite(motor2Pin2, LOW);
  digitalWrite(enable1Pin, HIGH);
  digitalWrite(enable2Pin, HIGH);
}

void Right(){
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, HIGH);
  digitalWrite(motor2Pin1, LOW);
  digitalWrite(motor2Pin2, HIGH);
  digitalWrite(enable1Pin, HIGH);
  digitalWrite(enable2Pin, HIGH);
}

void Pick(){
  digitalWrite(motor3Pin1, LOW);
  digitalWrite(motor3Pin2, HIGH);
  digitalWrite(enable3Pin, HIGH);
}

void Place(){
  digitalWrite(motor3Pin1, HIGH);
  digitalWrite(motor3Pin2, LOW);
  digitalWrite(enable3Pin, HIGH);
}

void Hold(){
  digitalWrite(motor4Pin1, LOW);
  digitalWrite(motor4Pin2, HIGH);
  digitalWrite(enable4Pin, HIGH);
}

void Free(){
```

```
  digitalWrite(motor4Pin1, HIGH);
  digitalWrite(motor4Pin2, LOW);
  digitalWrite(enable4Pin, HIGH);
}

void Stop(){    // Stop the motors
  digitalWrite(enable1Pin, LOW);
  digitalWrite(enable2Pin, LOW);
  digitalWrite(enable3Pin, LOW);
  digitalWrite(enable4Pin, LOW);
  digitalWrite(motor1Pin1, LOW);
  digitalWrite(motor1Pin2, LOW);
  digitalWrite(motor2Pin1, LOW);
  digitalWrite(motor2Pin2, LOW);
  digitalWrite(motor3Pin1, LOW);
  digitalWrite(motor3Pin2, LOW);
  digitalWrite(motor4Pin1, LOW);
  digitalWrite(motor4Pin2, LOW);
}
```

**Download apk from :**

https://github.com/ProjectsBySG/Pick-N-Place-Robot

**Output:**


**Result:**

Thus the execution of the Pick and place robot with Object Detection is done successfully.

**EX. NO: 7**

# WALL FOLLOWING BOT

**AIM:**

To execute the Wall Following bot using Arduino IDE.

**Hardware Requirement:**

1. Microcontroller
2. Motor drivers
3. Distance sensors
4. Power source
5. Chassis
6. Wheels
7. Jumper wires
8. Breadboard or PCB

**Software Requirement:**

Arduino IDE

**Procedure:**

1. Gather all the necessary components, including the microcontroller or microprocessor board, sensors, motor drivers, and power source.

2. Connect the motor drivers to the microcontroller board. The motor drivers control the movement of the robot. Typically, the motor drivers are connected to the microcontroller board through GPIO pins.

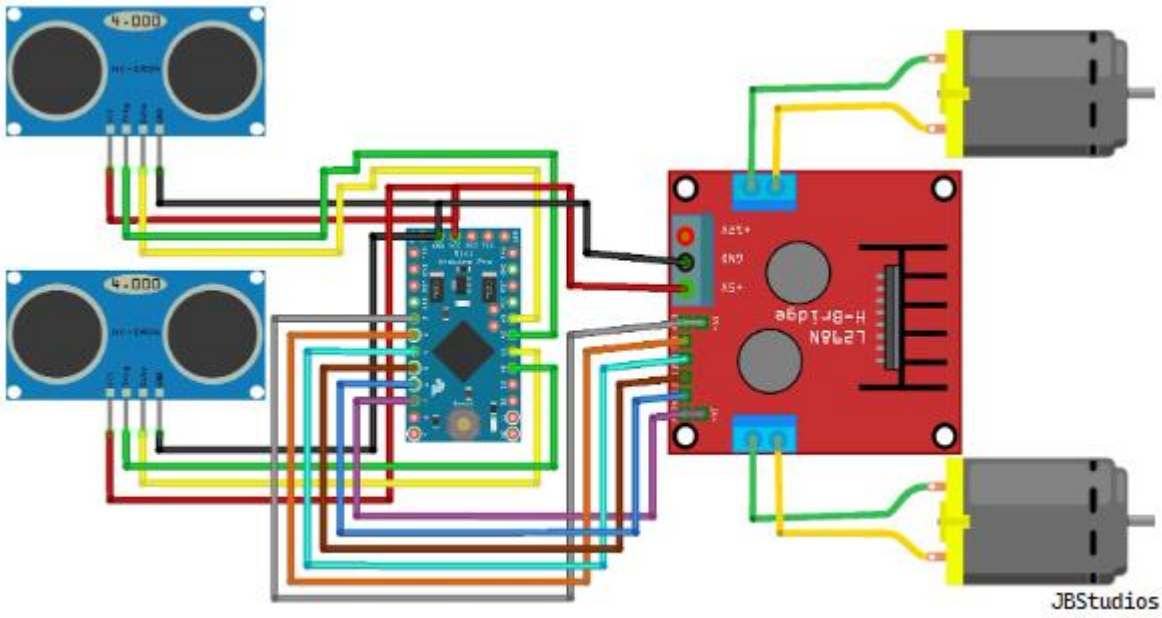
3. Connect the sensors to the microcontroller board. The sensors detect the distance between the robot and the wall, and they are typically connected to the microcontroller board through GPIO pins.

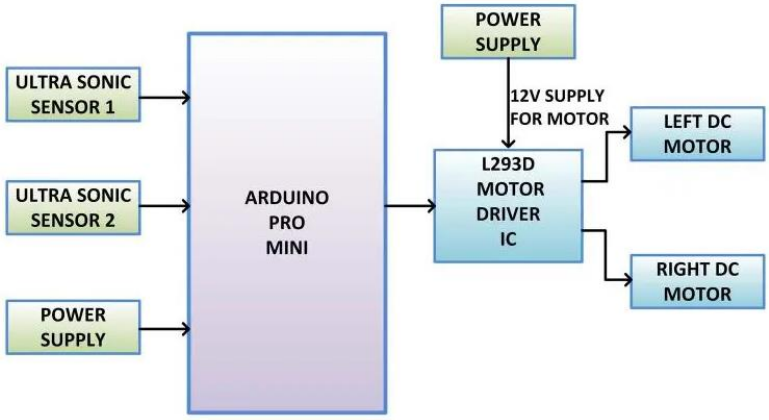
4. Connect the power source to the microcontroller board and the motor drivers. The power source should provide enough voltage and current to power the motors and the sensors.

5. Program the microcontroller board to read the sensor values and control the movement of the robot. The program should include algorithms to ensure that the robot follows the wall at a constant distance.

6. Test the robot and make any necessary adjustments to the program or hardware.

**Pinout Diagram:**




**Program Code:**

include
define SONAR_NUM 2
define MAX_DISTANCE 800
define PING_INTERVAL 33
define trigPin1 6
define echoPin15
define trigPin2 11
define echoPin210
define LM1 2
define LM2 3

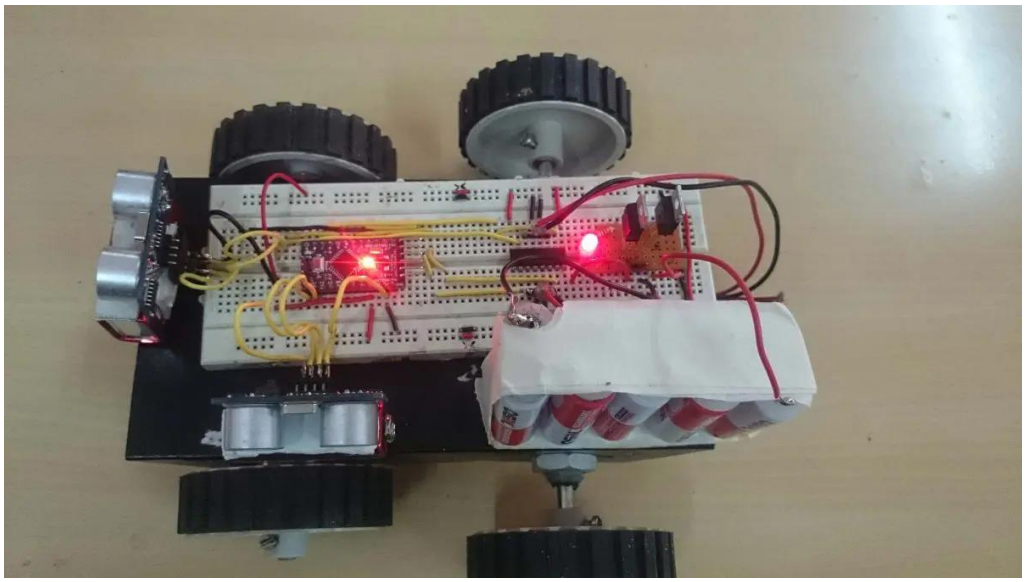
```
define RM1      7
define RM2      8
int rangeFront = 0;
int rangeWall  = 0;
NewPing sonar[SONAR_NUM] = {
  NewPing(trigPin1, echoPin1, MAX_DISTANCE),
  NewPing(trigPin2, echoPin2, MAX_DISTANCE)
}

void setup()
{
  Serial.begin(9600);
  int i;
  pinMode(LM1, OUTPUT);
  pinMode(LM2, OUTPUT);
  pinMode(RM1, OUTPUT);
  pinMode(RM2, OUTPUT);
  pinMode(en1, OUTPUT);
  pinMode(en2, OUTPUT);
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  digitalWrite(trigPin1, LOW);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  digitalWrite(trigPin2, LOW);
}

int toCloseWall = 1800;
int toFarWall = 2500;
int toCloseFront = 1000;

void loop()

{

Main:
  rangeFront = readRangeFront();
  Serial.print(rangeFront);
  Serial.print(" Front");
  Serial.println();
  rangeWall = readRangeWall();
  Serial.print(rangeWall);
  Serial.print(" Wall");
  Serial.println();


  if (rangeFront <= 400)
  {
    rangeFront = 3000;

  }
  if(rangeWall <= 400)
```

```
  {
    rangeWall = 3000;
  }
 if (rangeFront < toCloseFront)
 {
    delay(500);
    drive_backward();
    delay(500);
    Serial.print(" Drive Back");
    Serial.print(" Right Turn");
    Serial.println();
    delay(800);
    drive_forward();
    turn_left();
    delay(1700);
    goto Main;
  }

if(rangeWall > toCloseWall && rangeWall < toFarWall)
  {
    drive_forward();
    Serial.print(" Drive Forward");
    Serial.println();
    goto Main;
  }

  if (rangeWall < toCloseWall)
  {
    delay(100);
    turn_right();
    delay(500);
    Serial.print(" Turn Left");
    drive_forward();
    Serial.print(" Drive Forward");
    Serial.println();
    goto Main;
  }

  if (rangeWall > toFarWall)
  {
    delay(100);
    turn_left();
    Serial.print(" Turn Right");
    delay(500);
    drive_forward();
    Serial.print(" Drive Forward");
    Serial.println();
    goto Main;
  }
}

void motor_stop()
```

```
{
   digitalWrite(LM1, LOW);
   digitalWrite(LM2, LOW);
   digitalWrite(RM1, LOW);
   digitalWrite(RM2, LOW);
}

void drive_forward()
{
   digitalWrite(LM1, HIGH);
   digitalWrite(LM2, LOW);
   digitalWrite(RM1, HIGH);
   digitalWrite(RM2, LOW);
}


void drive_backward()
{
   digitalWrite(LM1, LOW);
   digitalWrite(LM2, HIGH);
   digitalWrite(RM1, LOW);
   digitalWrite(RM2, HIGH);
}

void turn_left()
{
   digitalWrite(LM1, HIGH);
   digitalWrite(LM2, LOW);
   digitalWrite(RM1, LOW);
   digitalWrite(RM2, LOW);
}

void turn_right()
{
   digitalWrite(LM1, LOW);
   digitalWrite(LM2, LOW);
   digitalWrite(RM1, HIGH);
   digitalWrite(RM2, LOW);
}

int readRangeFront()
{
  delay(50);
  unsigned rangeFront = sonar[0].ping();
  sonar[0].timer_stop();
  return rangeFront;
}

int readRangeWall()
{
  delay(50);
  unsigned rangeWall = sonar[1].ping();
```

```
  sonar[1].timer_stop();
  return rangeWall;
}
```

**Output:**



**Result:**

Thus the execution of the Wall Following robot is done successfully.

**EX. NO: 8**

# MAZE FOLLOWING ROBOT

**AIM:**

To execute the Maze Following Robot using Arduino IDE.

**Hardware Requirement:**

**Parts List:**

1. Arduino UNO
2. 12v DC motors (x2)
3. Wheels (x2)
4. Motor Driver (L298N)
5. Distance Sensor (UltraSonic)
6. Wires (Jumper wires 1-pin male-male)
7. 12v Battery (1000 mAh)

**Tools List:**

1. Battery Charger
2. Acrylic sheet
3. Soldering Iron
4. Wire cutter (optional)
5. Nylon Zip Wrap

**Software Requirement:**

Arduino IDE

**Procedure:**

1. Assemble the robot chassis according to the manufacturer's instructions.

2. Connect the motor driver board or H-bridge module to the microcontroller board using jumper wires. Make sure to connect the motor driver board's power and ground pins to the microcontroller board's power and ground pins.

3. Connect the motors to the motor driver board or H-bridge module. You may need to consult the manufacturer's instructions for the proper pin connections.

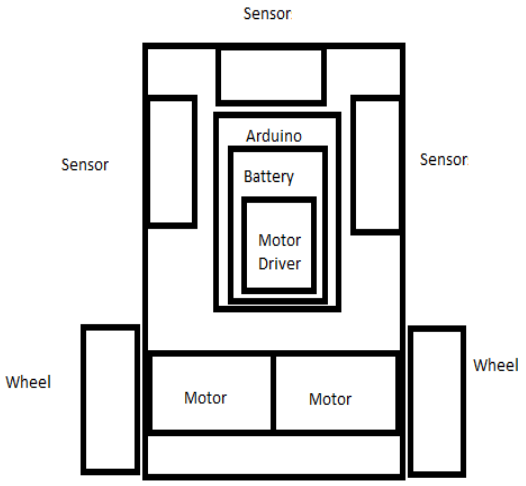
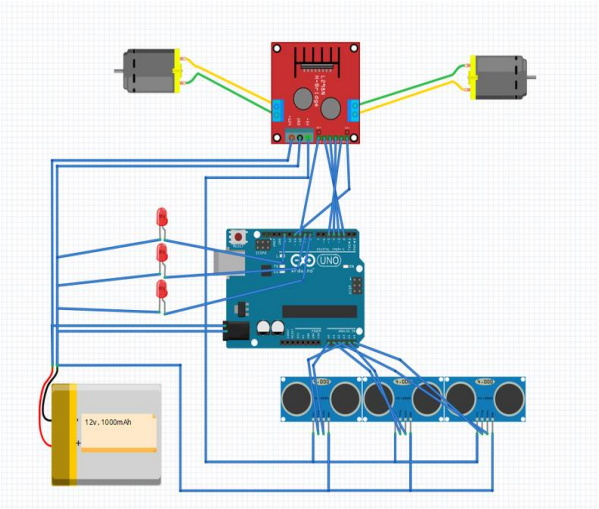
4. Connect the ultrasonic sensors or line sensors to the microcontroller board using jumper wires. Make sure to connect the sensors' power and ground pins to the microcontroller board's power and ground pins, and connect the sensor's signal pins to the microcontroller board's input pins.

5. Write the code to control the robot's movement and sensor readings. You may need to use a library for your microcontroller board to interface with the motor driver board or sensors.

6. Upload the code to the microcontroller board.

7. Power on the robot using a battery or power supply.

**Pinout Diagram:**

| Body Design | Wiring |
|---|---|
|   |  |

**Program Code:**

```
include <NewPing.h>
define TRIGGER_PINL  A3
define ECHO_PINL     A0
define MAX_DISTANCE 100
define TRIGGER_PINF  A4
define ECHO_PINF     A1
define TRIGGER_PINR  A5
define ECHO_PINR     A2

int dir;

define STOP 0
define FORWARD 1
define BACKWARD 2
define LEFT 3
define RIGHT 4
```

```
float P = 0.7 ;
float D = 0.5 ;
float I = 0.4 ;
float oldErrorP ;
float totalError ;
int offset = 5 ;

int wall_threshold = 13 ;
int left_threshold = 10 ;
int right_threshold = 10 ;
int front_threshold = 7 ;

boolean frontwall ;
boolean leftwall ;
boolean rightwall ;
boolean first_turn ;
boolean rightWallFollow ;
boolean leftWallFollow ;

int en1 = 2 ;
int en2 = 3 ;

int en3 = 4 ;
int en4 = 5 ;

int enA = 10 ;
int enB = 11 ;

int baseSpeed = 70 ;

int RMS ;
int LMS ;

int LED = 13 ;
int led1 = 8 ;
int led2 = 9 ;



NewPing sonarLeft(TRIGGER_PINL, ECHO_PINL, MAX_DISTANCE); // NewPing setup of pins and
maximum distance.
NewPing sonarRight(TRIGGER_PINR, ECHO_PINR, MAX_DISTANCE);
NewPing sonarFront(TRIGGER_PINF, ECHO_PINF, MAX_DISTANCE);

unsigned int pingSpeed = 30;
unsigned long pingTimer;   .

float oldLeftSensor, oldRightSensor, leftSensor, rightSensor, frontSensor, oldFrontSensor, lSensor, rSensor,
fSensor;
int TestNUM = 1  ;
void setup() {
```

```
  Serial.begin(115200);.

  for (int i = 2; i <= 13; i++) //For Motor Shield
    pinMode(i, OUTPUT);

  first_turn = false ;
  rightWallFollow = false ;
  leftWallFollow = false ;

}

void loop() {

  ReadSensors();
  walls();

  if ( first_turn == false )
{
    pid_start();
  }
  else if (leftWallFollow == true ) {
    PID(true) ;
  }
  else if (rightWallFollow == true ) {
    PID(false) ;
  }

  if (leftwall == true && rightwall == false && frontwall == true )
 {
    turnright();
    PID(false) ;

    if ( first_turn == false ) {

          right_threshold = right_threshold - offset ;
          left_threshold = left_threshold + offset ;


      first_turn = true ;
      rightWallFollow = true ;

      digitalWrite(led2 , LOW );
      digitalWrite(led1 ,HIGH );
    }
  }
   if (leftwall == false && rightwall == true && frontwall == true ) {

    turnleft();
    PID(true) ;

    if ( first_turn == false ) {
```

```
          left_threshold = left_threshold - offset ;
          right_threshold = right_threshold + offset ;

      first_turn = true ;
      leftWallFollow = true ;
      digitalWrite(LED , HIGH);

    }
  }
   if ( leftSensor == 0 || leftSensor > 100 && rightSensor == 0 || rightSensor > 100 && frontSensor == 0 ||
frontSensor > 100 ) {

    setDirection(STOP);
  }


  Serial.print(" Left : ");
  Serial.print(leftSensor);
  Serial.print(" cm ");
  Serial.print(" Right : ");
  Serial.print(rightSensor);
  Serial.print(" cm ");
  Serial.print(" Front : ");
  Serial.print(frontSensor);
  Serial.println(" cm ");
  Serial.print("error=");
  Serial.println(totalError);

}

void setDirection(int dir) {

  if ( dir == FORWARD ) {
    digitalWrite(en1, LOW);   // Left wheel forward
    digitalWrite(en2, HIGH);
    digitalWrite(en3, LOW);  // Right wheel forward
    digitalWrite(en4, HIGH);
  }
  else if ( dir == LEFT ) {
    digitalWrite(en1, HIGH);   // Left wheel forward
    digitalWrite(en2, LOW );
    digitalWrite(en3, LOW );  // Right wheel forward
    digitalWrite(en4, HIGH);
  }
  else if ( dir == RIGHT ) {
    digitalWrite(en1, LOW);   // Left wheel forward
    digitalWrite(en2, HIGH);
    digitalWrite(en3, HIGH);  // Right wheel forward
    digitalWrite(en4, LOW);
  }
  else if ( dir == STOP ) {
```

```
    digitalWrite(en1, HIGH);   // Left wheel forward
    digitalWrite(en2, HIGH );
    digitalWrite(en3, HIGH );  // Right wheel forward
    digitalWrite(en4, HIGH);
   }
   else if ( dir == BACKWARD ) {
    digitalWrite(en1, HIGH );   // Left wheel forward
    digitalWrite(en2, LOW );
    digitalWrite(en3, HIGH );  // Right wheel forward
    digitalWrite(en4, LOW );
   }
}


void ReadSensors() {

  leftSensor = sonarLeft.ping_median(TestNUM);     //accurate but slow
  rightSensor = sonarRight.ping_median(TestNUM);     //accurate but slow
  frontSensor = sonarFront.ping_median(TestNUM);     //accurate but slow

  leftSensor = sonarLeft.convert_cm(leftSensor);
  rightSensor = sonarRight.convert_cm(rightSensor);
  frontSensor = sonarFront.convert_cm(frontSensor);

  lSensor = sonarLeft.ping_cm(); //ping in cm
  rSensor = sonarRight.ping_cm();
  fSensor = sonarFront.ping_cm();


  leftSensor = (lSensor + oldLeftSensor) / 2; //average distance between old & new readings to make the change
smoother
  rightSensor = (rSensor + oldRightSensor) / 2;
  frontSensor = (fSensor + oldFrontSensor) / 2;


  oldLeftSensor = leftSensor; // save old readings for movement
  oldRightSensor = rightSensor;
  oldFrontSensor = frontSensor;

}
void pid_start() {

  float errorP = leftSensor - rightSensor ;
  float errorD = errorP - oldErrorP;
  float errorI = (2.0 / 3.0) * errorI + errorP ;

  totalError = P * errorP + D * errorD + I * errorI ;

  oldErrorP = errorP ;

  RMS = baseSpeed + totalError ;
  LMS = baseSpeed - totalError ;
```

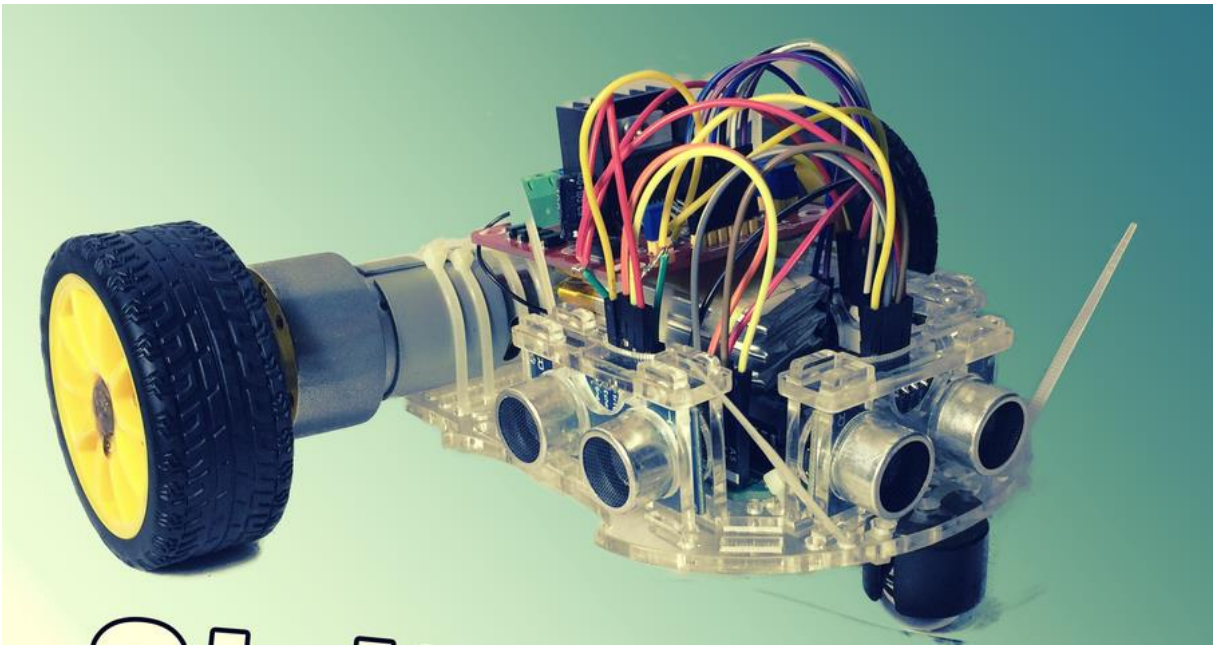
```
  if (RMS < 0) {

    RMS = map(RMS , 0 , -255, 0, 255);

    analogWrite(enA , RMS);
    analogWrite(enB , LMS);

    setDirection(RIGHT);

  }
  else if (LMS < 0) {
    LMS = map(LMS , 0 , -255, 0, 255);


    analogWrite(enA , RMS);
    analogWrite(enB , LMS);

    setDirection(LEFT);
  }
  else {

    analogWrite(enA , RMS);
    analogWrite(enB , LMS);

    setDirection(FORWARD);
  }

}

void PID( boolean left ) {

  if (left == true ) {

    float errorP = leftSensor - rightSensor - offset ;
    float errorD = errorP - oldErrorP;
    float errorI = (2.0 / 3) * errorI + errorP ;


    totalError = P * errorP + D * errorD + I * errorI ;

    oldErrorP = errorP ;


    RMS = baseSpeed + totalError ;
    LMS = baseSpeed - totalError ;

    //  if(RMS < -255) RMS = -255; if(RMS > 255)RMS = 255 ;
    //  if(LMS < -255) LMS = -255;  if(LMS > 255)LMS = 255 ;


    if (RMS < 0) {
```

```
    RMS = map(RMS , 0 , -255, 0, 255);

    analogWrite(enA , RMS);
    analogWrite(enB , LMS);

    setDirection(RIGHT);

   }
   else if (LMS < 0) {
    LMS = map(LMS , 0 , -255, 0, 255);


    analogWrite(enA , RMS);
    analogWrite(enB , LMS);

    setDirection(LEFT);
   }
   else {

    analogWrite(enA , RMS);
    analogWrite(enB , LMS);

    setDirection(FORWARD);
   }

 }
 else {

   float errorP = leftSensor - rightSensor + offset ;
   float errorD = errorP - oldErrorP;
   float errorI = (2.0 / 3) * errorI + errorP ;

   totalError = P * errorP + D * errorD + I * errorI ;

   oldErrorP = errorP ;


   RMS = baseSpeed + totalError ;
   LMS = baseSpeed - totalError ;

   //  if(RMS < -255) RMS = -255; if(RMS > 255)RMS = 255 ;
   //  if(LMS < -255) LMS = -255;  if(LMS > 255)LMS = 255 ;


   if (RMS < 0) {

    RMS = map(RMS , 0 , -255, 0, 255);

    analogWrite(enA , RMS);
    analogWrite(enB , LMS);
```

```
      setDirection(RIGHT);

    }
    else if (LMS < 0) {
      LMS = map(LMS , 0 , -255, 0, 255);


      analogWrite(enA , RMS);
      analogWrite(enB , LMS);

      setDirection(LEFT);
    }
    else {

      analogWrite(enA , RMS);
      analogWrite(enB , LMS);

      setDirection(FORWARD);
    }

  }

}

void walls() {

  if ( leftSensor < wall_threshold ) {
    leftwall = true ;
  }
  else {
    leftwall = false ;
  }

  if ( rightSensor < wall_threshold ) {
    rightwall = true ;
  }
  else {
    rightwall = false ;

  } if ( frontSensor < front_threshold ) {
    frontwall = true ;
  }
  else {
    frontwall = false ;
  }

}

void turnright() {

  LMS = baseSpeed ;
```

```
  RMS = LMS * rightSensor / ( rightSensor + 11 ) ;

}

void turnleft() {

  RMS = baseSpeed ;
  LMS = RMS * leftSensor / ( leftSensor + 11 ) ;

}
```

**Output:**



**Result:**

Thus the execution of the Maze Following robot is done successfully.

**EX. NO: 9**

# FORWARD AND REVERSE KINEMATICS BASED EXPERIMENT USING OPEN SOURCE PLATFORMS

**AIM:**

To execute the Forward and reverse kinematics based experiment using open source platforms using Arduino IDE.

**Hardware Requirement:**

1. Arduino UNO
2. Servo
3. Jumper wires

**Software Requirement:**

Arduino IDE

**Procedure:**

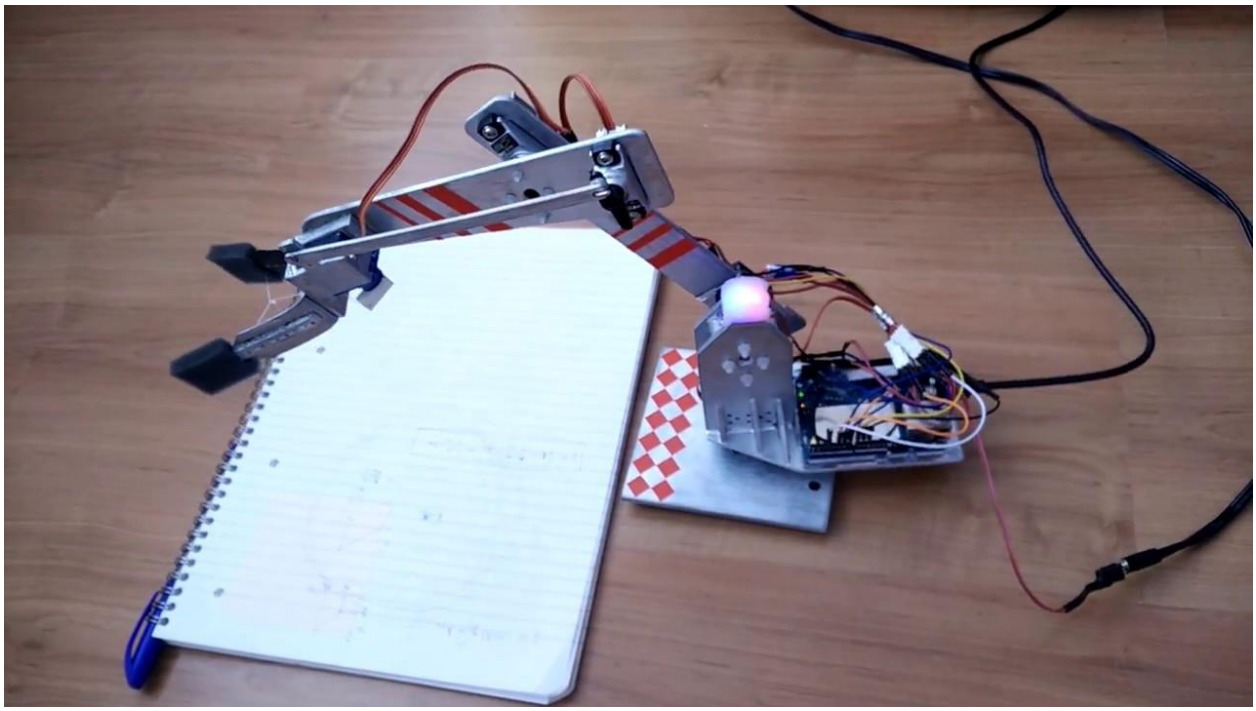
1. Arduino board (e.g., Arduino Uno or Arduino Mega)
2. Two servo motors compatible with your Arduino board
3. Jumper wires for making connections
4. Make the physical connections:
    a. Connect the power supply (usually 5V) to the VCC pin of the servo motors.
    b. Connect the ground (GND) pin of the servo motors to the GND pin of the Arduino.
    c. Connect the signal pin of the first servo motor to a digital pin on the Arduino (e.g., pin 9).
    d. Connect the signal pin of the second servo motor to another digital pin on the Arduino (e.g., pin 10).

**Program Code:**

```
#include <Servo.h>
#include <math.h>

// Define servo pin numbers
const int servoPin1 = 9;
const int servoPin2 = 10;

// Define servo objects
Servo servo1;
Servo servo2;

// Define arm dimensions (link lengths in cm)
```

```
const float link1 = 10.0;
const float link2 = 10.0;

// Define target position (end-effector coordinates)
float targetX = 10.0;
float targetY = 5.0;

// Forward kinematics function
void forwardKinematics(float theta1, float theta2, float& x, float& y) {
  x = link1 * cos(theta1) + link2 * cos(theta1 + theta2);
  y = link1 * sin(theta1) + link2 * sin(theta1 + theta2);
}

// Inverse kinematics function
void inverseKinematics(float x, float y, float& theta1, float& theta2) {
  float c2 = (pow(x, 2) + pow(y, 2) - pow(link1, 2) - pow(link2, 2)) / (2 * link1 * link2);
  float s2 = sqrt(1 - pow(c2, 2));
  theta2 = atan2(s2, c2);
  theta1 = atan2(y, x) - atan2(link2 * s2, link1 + link2 * c2);
}

// Setup function
void setup() {
  // Attach servos to pins
  servo1.attach(servoPin1);
  servo2.attach(servoPin2);

  // Set initial angles for the servos
  servo1.write(0);
  servo2.write(0);

  // Wait for servo initialization
  delay(1000);
}

// Loop function
void loop() {
  // Calculate inverse kinematics for the target position
  float theta1, theta2;
  inverseKinematics(targetX, targetY, theta1, theta2);

  // Convert joint angles to servo positions
  int servoPos1 = map(theta1, -PI, PI, 0, 180);
  int servoPos2 = map(theta2, -PI, PI, 0, 180);

  // Set servo positions
  servo1.write(servoPos1);
  servo2.write(servoPos2);

  // Wait for the servos to reach the desired positions
  delay(1000);
}
```

**Output:**



**Result:**

Thus the execution of the Forward and reverse kinematics based experiment using open source platforms is done successfully.

**EX. NO: 10**

# COMPUTER VISION BASED ROBOTIC TASKS EXECUTION

**AIM:**

To execute the Computer Vision based on robotics tasks using Arduino.

**Hardware Requirement:**

1. Arduino UNO
2. Servo
3. An RGB color sensor (TCS34725)
4. A webcam or camera module

**Software Requirement:**

Arduino IDE

**Procedure:**

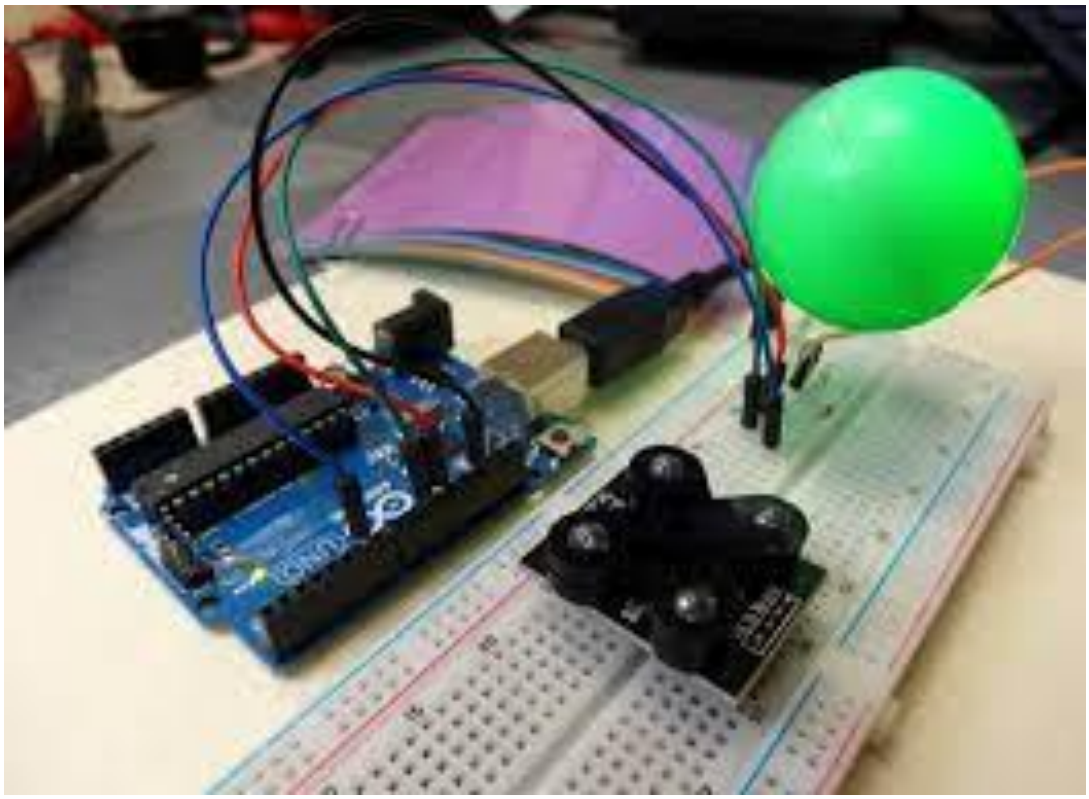
1. The code assumes that the RGB color sensor (TCS34725) is set up and working correctly.
2. The code assumes that the webcam or camera module is connected and accessible by OpenCV.
3. Make sure the Arduino board is properly connected to the computer running the code.
4. Ensure that the correct pins are used for the servo motor and RGB color sensor as defined in the code (SERVO_PIN, TCS34725_INTEGRATIONTIME_50MS, TCS34725_GAIN_4X).
5. Adjust the lower_red and upper_red values (HSV color range) according to your requirements for detecting the desired color.
6. Set the desired frame width and height for video capture using the cap.set() function.
7. You may need to install additional libraries or drivers depending on your specific hardware setup.

**Program Code:**

```
#include <Servo.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TCS34725.h>
#include <opencv2/opencv.hpp>

#define SERVO_PIN 3
#define MIN_ANGLE 0
#define MAX_ANGLE 180
#define SERVO_DELAY 10
#define MIN_SATURATION 50

Servo servo;
```

```
Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,
TCS34725_GAIN_4X);
cv::VideoCapture cap(0);
cv::Scalar lower_red = cv::Scalar(0, 150, 150);
cv::Scalar upper_red = cv::Scalar(10, 255, 255);

void setup() {
  servo.attach(SERVO_PIN);
  tcs.begin();
  cap.set(cv::CAP_PROP_FRAME_WIDTH, 640);
  cap.set(cv::CAP_PROP_FRAME_HEIGHT, 480);
}

void loop() {
  cv::Mat frame, hsv, mask;
  bool success = cap.read(frame);

  if (success) {
    cv::cvtColor(frame, hsv, cv::COLOR_BGR2HSV);
    cv::inRange(hsv, lower_red, upper_red, mask);

    int x = 0, y = 0, count = 0;
    for (int i = 0; i < mask.rows; i++) {
      for (int j = 0; j < mask.cols; j++) {
        if (mask.at<uchar>(i, j) > 0) {
          x += j;
          y += i;
          count++;
        }
      }
    }

    if (count > 0) {
      x /= count;
      y /= count;
      int angle = map(x, 0, mask.cols, MIN_ANGLE, MAX_ANGLE);

      if (tcs.read16(TCS34725_CDATA) > MIN_SATURATION) {
        servo.write(angle);
        delay(SERVO_DELAY);
      }
    }
  }
}
```

**Output:**



**Result:**

Thus the execution of the Computer Vision based on robotics tasks is done successfully.