A Report on Arduino - Powered Precision Robotic Limb with Potentiometer Control and Flex Control

Team Members:

1) Shantanu Swami – CSD (1DT21CG037)
2) Kunal Keshav – CSD (1DT21CG024)
3) Yuvraj Dadhich – CSD (1DT21CG049)

ABSTRACT

In today's world there is an increasing need to create artificial arms for different inhuman situations where human interaction is difficult or impossible. They may involve taking readings from an active volcano to diffusing a bomb. Here we propose to build a robotic arm controlled by natural human arm movements whose data is acquired through the use of accelerometers. For proper control mechanism and to reduce the amount of noise coming in from the sensor's, proper averaging algorithm is used for smoothening the output of the accelerometer. The development of this arm is based on ATmega32 and ATmega640 platform along with a personal computer for signal processing, which will all be interfaced with each other using serial communication. Finally, this prototype of the arm may be expected to overcome the problem such as placing or picking hazardous objects or non-hazardous objects that are far away from the user.

CONTENTS

1. INTRODUCTION

1.1 Introduction

Nowadays, robots are increasingly being integrated into working tasks to replace humans specially to perform the repetitive task. In general, robotics can be divided into two areas, industrial and service robotics. International Federation of Robotics (IFR) defines a service robot as a robot which operates semi- or fully autonomously to perform services useful to the wellbeing of humans and equipment, excluding manufacturing operations. These robots are currently used in many fields of applications including office, military tasks, hospital operations, dangerous environment and agriculture. Besides, it might be difficult or dangerous for humans to do some specific tasks like picking up explosive chemicals, defusing bombs or in worst case scenario to pick and place the bomb somewhere for containment and for repeated pick and place action in industries. Therefore, a robot can be replaced human to do work.

1.2 Robotic arm definition

A robotic arm is a robot manipulator, usually programmable, with similar functions to a human arm. The links of such a manipulator are connected by joints allowing either rotational motion (such as in an articulated robot) or translational (linear) displacement. The links of the manipulator can be considered to form a kinematic chain. The business end of the kinematic chain of the manipulator is called the end effectors and it is analogous to the human hand. The end effectors can be designed to perform any desired task such as welding, gripping, spinning etc., depending on the application. The robot arms can be autonomous or controlled manually and can be used to perform a variety of tasks with great accuracy. The robotic arm can be fixed or mobile (i.e. wheeled) and can be designed for industrial or home applications.

This report deals with a robotic arm whose objective is to imitate the movements of a human arm using accelerometers as sensors for the data acquisition of the natural arm movements. This method of control allows greater flexibility in controlling the robotic arm rather than using a controller where each actuator is controlled separately. The processing unit takes care of each actuator's control signal according to the inputs from accelerometer, in order to replicate the movements of the human arm. Figure 1 shows the block diagram representation of the system to be designed and implemented.

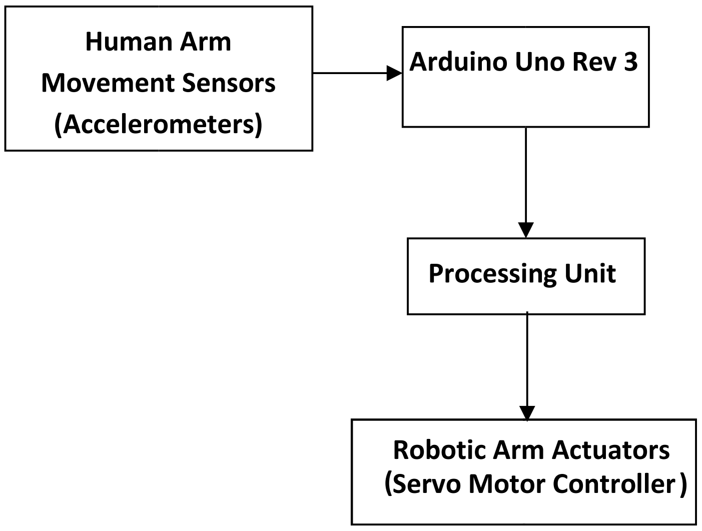

Human Arm Movement Sensors (Accelerometers)
Arduino Uno Rev 3
Processing Unit
Robotic Arm Actuators (Servo Motor Controller)

Fig1. Block Diagram Representation of the Proposed Robotic Arm System

1.3 Literature Review

There are various ways in which a robotic arm may be controlled. In the past there have been many researchers working to control robotic arm through computer terminals, Joysticks, even interfacing them with the internet so they can be controlled from anywhere in the world. Usually most of the robotic arms are controlled by a central controller which makes uses of values taken in from the terminal that are entered by the user at the terminal to move the arm to a particular coordinate in space. This makes the control very difficult as the control values of the motors are very difficult to predict to achieve a particular movement. This is easily achieved by our project.

This Project represents a simple potentiometer controlled robotic arm using Arduino Uno Rev3 embedded system as the core of this robot and also a computer to interface the robot with the sensors. The robot does not require training because the robotic arm is fully controlled by the user. This interfacing is done using wired communication but it can easily be switched to wireless with ease.

1.4 Project Overview

In this Project, the hardware and software function are combined to make the system reliable. The Arduino Uno Rev3 will be interfacing the robot with the actuators i.e., servo motors which will control the movement of the robot respectively. The chapter that follows describe the hardware (Chapter 2), which is followed by the description of the software being used (Chapter 3) Chapter 4 describes the implementation of the project and Chapter 5 concludes the discussion followed by the future scope of the project.

2. HARDWARE DESIGN AND DESCRIPTION

This chapter describes the hardware that is being used in the project.

2.1 Hardware Requirements

1. Servo Motors (Actuator)
2. Arduino Uno Rev3 (Controller)
3. Jumper Wires
4. Potentiometer 10k Ohms
5. 9V Constant DC Power Supply
6. Breadboard
7. Frame (Sunboard)
8. 1.5V Constant DC Power Supply

2.2 Servo Motors

Servo motors are a type of electromechanical actuators that do not rotate continuously like DC/AC or stepper motors; rather, they are used to position and hold some object. They are used where continuous rotation is not required so they are not used to drive wheels (unless a servo is modified). In contrast they are used where something is needed to move to particular position and then stopped and hold there. Most common use is to position the rudder of aircrafts and boats etc. The servo can be commanded to rotate to a particular angle (say 30) and then hold its position there. Servos also employ a feedback mechanism, so it can sense an error in its positioning and correct it. This is called servomechanism. Say if you ask servo to go and lock itself to 30 degrees and then try to rotate it with your hand, the servo will try hard and its best to overcome the force and keep servo locked in its specified angle. Controlling a servo is easy by using a microcontroller, no external driver like h-bridge etc. are required. Just a control signal is needed to be feed to the servo to position it in any specified angle. The frequency of the control signal is 50 Hz (i.e., the period is 20ms) and the width of positive pulse controls the angle. We can use the AVR microcontrollers PWM feature to control servo motors. In this way the PWM with automatically generate signals to lock servo and the CPU is free to do other tasks.

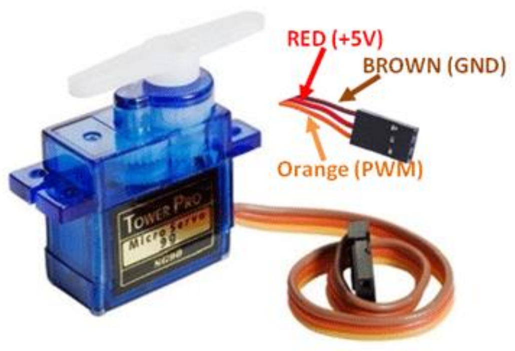


Fig.2.2. (a) Micro Servo Motor

2.3 Arduino Uno Rev3

Arduino Uno Rev 3 is an open-source microcontroller board based on the ATmega328P microcontroller. It is one of the most popular boards in the Arduino family, widely used in many applications such as robotics, home automation, and Internet of Things (IoT) projects.

The board has 14 digital input/output pins, 6 analog inputs, and a 16 MHz quartz crystal oscillator. It also has a USB interface for programming and power supply, and an ICSP header for programming with an external programmer.

The board is compatible with a wide range of shields, which are add-on boards that extend its functionality. It can be programmed using the Arduino Software (IDE) using a simplified version of C++ language.

Overall, the Arduino Uno Rev 3 is a versatile and easy-to-use microcontroller board suitable for beginners and experienced developers alike.

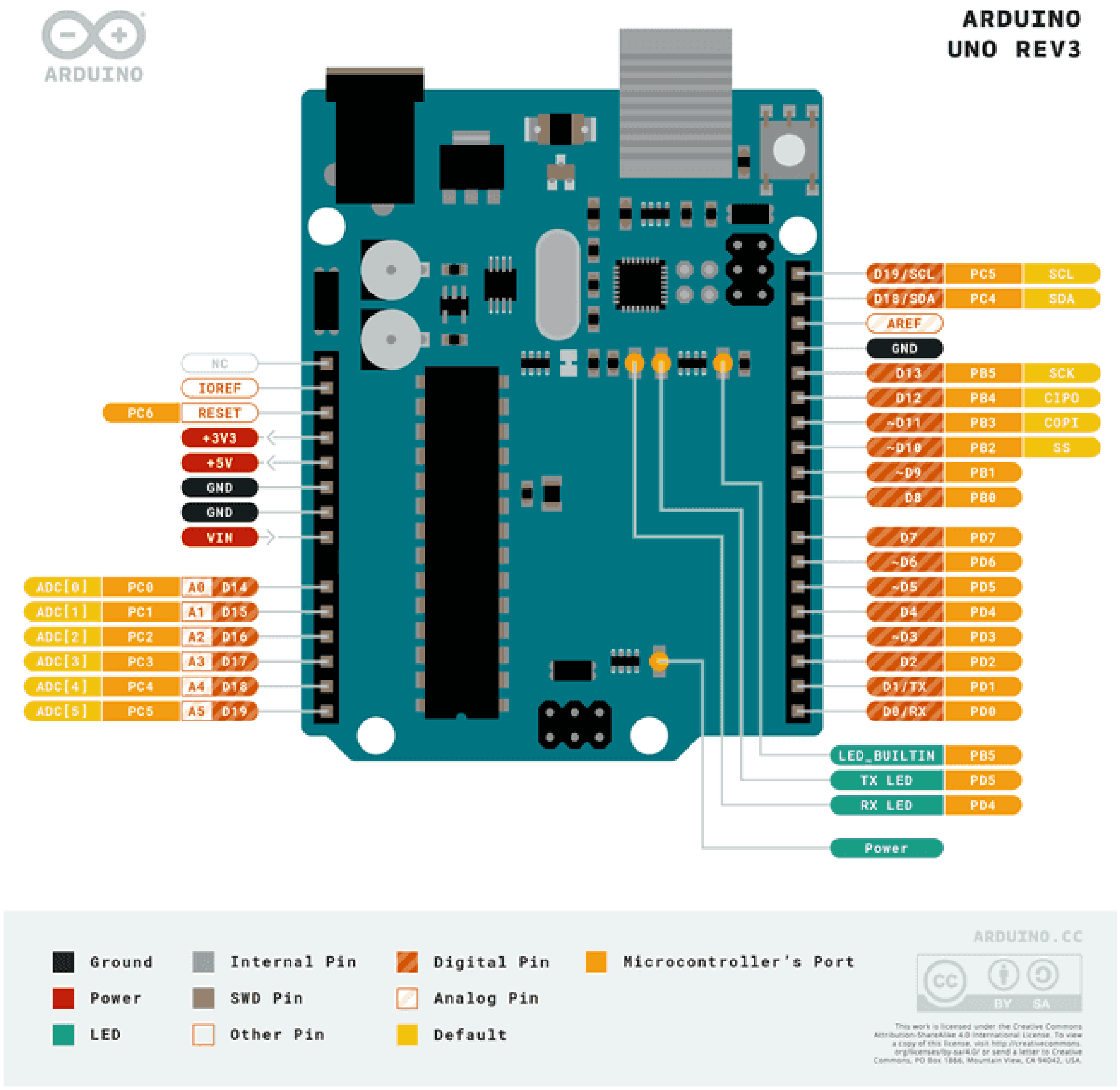


Fig.2.3. (a) Pinout Diagram

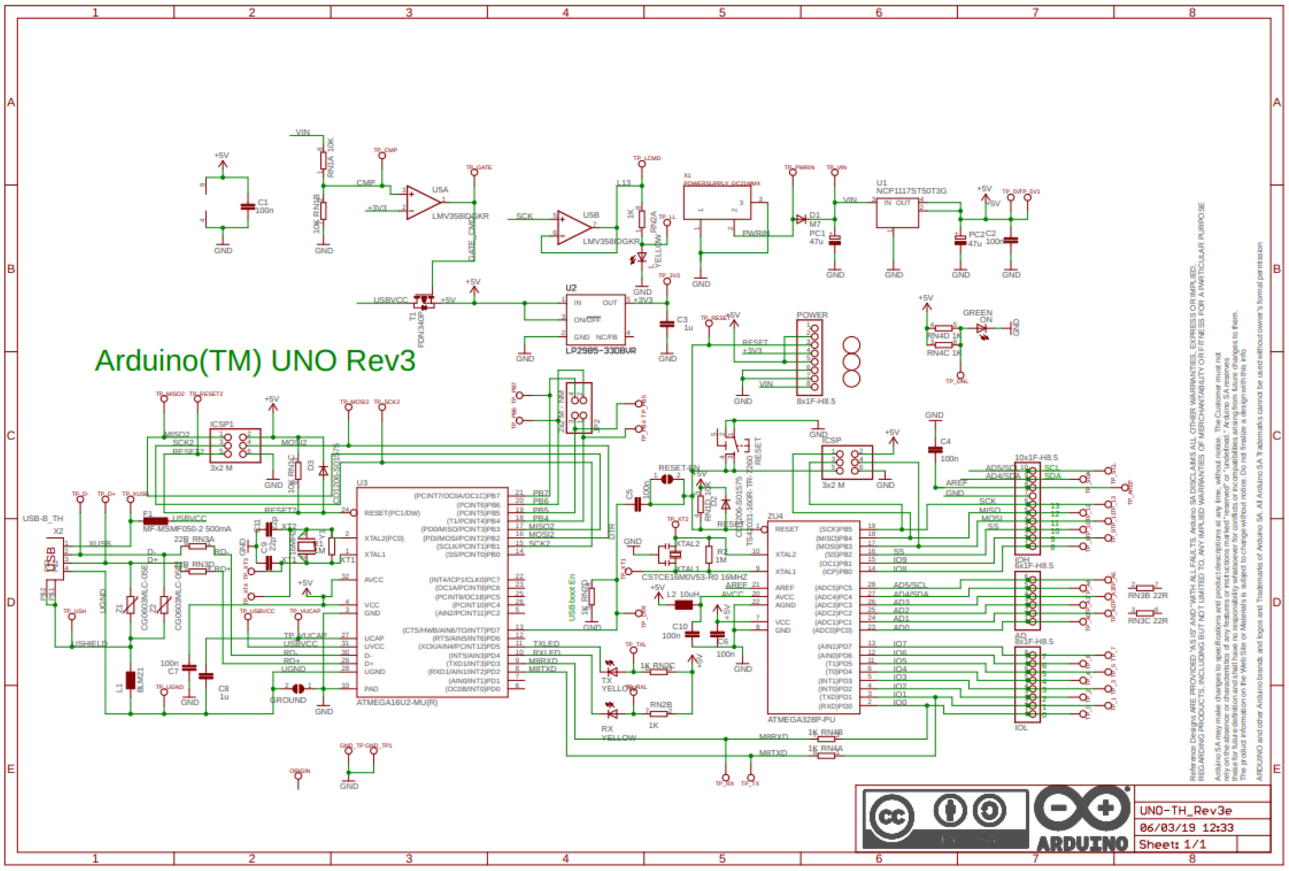


Fig.2.3. (b) Schematic of the Board

2.4 Potentiometer

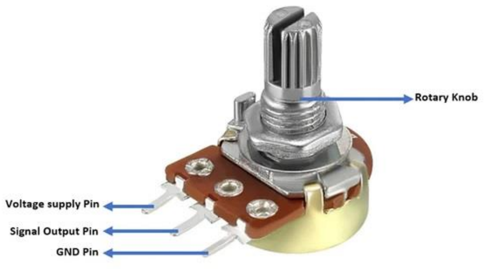
A potentiometer, also known as a "pot," is an electronic component used to vary the resistance in a circuit. It consists of a resistor with a movable contact that can be adjusted to change the resistance.

Potentiometers are commonly used in audio equipment, such as volume controls and tone controls, to adjust the level or frequency response of a signal. They are also used in variable power supplies and as sensors in certain applications.

Potentiometers come in a variety of sizes, shapes, and resistance values. Linear potentiometers have a linear change in resistance as the knob is turned, while logarithmic or audio taper potentiometers have a more gradual change in resistance, which is more appropriate for audio applications.

Potentiometers can also be classified by their number of turns. Single-turn potentiometers have a limited range of adjustment, while multi-turn potentiometers have a greater range of adjustment but require more turns to go from one end of the range to the other.

Potentiometers can wear out over time, especially if they are frequently adjusted. This can cause scratchy or intermittent behaviour, and may require replacement of the potentiometer.



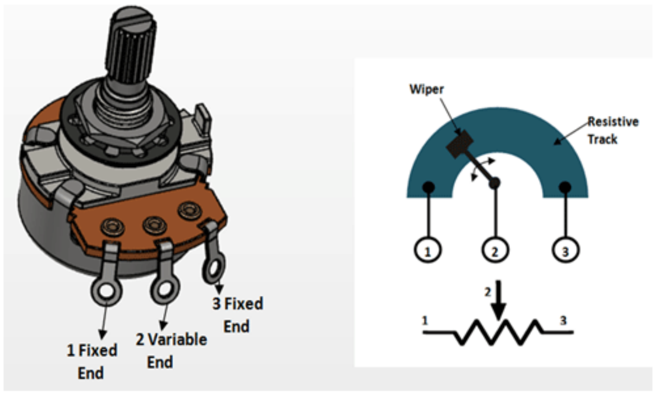


Fig.2.4. (a) Potentiometer Pinout

2.5 Jumper Wires

Jumper wires are essential components in Arduino prototyping. They are used to connect the Arduino board to other electronic components like sensors, LEDs, and motors.

Jumper wires are typically made up of a thin, flexible wire with pins on each end that can be easily plugged into a breadboard or directly connected to an electronic component. They come in different lengths, colours, and male/female configurations, which allow for easy identification and organization of connections.

To use jumper wires in an Arduino project, connect one end of the wire to a pin on the Arduino board and the other end to the corresponding pin on the component or breadboard. This allows the Arduino to send and receive data and control the attached component.

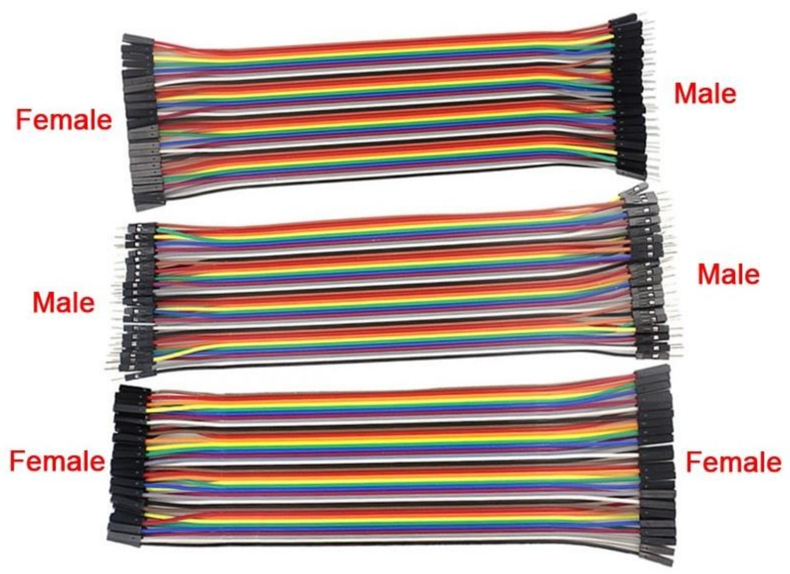


Fig.2.5. (a) Jumper Wires

2.6 Breadboard

A breadboard is a device used to build and prototype electronic circuits without the need for soldering. It consists of a plastic board with a grid of holes into which electronic components such as resistors, capacitors, and ICs can be inserted and connected together using wires. The holes are typically arranged in rows and columns, with each row and column electrically connected internally.

Breadboards are useful for quickly testing and prototyping circuits because components can be easily inserted, moved, and removed without damaging them. They are also reusable, as components can be taken out and used in other circuits.

Breadboards come in different sizes and shapes, with the most common being the standard or full-size breadboard. Mini breadboards and stripboards are also available, which are smaller and more compact. Some breadboards also come with built-in power rails and voltage regulators, which can simplify circuit prototyping.

Overall, breadboards are an essential tool for anyone interested in electronics or electrical engineering, as they allow for quick and easy experimentation and testing of circuits.

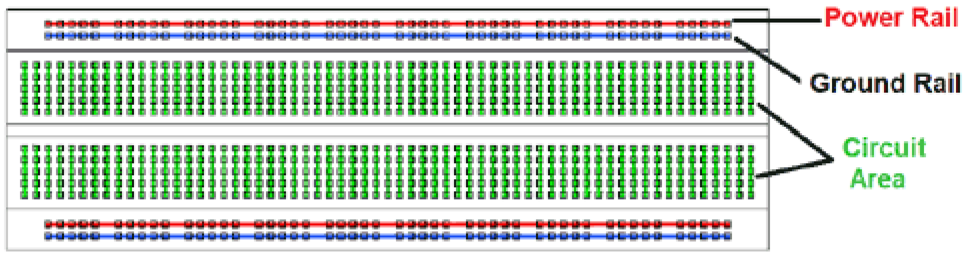


Fig.2.6. (a) Breadboard

2.7 Frame (Sunboard)

Sunboard is a brand name for a type of foam board that is commonly used for advertising and display purposes. It is made of expanded polystyrene foam (EPS) core that is sandwiched between two layers of paper or plastic. The foam core provides lightweight and rigid support while the outer layers provide a smooth surface that can be printed on or laminated with graphics. Sunboard is also known for its resistance to moisture, chemicals, and UV radiation, which makes it suitable for outdoor use. Sunboard comes in different thicknesses, sizes, and colors, making it a versatile material for a wide range of applications such as signage, point-of-purchase displays, and exhibition stands.

Fig.2.6. (a) Frame (Sunboard)

2.8 Constant 9V DC Power Supply

A constant 9V DC power supply is an electrical device that provides a continuous and stable output of 9 volts direct current (DC) to power electronic devices. It is commonly used to power various types of devices such as musical instruments, amplifiers, and electronic toys. A constant 9V DC power supply can be designed using a transformer to step down the voltage from the mains AC power supply to a lower AC voltage, which is then rectified to DC using a diode bridge. The DC output is then regulated using a voltage regulator circuit to maintain a constant 9V output. There are different types of constant 9V DC power supplies, such as linear power supplies and switching power supplies, each with their own advantages and disadvantages. The choice of power supply depends on the specific application and the required performance characteristics, such as efficiency, noise, and stability.

Fig.2.8. (a) Constant 9V DC Power Supply

2.9 Flex Sensor

A **flex sensor** or **bend sensor** is a sensor that measures the amount of deflection or bending. Usually, the sensor is stuck to the surface, and resistance of sensor element is varied by bending the surface. Since the resistance is directly proportional to the amount of bend it is used as goniometer, and often called flexible potentiometer.

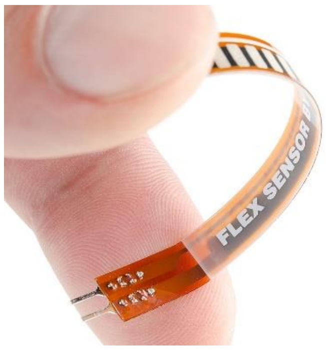


Fig.2.9. (a) Flex Sensor

2.10 MPU 6050 3-Axis Gyroscope

The MPU-6050 IMU (Inertial Measurement Unit) is a 3-axis accelerometer and 3-axis gyroscope sensor. The accelerometer measures the gravitational acceleration, and the gyroscope measures the rotational velocity. Additionally, this module also measures temperature. This sensor is ideal for determining the orientation of a moving object.

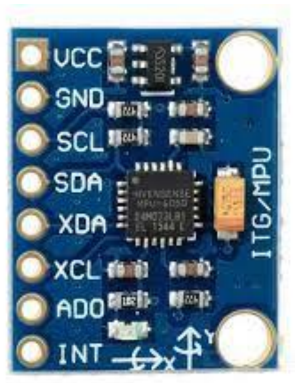


Fig.2.10. (a) MPU 6050 3-Axis Gyroscope

3. SOFTWARE DESIGN AND DESCRIPTION

This chapter describes the software that is being used in the project.

3.1 Software Requirements

1. Arduino IDE
2. Fritzing

3.2 Source Code (Potentiometer)

```
#include <Servo.h>

Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;// create servo object to control a servo

int potpin1 = A0;
int potpin2 = A1;
int potpin3 = A2;
int potpin4 = A3; // analog pin used to connect the potentiometer
int val1;
int val2;
int val3;
int val4;// variable to read the value from the analog pin



void setup() {
  Serial.begin(9600);
  servo1.attach(10);
  servo2.attach(5);
  servo3.attach(3);
  servo4.attach(11);// attaches the servo on pin 9 to the servo object
pinMode(btnpin,INPUT);
}

void loop() {
  val1 = analogRead(potpin1);
  val2 = analogRead(potpin2);
  val3 = analogRead(potpin3);
  val4 = analogRead(potpin4); // reads the value of the potentiometer (value
between 0 and 1023)

  val1 = map(val1, 1023, 0, 0, 180);
  val2 = map(val2, 0, 1023, 0, 180);
  val3 = map(val3, 1023, 0, 0, 180);
  val4 = map(val4, 0, 1023, 0, 180);  // scale it to use it with the servo (value
between 0 and 180)

  // sets the servo position according to the scaled value

   Run();

}
void Run(){

   servo1.write(val4);
   servo2.write(val3);
   servo3.write(val2);
   servo4.write(val1);
  }
```

3.2 Source Code (Flex Control)

```
#include<Wire.h>                    //I2C Wire Library

#include<Servo.h>                   //Servo Motor Library


Servo servo_1;

Servo servo_2;
Servo servo_3;

Servo servo_4;

const int MPU_addr=0x68;        //MPU6050 I2C Address

int16_t axis_X,axis_Y,axis_Z;

int minVal=265;

int maxVal=402;

double x;

double y;

double z;


void setup()

{

  Serial.begin(9600);

  pinMode(A0,INPUT_PULLUP);

  Wire.begin();                      //Initilize I2C Communication

  Wire.beginTransmission(MPU_addr);  //Start communication with MPU6050

  Wire.write(0x6B);                  //Writes to Register 6B

  Wire.write(0);                     //Writes 0 into 6B Register to Reset

  Wire.endTransmission(true);        //Ends I2C transmission


  servo_1.attach(2);   // Forward/Reverse_Motor
  servo_2.attach(3);   // Up/Down_Motor

  servo_3.attach(4);   // Gripper_Motor

  servo_4.attach(5);   // Left/Right_Motor



}

void loop()

{

  Wire.beginTransmission(MPU_addr);

  Wire.write(0x3B);                  //Start with regsiter 0x3B

  Wire.endTransmission(false);

  Wire.requestFrom(MPU_addr,14,true);  //Read 14 Registers



  axis_X=Wire.read()<<8|Wire.read();                //Reads the MPU6050 X,Y,Z
AXIS Value

  axis_Y=Wire.read()<<8|Wire.read();

  axis_Z=Wire.read()<<8|Wire.read();



  int xAng = map(axis_X,minVal,maxVal,-90,90);     // Maps axis values in terms
of -90 to +90

  int yAng = map(axis_Y,minVal,maxVal,-90,90);

  int zAng = map(axis_Z,minVal,maxVal,-90,90);



  x= RAD_TO_DEG * (atan2(-yAng, -zAng)+PI);       //Formula to convert into
degree
```

```
  y= RAD_TO_DEG * (atan2(-xAng, -zAng)+PI);

  z= RAD_TO_DEG * (atan2(-yAng, -xAng)+PI);


  int gripper;

  int flex_sensorip = analogRead(A0);            //Reads flex sensor output



  if(flex_sensorip < 1023)

        {

          gripper = 0;


        }

        else

        {

          gripper = 180;

        }



   servo_4.write(gripper);                    //Writes gripper value to 3rd servo
motor



  if(x >=0 && x <= 60)

  {

     int mov1 = map(x,0,60,0,90);

     Serial.print("Movement in F/R = ");

     Serial.print(mov1);
```

```
    Serial.println((char)176);

    servo_1.write(mov1);
    delay(100);

 }



 else if(x >=300 && x <= 360)

 {

    int mov2 = map(x,360,250,0,180);

    Serial.print("Movement in Up/Down = ");

    Serial.print(mov2);

    Serial.println((char)176);

    servo_2.write(mov2);
    delay(100);
 }


if(y >=0 && y <= 60)

 {

    int mov3 = map(y,0,60,90,180);

    Serial.print("Movement in Left = ");

    Serial.print(mov3);

    Serial.println((char)176);

    servo_3.write(mov3);
    delay(100);
 }
```

```
  else if(y >=300 && y <= 360)

  {

     int mov3 = map(y,360,300,90,0);

     Serial.print("Movement in Right = ");

     Serial.print(mov3);

     Serial.println((char)176);

     servo_3.write(mov3);
     delay(100);
  }


}
```

3.3 Circuit Diagram

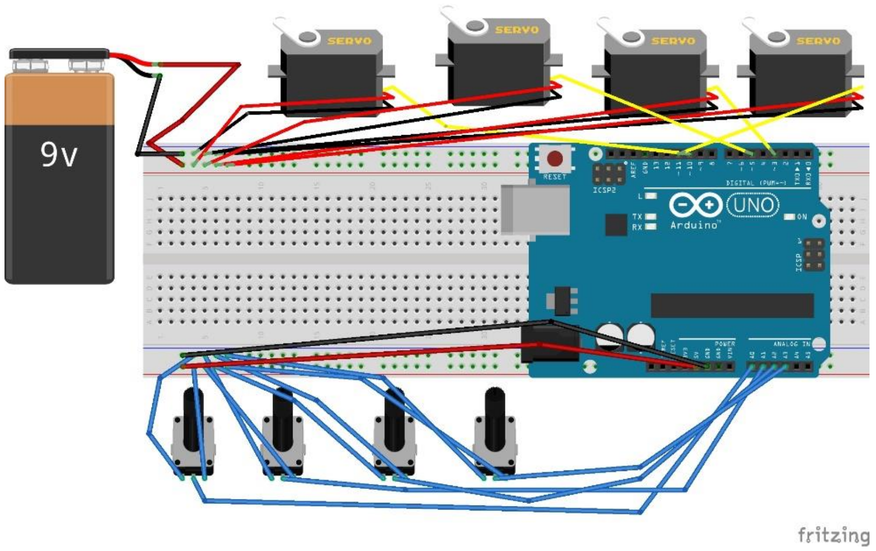


Fig.3.3. (a) Circuit Diagram

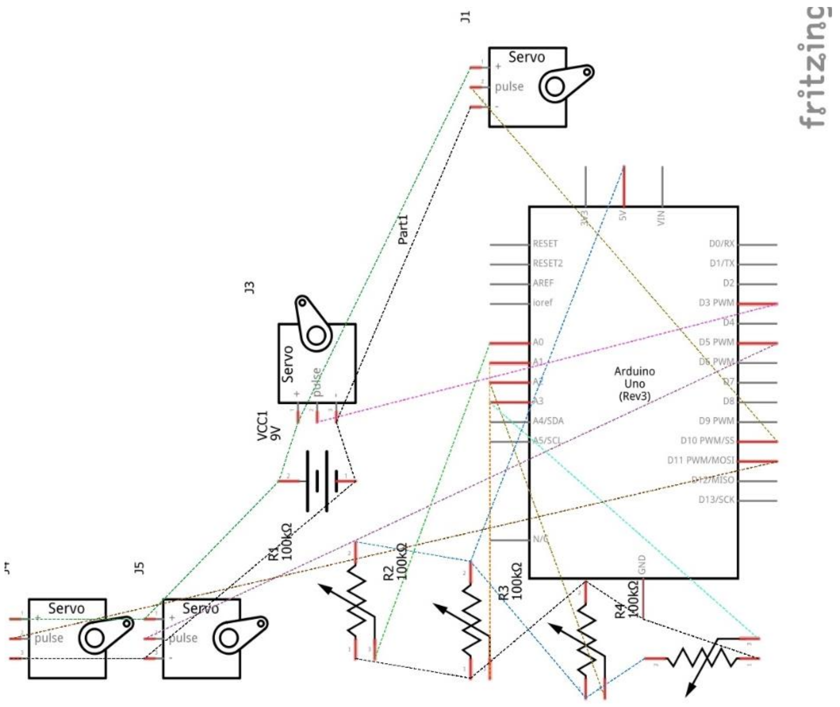


Fig.3.3. (b) Schematic

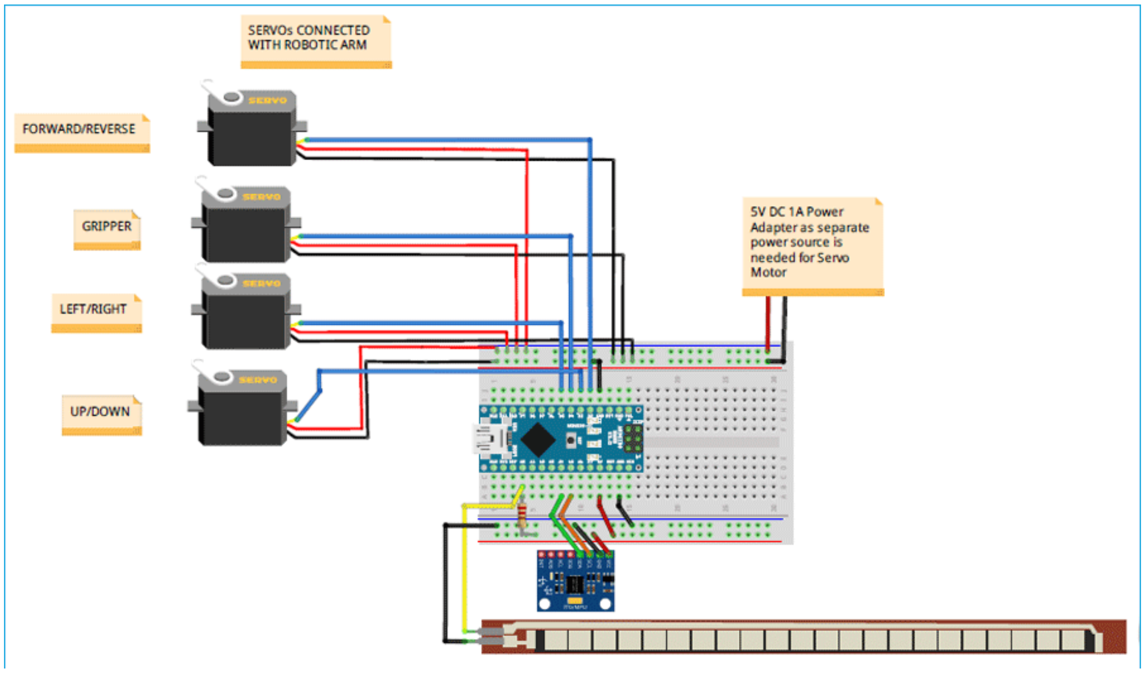
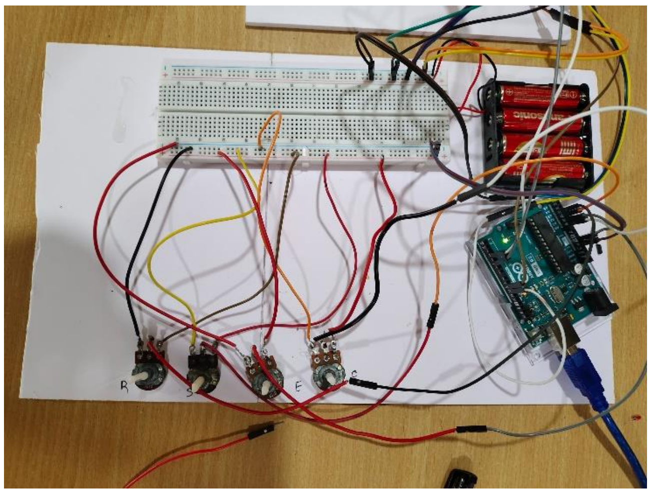

SERVOs CONNECTED WITH ROBOTIC ARM
FORWARD/REVERSE
GRIPPER
LEFT/RIGHT
UP/DOWN
5V DC 1A Power Adapter as separate power source is needed for Servo Motor

Fig.3.3. (c) Flex Control Circuit Diagram

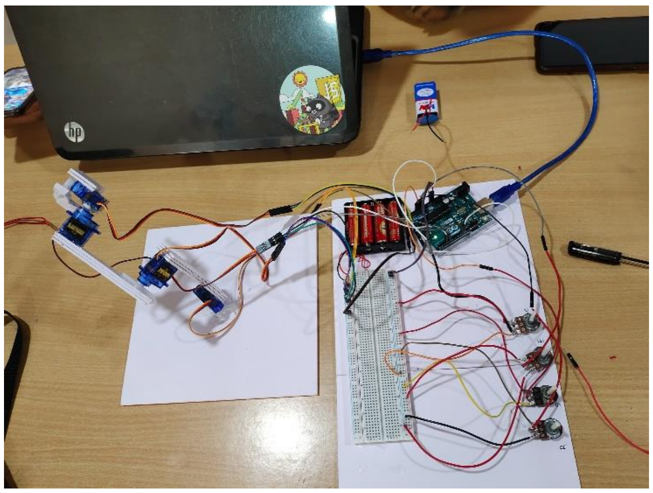
4. IMPLEMENTATION

4.1 Implementation

The Potentiometers are connected to the Arduino Uno Rev3 which is then connected to the computer via serial communication. Now the data received by the computer is processed to remove as much noise as possible.

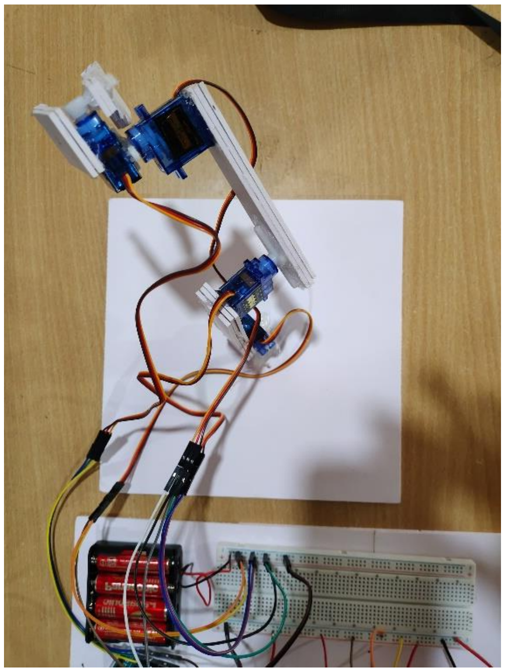


(a)



(b)

Fig.12. (a) Physical Implementation of the system; (b) Overview of the Arm


(c)

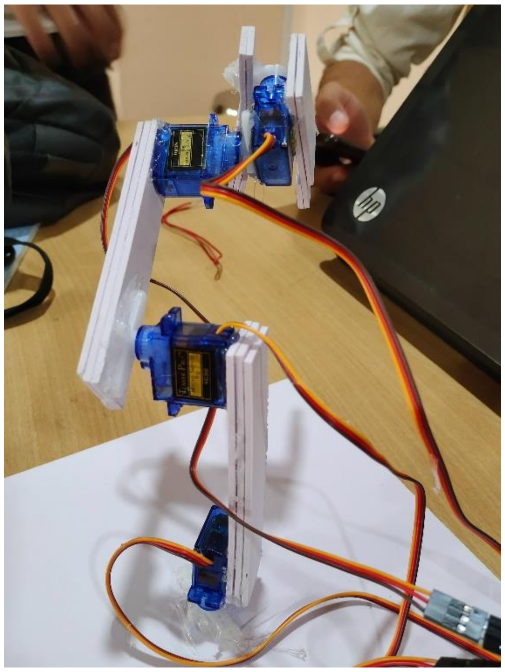
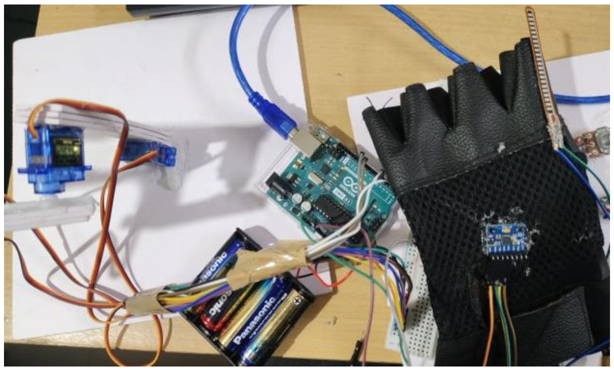

(d)

Fig.12. (c) Closeup of the Physical Implementation of the system; (d) Robotic Arm Only


(e)

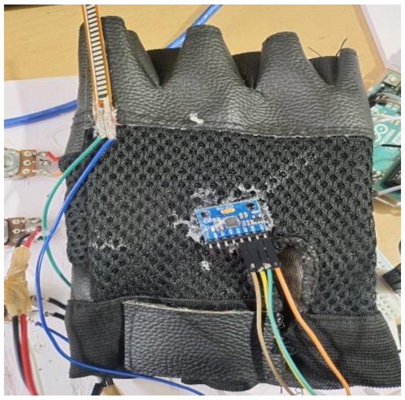

(f)

Fig.12. (d) Closeup of the Physical Implementation of the system; (e) Glove with sensors Only

Each motor moves the arm in one plane. As we have implemented two motors at the shoulder joint M1 is to move the arm in Y-Z plane and M2 is for the movement along the X-Z plane. In this way the two motors provide the shoulder joint to be moved in any direction in space, it can be seen that we have implemented three motors at this joint. The Motor M3 is for the movement of the arm along the Z-axis in the X-Y plane. The Motor M4 is used for the claw.

5. CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

The objectives of this project have been achieved which was developing the hardware and software for a potentiometer meter controlled robotic arm. From observation that has been made, it clearly shows that its movement is precise, accurate, and is easy to control and user friendly to use. The robotic arm has been developed successfully as the movement of the robot can be controlled precisely. This robotic arm control method is expected to overcome the problem such as placing or picking object that away from the user, pick and place hazardous object in a very fast and easy manner.

5.2 Future Scope

The project is built on a wired model. It could further be developed to work on wireless communication, thus allowing the user to move in an even easier unrestricted manner.

BIBLIOGRAPHY

[1] https://www.instructables.com/Simple-Mimicking-Robot-Arm-Using-Arduino/

[2] https://www.wikipedia.org/

[3] https://docs.arduino.cc/hardware/uno-rev3

[4] https://www.google.com/

[5] https://fritzing.org/

[6] http://ethesis.nitrkl.ac.in/5283/1/109EI0297.pdf

[7] https://circuitdigest.com/microcontroller-projects/diy-hand-gesture-controlled-robotic-arm-using-arduino-nano#:~:text=In%20this%20Gesture%20controlled%20Robotic,rotates%20and%20the%20gripper%20opens.