Project Report

on

5 DOF Robotic Arm

Submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

Session 2020-2021

Electronics & Communication Engineering

By :

TANURAJ TOMAR
1721631088

NEERAJ KUMAR
1721631055

RAJA HASSAN
1721631065

Under the guidance of

DR. ABHISHEK SHARMA

IIMT COLLEGE OF ENGINEERING, GREATER NOIDA

AFFILIATED TO

DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

Student’s Declaration

I / We hereby declare that the work being presented in this report entitled **“5 DOF ROBOTIC ARM”** is an authentic record of my / our own work carried out under the supervision of **“DR. ABHISHEK SHARMA”.**

The matter embodied in this report has not been submitted by me / us for the award of any other degree.

Dated:

Signature of students(s)
Tanuraj Tomar
Neeraj Kumar
Raja Hassan

This is to certify that the above statement made by the candidate(s) is correct to the best of my knowledge.

Signature of HOD
Prof. (Dr.) Seema Nayak

Signature of Supervisor
Dr. Abhishek Sharma

Date.......................

Dept. of ECE

IIMT COLLEGE OF ENGINEERING GREATER NOIDA

ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the project titled “5 DOF ROBOTIC ARM” is the Bonafide work carried out by , **Neeraj Kumar**, **Tanuraj Tomar** and **Raja Hassan** the students of B.Tech (ECE) of IIMT College of Engineering, Greater Noida, UP (India) during the academic year 2020-21, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Electronics & Communication Engineering) and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Dr. Abhishek Sharma	**Mr. Basanta Mahato**	**Prof. (Dr.) Seema Nayak**
(Assistant Professor)	**(Assistant Professor)**	**(Professor)**
Project Guide	**Project Coordinator**	**H.O.D (ECE)**

ACKNOWLEDGEMENT

Our project title is **“5 DOF ROBOTIC ARM”**. we completed this project in time of requirement of the degree of Bachelor Engineering **(Electronics & Communication engineering)**. we would like to thank to all people had assisted us either directly or indirectly in completing our final year project. Our special thanks to **Dr.Abhishek Sharma**, our intelligent supervisor for the project whom had given support, knowledge, advice and guidance that we need. He had been guiding us from the beginning of the project until the final thesis had been done. With his support, we have learned a lot of knowledge regarding this project, many things that we can use in our future especially the experience making the project. Without him, we not expect that, we will finish this project same as university requirement. And also thank to **Mr. Basanta Mahato** , our project In charge. A million thank also to other lecturers that had teach and helped in completing this project.

Signature of students

NEERAJ KUMAR
1721631055

TANURAJ TOMAR
1721631088

RAJA HASSAN
1721631065

Table of Contents

ABSTRACT

5 DOF robotic arm

Nowadays, under the progress of science and technology, the biggest difference between a robotic arm and a human arm lies in flexibility and strength. That is, the biggest advantage of the robotic arm is that normally it can repeat the same motion without feeling tired.This paper presents the design and development of a 5-Degree of Freedom (DOF) robotic arm, where the position of the joints are controlled by the user. A gui interface is designed for controlling the robotic arm through sending the ascii signals over the wired serial communication and additionally an android application is also developed to control the robotic arm over a wireless link using a bluetooth module. The Arduino UNO R3 board is the main heart of this project which interfaces with the Graphical User Interface and servo motors.

CHAPTER 1. 5 DOF ROBOTIC ARM

1.1 WHAT IS ROBOTIC ARM?

A robotic arm is any of a variety of mechanical, programmable devices that are meant to move things in a manner comparable to that of a human arm. The robotic arm was one of the most beneficial pieces of technology developed in the twentieth century, soon becoming a cornerstone in many industries. It can do a variety of tasks and activities that would be too time-consuming, difficult, or dangerous for a human to perform. Robotic arms are often associated with the automotive sector, but they may also be used for a variety of other activities than welding and painting vehicle components.

A scientist called George Devol, Jr. created the first robotic arm in the 1950s, before which robotics were mostly the stuff of science fiction and the imagination. For a long time, robotics progressed slowly, with many of the most useful applications involving space exploration. Robotic arms were not completely realised as an aid to industrialization until the 1980s, when they were incorporated into automotive and other industry assembly lines.

Robot arms, while operating in a similar manner to human arms, can nevertheless have a considerably broader range of motion because their design is entirely up to the creator's creativity. For example, the joint connecting the segments of a robotic arm can rotate as well as move like a hinge. The end effector is the part of the robotic arm that does the function it was meant to accomplish. It may be built for almost any duty, such as grasping like a hand, painting, tightening screws, and so on. These robots can be fixed in one location, such as along an assembly line, or they can be movable, allowing them to perform a variety of jobs in several locations.

Autonomous robotic arms are intended to be programmed and then left alone to do tasks without human intervention. A robotic arm, on the other hand, can be built to be operated and controlled by a human. In space exploration, where robotic arms can be used to move a large cargo or do other tasks in orbit that would be difficult or impossible for an astronaut to undertake, human-controlled robotic arms are critical.

1.2 DEGREE OF FREEDOM

The degrees of freedom, or DOF, is a crucial concept to grasp. Each degree of freedom corresponds to a joint on the arm that may bend, rotate, or translate.
The number of actuators on the robotic arm is usually used to determine the number of degrees of freedom. Each degree necessitates the use of a motor, frequently an encoder, as well as increasingly complex algorithms and costs. A 5-DOF robotic arm, for example, has five joints and requires five motors to operate.

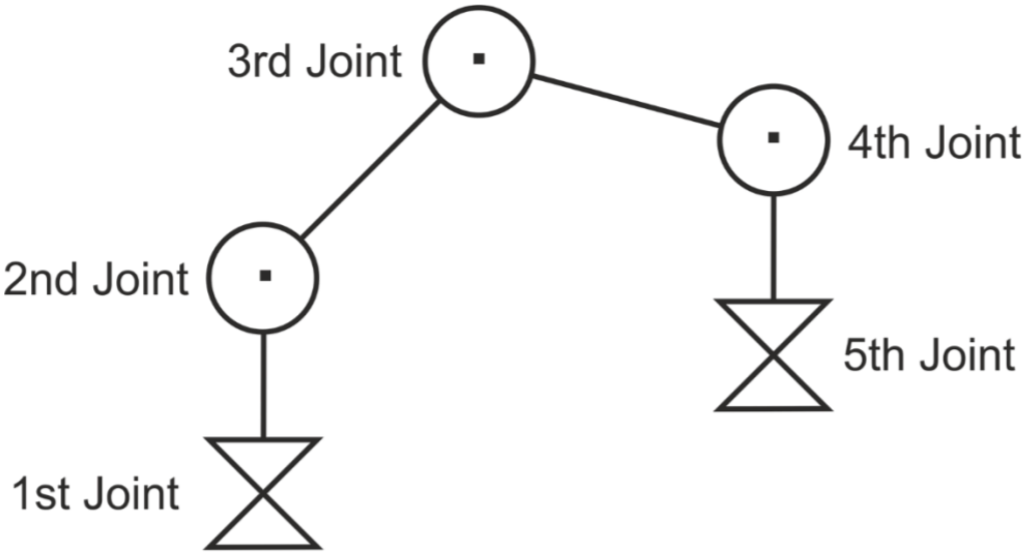


Figure 1.1 5-DOF Robotic Arm Representation

1.3 ARM DESIGN & MECHANICAL CONFIGURATION

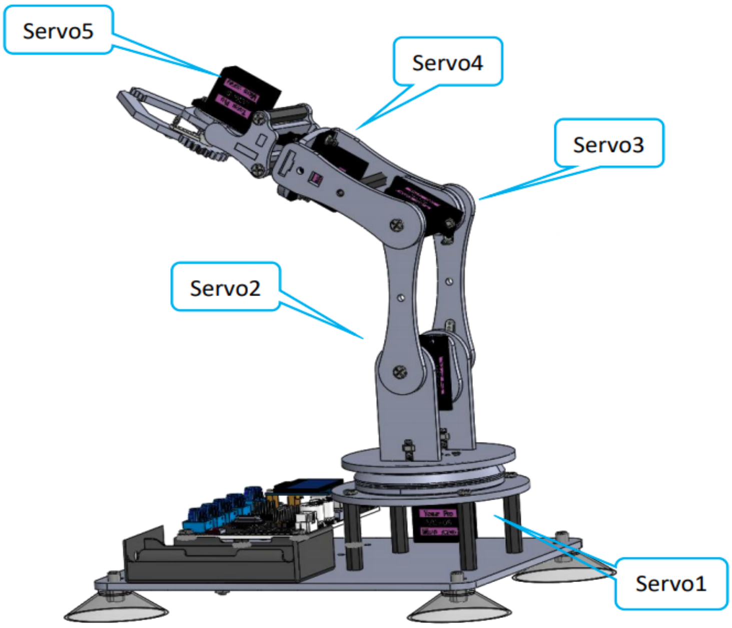


Figure 1.2 5 DOF robotic arm model

Figure 1.2 depicts the actual model of the created robotic arm, which was constructed utilizing acrylic plates due to its lightweight design. An Arduino board is used to control the actuators.In Figure 1.3, you can see the Arduino board that to be used in this project.

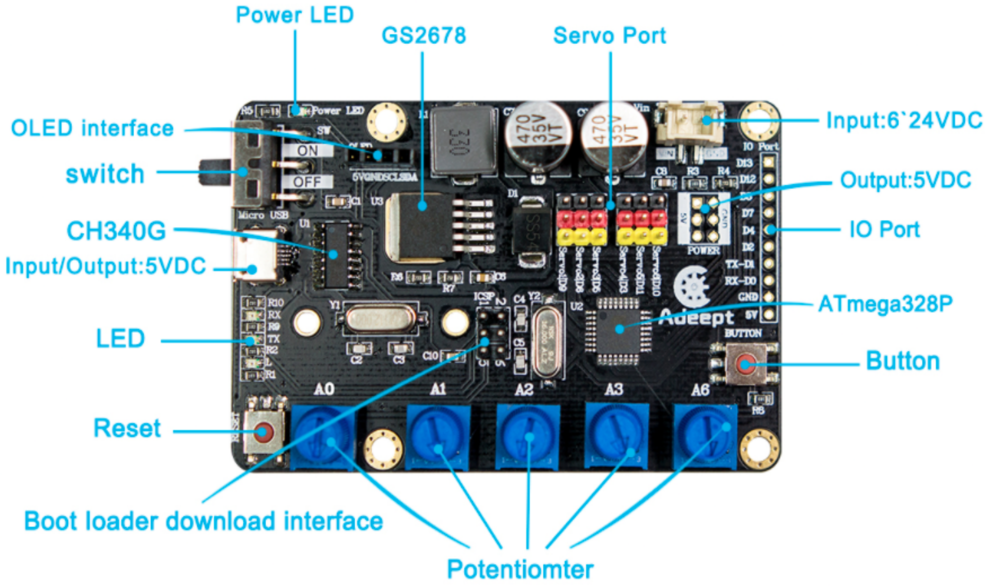


figure 1.3 Arm drive board

The rotary joints are formed by servo motors 1 and 4, while the revolute joints are formed by the other servo motors.

Base, shoulder, elbow, wrist, and gripper are the names we've given to the joints of the arm. All of the servo motors are wired to the Arduino's pwm pins 3, 5, 6, 9, and 11. These servo motors are the actuators that allow the arm to move physically. The connections between actuators (servo motors) and Arduino pwm pins are shown in the table below.

Actuators	**Joints**	**Arduino pwm pins**
servo 1	base	9
servo 2	shoulder	6
servo 3	elbow	5
servo 4	wrist	3
servo 5	gripper	11

1.4 REQUIRED COMPONENTS

Components	*quantity*
Servo motors	5
Servo Extension Cable	5
Winding Pipe	1
18650x2 Battery Holder	1
18650x2 Battery	2
Micro USB Cable	1
Arduino UNO R3	1
Potentiometer	5
OLED display	1
Bluetooth module HC-05	1

1.5 HC 05 BLUETOOTH MODULE

A bluetooth module is installed on the Arduino board which can receive data through bluetooth from android mobile so we can control the arm with mobile.

Bluetooth is a wireless communication technology. It's meant to take the place of cable connections. It communicates with devices using serial communication. It uses a serial port to connect with the microcontroller (USART). It usually uses a short-range wireless connection to connect tiny devices such as phones, PDAs, and televisions to exchange documents. It operates at a frequency of 2.45GHz. A point-to-point or multi-point connection with a maximum range of 10 meters is possible. The data is transferred at a rate of 1Mbps.

HC-05 The Bluetooth module allows you to switch between master and slave mode, which means you may use it for both receiving and delivering data.

When compared to the HC-06 module, which can only be set as a slave, the HC-05 can also be set as a master, allowing communication between two Arduino boards.

You can use the Bluetooth module as a serial port replacement to connect your MCU, PC, and embedded project, for example.

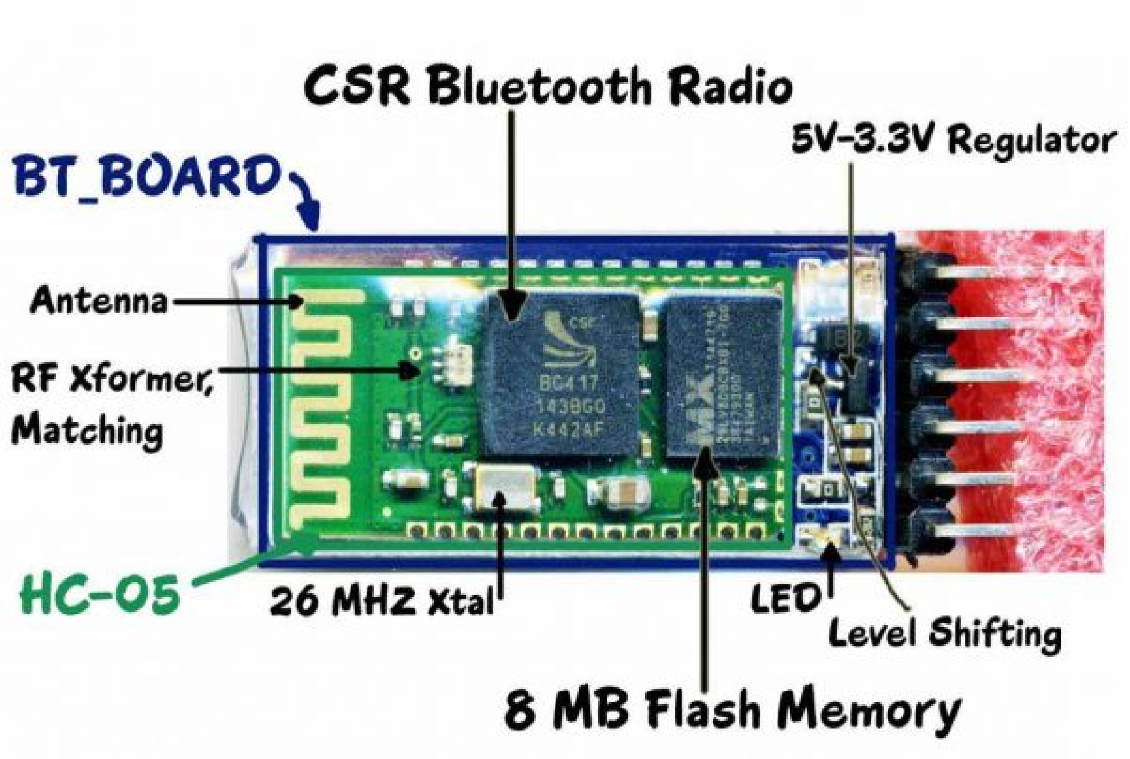


figure 1.4 HC 05 bluetooth module

HC-05 Specification

1. Bluetooth protocol: Bluetooth Specification v2.0+EDR
2. Frequency: 2.4GHz ISM band
3. Modulation: GFSK(Gaussian Frequency Shift Keying)
4. Emission power: ≤4dBm, Class 2
5. Sensitivity: ≤-84dBm at 0.1% BER
6. Speed: Asynchronous: 2.1Mbps(Max) / 160 kbps, Synchronous: 1Mbps/1Mbps
7. Security: Authentication and encryption
8. Profiles: Bluetooth serial port
9. Power supply: +3.3VDC 50mA
10. Working temperature: -20 ~ +75Centigrade
11. Dimension: 26.9mm x 13mm x 2.2 mm
12. It is IEEE 802.15.1 standardized protocol, through which one can build wireless Personal Area Network (PAN). It uses frequency-hopping spread spectrum (FHSS) radio technology to send data over air.

Pin Description

It has six pins:

1.Key/EN: This pin is used to turn on the Bluetooth module and enable AT instructions. This pin is set to data mode by default. To use Bluetooth in command mode, the key/EN pin must be high. In command mode, the HC-05's default baud rate is 38400bps, while in data mode, it's 9600bps. There are two modes on the HC-05 module.

Data mode refers to the exchange of information between devices. In data mode, the baud rate is 9600bps.
AT commands are used to adjust the settings of the HC-05 in command mode. In command mode, the baud rate is 38400bps.

2. VCC: Connect 5 V or 3.3 V to this Pin.

3. GND: Ground Pin of module.

4. TXD: Connect with Microcontroller RXD pin of Microcontroller. Transmit Serial data (wirelessly received data by Bluetooth module transmitted out serially on TXD pin).

5. RXD: Connect with Microcontroller TXD pin of Microcontroller. Received data will be transmitted wirelessly by Bluetooth module.

CHAPTER 2. READING DATA FROM POTENTIOMETER

2.1 INTRODUCTION OF POTENTIOMETER

The potentiometer is a resistance element with three terminals and the resistance value can be adjusted according to a certain change law, which is equivalent to a variable resistor. Because its role in the circuit is to obtain a certain relationship with the input voltage (external voltage) to output voltage, so called potentiometer. Potentiometers can be divided into rotary potentiometers, push-pull potentiometers, straight slide potentiometers, etc. according to the adjustment method. Our course experiment uses a rotary potentiometer.Its three pins are shown below:

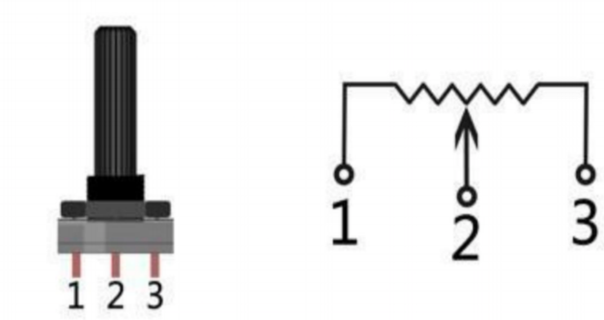


figure 2.1 potentiometer

The rotary potentiometer is an adjustable resistance element. It is composed of a resistor and a rotating system. When a voltage is applied between the two fixed contacts of the resistive body, the position of the contact on the resistive body is changed by the rotating system, and a voltage that has a certain relationship with the position of the moving contact can be achieved between the moving contact and the fixed contact. A potentiometer can be used to adjust the voltage and current. Our course uses a rotary potentiometer. Its structure is as shown in the figure below. By rotating the knob, the position of pin 2 is changed, thereby changing the resistance value from pin 2 to both ends. In the experiment. Connect pin 1 and pin 3 to the GND and 5V of the development board respectively. And then read the voltage divided by the pin 2 of the potentiometer through the analog input pin A0. The range is between 0V and 5V. The analog input function of Arduino has 10-bit precision, that is, it can convert the voltage signal of 0 to 5V into an integer form of 0 to 1024.

2.2 WIRING DIAGRAM (CIRCUIT DIAGRAM)

figure 2.2 shows the potentiometers connected to analog pins of the arm drive board.

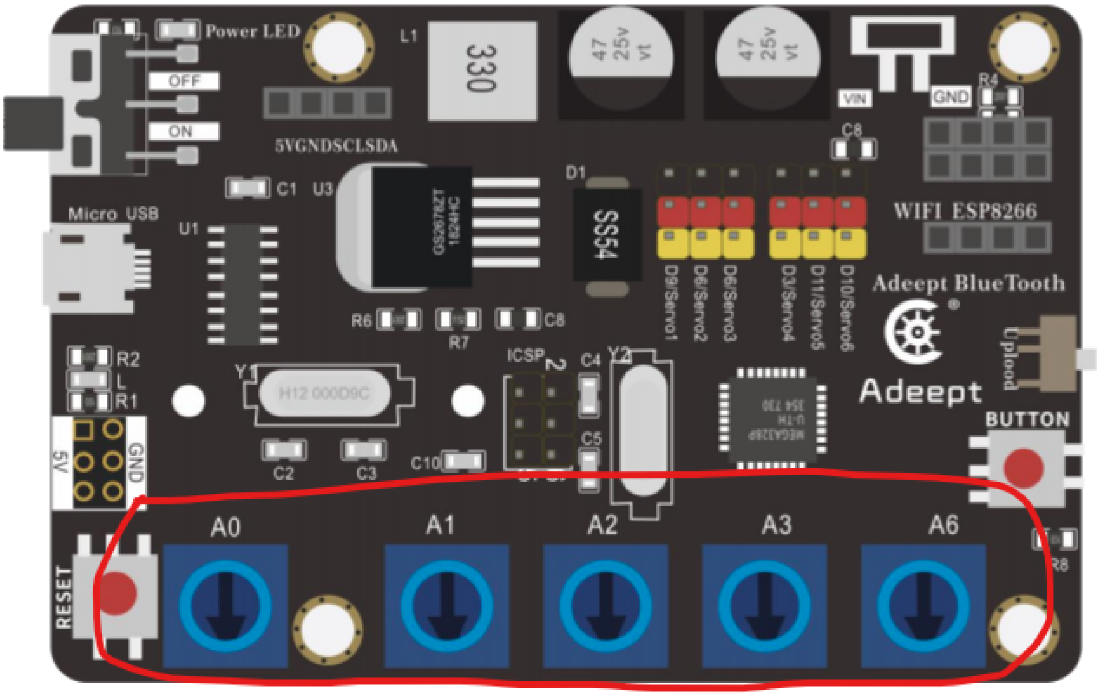


figure 2.2 potentiometers on arm drive board.

2.3 READING THE VALUE OF THE POTENTIOMETER AND CONVERTING IT INTO AN ANGLE

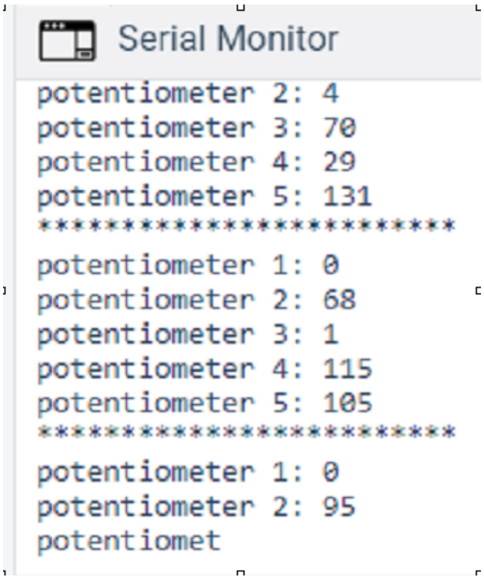
The analogRead() and map() functions in the Arduino library allow you to read data from analog pins and mapping of data respectively.

The following Arduino program reads data from potentiometers and maps it to a range of 0 to 180.

```
#define P1_PIN 0      // potentiometers connected to pin 0, 1, 2, 3 and 6
#define P2_PIN 1
```

```
#define P3_PIN 2
#define P4_PIN 3
#define P5_PIN 6

int pot_1, pot_2, pot_3, pot_4, pot_5;

void setup() {
  Serial.begin(9600);
}

void loop() {
  // read data from potentiometers
  pot_1 = map(analogRead(P1_PIN), 0, 1023, 0, 180);
  pot_2 = map(analogRead(P2_PIN), 0, 1023, 0, 180);
  pot_3 = map(analogRead(P3_PIN), 0, 1023, 0, 180);
  pot_4 = map(analogRead(P4_PIN), 0, 1023, 0, 180);
  pot_5 = map(analogRead(P5_PIN), 0, 1023, 0, 180);


  // print out read values
  delay(15);

  Serial.print("potentiometer 1: "); Serial.println(pot_1);
  Serial.print("potentiometer 2: "); Serial.println(pot_2);
  Serial.print("potentiometer 3: "); Serial.println(pot_3);
  Serial.print("potentiometer 4: "); Serial.println(pot_4);
  Serial.print("potentiometer 5: "); Serial.println(pot_5);
```

OUTPUT



```
potentiometer 2: 4
potentiometer 3: 70
potentiometer 4: 29
potentiometer 5: 131
************************
potentiometer 1: 0
potentiometer 2: 68
potentiometer 3: 1
potentiometer 4: 115
potentiometer 5: 105
************************
potentiometer 1: 0
potentiometer 2: 95
potentiomet
```

CHAPTER 3. CONTROLLING THE SERVO MOTORS

3.1 SERVO MOTOR

A servomotor is a rotary or linear actuator that can control angular or linear position, velocity, and acceleration with precision. It is made of a suitable motor and a position feedback sensor. Servo Motors are used in robotics, CNC machines, and automated manufacturing, among other uses. Servo motor is shown in figure 3.1.

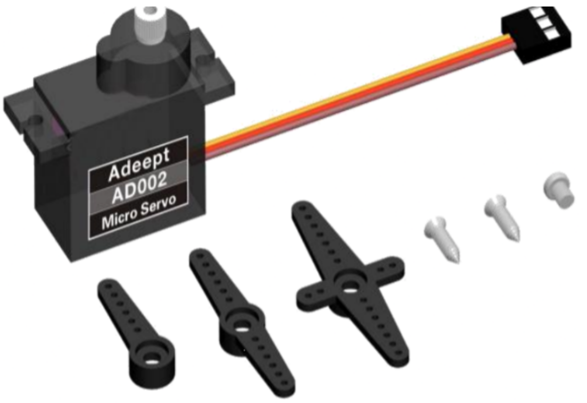


figure 3.1 servo motor

The servo mechanism is a self-contained control system that allows the object's position, orientation, state, and other output controlled qualities to follow arbitrary commands.the input target has changed (or given value).

Pulse is mostly responsible for the servo's placement. To summarize, the servo motor receives an impulse and rotates the angle corresponding to the impulse in order to achieve displacement. Because the servo motor's duty is to send out pulses, it rotates at an angle every time, causing a matching number of pulses to be sent out. The pulses received by the servo motor form a response, or a closed loop, in this fashion. The system will be able to tell how many pulses are supplied to the servo motor and how many pulses are received this way. It is feasible to carefully control the motor's rotation in this manner, resulting in exact positioning.

A PWM signal is sent from a microcontroller to a servo motor, which is then processed by an IC on the circuit board to compute the rotation direction of the drive motor, which is then communicated to the swing arm via a reduction gear. Simultaneously, the position detector sends out a position signal to determine whether the set position has been achieved.

3.2 THE WORKING PRINCIPLE OF SERVO MOTOR

The servo mechanism is an automatic control system that enables the object's position, orientation, state and other output controlled quantities to follow arbitrary changes in the input target (or given value). The servo mainly depends on Pulse for location. It can be understood that the servo motor receives an impulse and rotates the angle corresponding to the impulse to realize displacement. Because the servo motor itself has the function of sending out pulses, the servo motor rotates every time at an angle, and a corresponding number of pulses will be sent out. In this way, the pulses received by the servo motor form a response, or a closed loop. In this way, the system will know how many pulses are sent to the servo motor and how many pulses are received. In this way, it is possible to control the rotation of the motor precisely, thereby achieving precise positioning.

Arm Drive Board sends a PWM signal to a servomotor, which is then processed by an IC on the circuit board to calculate the rotation direction of the drive motor, which is then transmitted through a reduction gear to the swing arm. At the same time, the position detector returns a position signal to determine whether the set position has been reached or not.

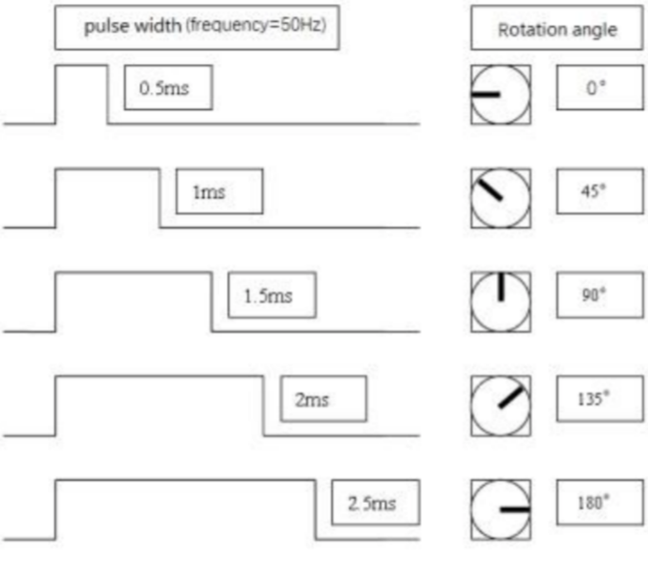


figure 3.2 Servo rotation corresponding to pulse

3.3 CONTROLLING THE SERVO MOTOR WITH ARDUINO

Arduino reads the from potentiometers and uses it to set the angle of servo motors. The Arduino library provides a header file to control the servo motor called Servo.h.

peace of code to control the servo motor

```
//...
#include<Servo.h>
Servo servo;                     // create an object of Servo

void setup() {
   //..
   servo.attach(9);              // attach the servo to pin 9
}

void loop() {
// read data from potentiometer connected to pin a0
int value = readAnalog(0);
int angle = map(0, 1023, 0, 180);

// update the rotation of servo motor
servo.write(angle);
}
```

CHAPTER 4. DISPLAYING TEXT ON THE OLED SCREEN

4.1 OLED SCREEN

OLED (Organic Light-Emitting Diode), also known as organic electric laser display, organic light emitting semiconductor (Organic Electroluminesence Display, OLED). OLED is a kind of current-type organic light-emitting device, which produces light by the injection and recombination of carriers, and the luminous intensity is proportional to the injected current. The Alter robot uses an OLED screen to display the expressions or some parameters of the robot. OLED Screen is a commonly used module on robot products. Due to the black non-luminous feature of OLED Screen, this type of screen has extremely high contrast. Even if the ambient light is strong, you can see the information on the OLED Screen clearly, and the power consumption is relatively low

When using the OLED Screen, you need to connect it to the OLED interface on the Adeept Arm Drive Board.

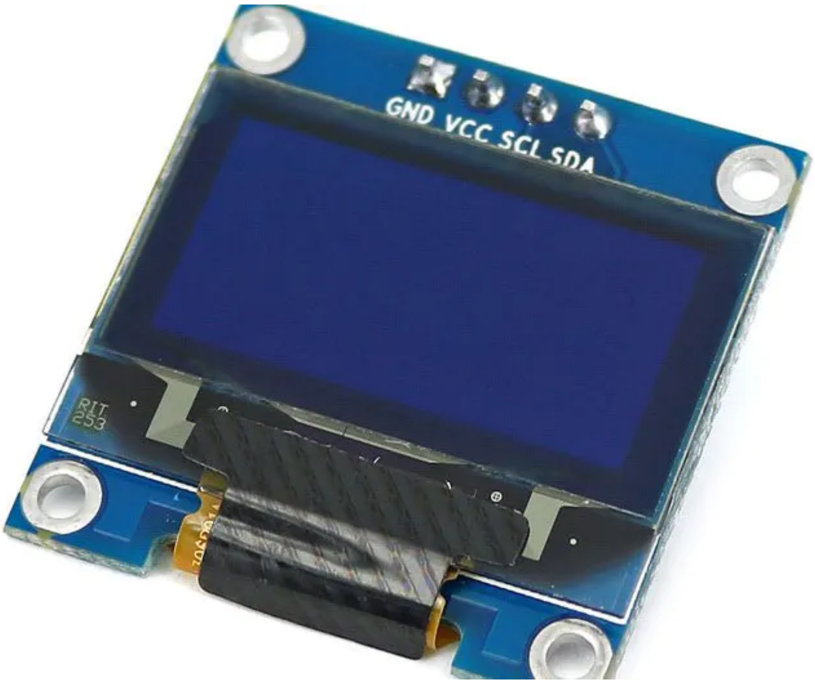


figure 4.1 OLED DISPLAY

4.2 CODE TO DISPLAY TEXT ON OLED SCREEN

```
#include<Adafruit_SSD1306.h>
#include<Adafruit_GFX.h>
#include<Wire.h>

#define OLED_RESET 4
#define SCREEN_ADDRESS 0x3c
#define OLED_WIDTH 128
#define OLED_HEIGHT 64

Adafruit_SSD1306 oled;

void setup() {
   // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
   if (!oled.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
     Serial.println(F("SSD1306 allocation failed"));
     for (;;); // Don't proceed, loop forever
   }

   oled.display();
   delay(2000);            // display the logo for 2 seconds
}

void loop() {
   clear();    // clear the display
   drawString(String("Hello world!"), 1, 1, 2); // // show hello world
}

 void drawString(String str, int x, int y, int s) {
      oled.setTextColor(WHITE);
      oled.setTextSize(s);
      oled.setCursor(x, y);
      oled.print(str);
      oled.display();
    }

 void clear() {
      oled.clearDisplay();
    }
```

CHAPTER 5. FINALIZING THE COMPLETE CODE FOR ARDUINO

5.1 Class RobotArm

An object of RobotArm class represents a robotic arm that includes all of the servo motors' angles and methods to update the angle of the servo motor.

```
class RobotArm {
  private:
    Servo base;
    Servo shoulder;
    Servo elbow;
    Servo wrist;
    Servo grip;

    int baseAngle = 90;
    int shoulderAngle = 90;
    int elbowAngle = 90;
    int wristAngle = 90;
    int gripAngle = 90;

  public:
    void setServoPins(int p1, int p2, int p3, int p4, int p5)
    {
      base.attach(p1);
      shoulder.attach(p2);
      elbow.attach(p3);
      wrist.attach(p4);
      grip.attach(p5);
    }

    void init()
    {
      base.write(90);
      shoulder.write(90);
      elbow.write(90);
      wrist.write(90);
      grip.write(90);
    }

    void setBaseRot(int a) {
      base.write(a);
    }

    void setShoulderRot(int a) {
      shoulder.write(a);
    }

    void setElbowRot(int a) {
      elbow.write(a);
    }

    void setWristRot(int a)
    {
      wrist.write(a);
    }

    void setGripRot(int a)
    {
      grip.write(a);
    }

    void rotateLeft(int dr)
    {
      baseAngle += dr;
      if (baseAngle > 180) baseAngle = 180;
      base.write(baseAngle);
    }

    void rotateRight(int dr)
    {
      baseAngle -= dr;
      if (baseAngle < 0) baseAngle = 0;
      base.write(baseAngle);
    }

    void openShoulder(int dr)
    {
      shoulderAngle -= dr;
      if (shoulderAngle < 0) shoulderAngle = 0;
      shoulder.write(shoulderAngle);
    }

    void closeShoulder(int dr)
    {
      shoulderAngle += dr;
      if (shoulderAngle > 180) shoulderAngle = 180;
      shoulder.write(shoulderAngle);
    }


    // control the elbow at the elbow
    void openElbow(int dr)
    {
      elbowAngle += dr;
      if (elbowAngle > 180) elbowAngle = 180;
      elbow.write(elbowAngle);
    }

    void closeElbow(int dr)
    {
      elbowAngle -= dr;
      if (elbowAngle < 0) elbowAngle = 0;
      elbow.write(elbowAngle);
    }



    void wristLeft(int dr)
    {
      wristAngle += dr;
      if (wristAngle > 180) wristAngle = 180;
      wrist.write(wristAngle);
    }

    void wristRight(int dr)
    {
      wristAngle -= dr;
      if (wristAngle < 0) wristAngle = 0;
      wrist.write(wristAngle);
    }



    void openGrip(int dr)
    {
      gripAngle -= dr;
      if (gripAngle < 0) gripAngle = 0;
      grip.write(gripAngle);
    }

    void closeGrip(int dr)
    {
      gripAngle += dr;
      if (gripAngle > 90) gripAngle = 90;
      grip.write(gripAngle);
    }
};
```

This robot arm has to be controlled in 2 modes
1. Controlling via Potentiometers
2. Controlling via Serial commands (receiving from mobile or desktop)

To switch between controlling modes a push button is connected to digital pin 4.

5.2 class KnobController

The knob of the potentiometers given on the Arduino arm drive board can be used to control the arm.

```
class KnobController {
  private:
    // hold int data in the range from 0 to 180
    int pot_1;
    int pot_2;
    int pot_3;
    int pot_4;
    int pot_5;

    RobotArm m_arm;

  public:
    KnobController(RobotArm robot_arm) :  m_arm(robot_arm) { }

    void poll() {
      pot_1 = map(analogRead(P1_PIN), 0, 1023, 0, 180);
      pot_2 = map(analogRead(P2_PIN), 0, 1023, 0, 180);
      pot_3 = map(analogRead(P3_PIN), 0, 1023, 0, 180);
      pot_4 = map(analogRead(P4_PIN), 0, 1023, 0, 180);
      pot_5 = map(analogRead(P5_PIN), 0, 1023, 0, 180);
    }

    void step() {
      m_arm.setBaseRot(pot_1);
      m_arm.setShoulderRot(pot_2);
      m_arm.setElbowRot(pot_3);
      m_arm.setWristRot(pot_4);
      m_arm.setGripRot(pot_5);
    }
};
```

5.3 class OledDisplay

```
class OledDisplay {
  private:
    Adafruit_SSD1306 oled;

  public:
    OledDisplay(): oled(128, 64, &Wire, OLED_RESET) {}

    void init() {
      // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
      if (!oled.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
        Serial.println(F("SSD1306 allocation failed"));
        for (;;); // Don't proceed, loop forever
      }

      oled.display();
      delay(1500);
    }

    void clear() {
      oled.clearDisplay();
    }

    void drawString(String str, int x, int y, int s) {
      oled.setTextColor(WHITE);
      oled.setTextSize(s);
      oled.setCursor(x, y);
      oled.print(str);
      oled.display();
    }
};
```

5.4 COMPLETE CODE FOR ARDUINO

```
#include<Servo.h>
#include<SoftwareSerial.h>
#include<Adafruit_SSD1306.h>
#include<Adafruit_GFX.h>
#include<Wire.h>

// servo pins
#define BASE_PIN 9
#define SHOULDER_PIN 6
#define ELBOW_PIN 5
#define WRIST_PIN 3
#define GRIPPER_PIN 11

// pins for potentiometers
#define P1_PIN 0
#define P2_PIN 1
#define P3_PIN 2
#define P4_PIN 3
#define P5_PIN 6

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
// The pins for I2C are defined by the Wire-library.
// On an arduino UNO:       A4(SDA), A5(SCL)
```

```
#define OLED_RESET 4
#define SCREEN_ADDRESS 0x3c
#define OLED_WIDTH 128
#define OLED_HEIGHT 64



//-----------------------------------
//    STARTS FROM HERE
//-----------------------------------

RobotArm arm;
KnobController knobController(arm);
OledDisplay scr;
SoftwareSerial btSerial(7, 8); // RX | TX
int mode = 2;                  // by default


// imaginary (fake) controlling buttons can be pressed and released via Serial communication.
// -------------------------------------------------
// buttons      //      when button is down
// -------------------------------------------------
// btn_0        //      rotate left
// btn_1        //      rotate Right
// btn_2        //      open shoulder
// btn_3        //      close shoulder
// btn_4        //      open elbow
// btn_5        //      close elbow
// btn_6        //      rotate wrist clockwise
// btn_7        //      rotate wrist anti-clockwise
// btn_8        //      open grip
// btn_9        //      close grip
// --------------------------------------------------


boolean btn[10] = {false};  // contains all buttons from 0 to 9, boolean true indicates that the button is down, otherwise not.
int buttonPressed;          // state for physical button given on the board


void setup() {
  btSerial.begin(9600);
  Serial.begin(9600);

  pinMode(4, INPUT);
  pinMode(BASE_PIN, OUTPUT); //Set the servo interface as the
```

```
output interface
  pinMode(SHOULDER_PIN, OUTPUT); //Set the servo interface as the output interface
  pinMode(ELBOW_PIN, OUTPUT); //Set the servo interface as the output interface
  pinMode(WRIST_PIN, OUTPUT); //Set the servo interface as the output interface
  pinMode(GRIPPER_PIN, OUTPUT); //Set the servo interface as the output interface

  scr.init();
  arm.setServoPins(BASE_PIN, SHOULDER_PIN, ELBOW_PIN, WRIST_PIN, GRIPPER_PIN);
  arm.init();

  scr.clear();
  scr.drawString(String("mode: ") + mode, 1, 1, 2);

}

void loop() {

  if (digitalRead(4) == LOW) {
// if the on board BUTTON is pressed, PIN4 will connect to ground
    buttonPressed++;
  } else {
    buttonPressed = 0;
  }
  if (buttonPressed == 1) {
// is button down once
    mode = (mode > 1) ? 1 : ++mode;
    scr.clear();
    scr.drawString(String("mode: ") + mode, 1, 1, 2);
  }

  switch (mode) {
    case 1:
      // scrtrolling mode 1 : with potentiometers
      // read potentiometers at analog pins a0, a1, a2, a3, a6
      knobscrtroller.poll();
      knobscrtroller.step();
      break;

    case 2:
      // process commands
      if (Serial.available() > 0) {
        int command = Serial.read();
        if (command == 'a') btn[0] = !btn[0]; else if (command == 'd') btn[1] = !btn[1];
        if (command == 'k') btn[2] = !btn[2]; else if (command == 'i') btn[3] = !btn[3];
        if (command == 's') btn[4] = !btn[4]; else if (command == 'w') btn[5] = !btn[5];
        if (command == 'j') btn[6] = !btn[6]; else if (command == 'l') btn[7] = !btn[7];
        if (command == 'q') btn[8] = !btn[8]; else if (command == 'e') btn[9] = !btn[9];
        if (command == '>') Serial.println('<');
      }

      while (btSerial.available() > 0) {
        int command = btSerial.read();
        if (command == 'a') btn[0] = !btn[0]; else if (command == 'd') btn[1] = !btn[1];
        if (command == 'k') btn[2] = !btn[2]; else if (command == 'i') btn[3] = !btn[3];
        if (command == 's') btn[4] = !btn[4]; else if (command == 'w') btn[5] = !btn[5];
        if (command == 'j') btn[6] = !btn[6]; else if (command == 'l') btn[7] = !btn[7];
        if (command == 'q') btn[8] = !btn[8]; else if (command == 'e') btn[9] = !btn[9];
      }

      if (btn[0]) arm.rotateLeft(1);
      if (btn[1]) arm.rotateRight(1);
      if (btn[2]) arm.openShoulder(1);
      if (btn[3]) arm.closeShoulder(1);
      if (btn[4]) arm.openElbow(1);
      if (btn[5]) arm.closeElbow(1);
      if (btn[6]) arm.wristLeft(1);
      if (btn[7]) arm.wristRight(1);
      if (btn[8]) arm.openGrip(1);
      if (btn[9]) arm.closeGrip(1);
  }
  delay(16);
}
```

- In the loop function there is a switch statement which executes the code depending on the current controlling mode.
- To change the controlling mode, a push button is connected to the Arduino digital pin 4.

- If the mode is 1 Arduino updates the servo motor rotation corresponding to potentiometers otherwise it reads and process the commands coming through serial communication.

CHAPTER 6. ANDROID APPLICATION (ARM CONTROLLER)

As mentioned in Chapter 1, the HC 05 bluetooth module is mounted on the Arduino board and is used to receive commands through a wireless link between the android and the HC 05 bluetooth module.

6.1 WHAT IS ANDROID?

Android is an operating system based on the Linux kernel with a user interface based on direct manipulation, designed primarily for touchscreen mobile devices such as smartphones and tablet computers, using touch inputs, that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, and a virtual keyboard. Despite being primarily designed for touchscreen input, it also has been used in televisions, games consoles, digital cameras, and other electronics.

6.2 ANDROID PROGRAMMING

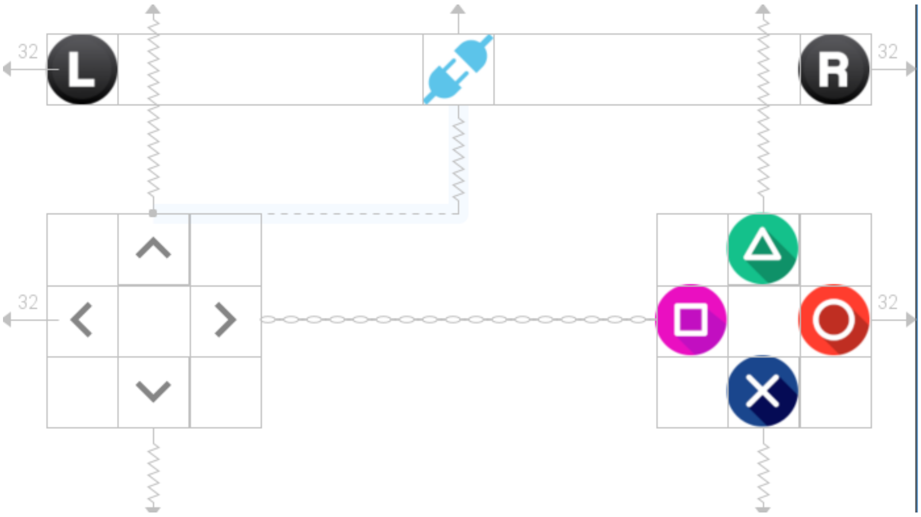
Firstly, a platform is need to write, run and debug the code by user. This platform is **android studio**.

DEVELOPMENT ENVIRONMENT

1. Android phone
2. Android studio IDE
3. minimum SDK version 19
4. Window 10

6.3 MAIN ACTIVITY LAYOUT

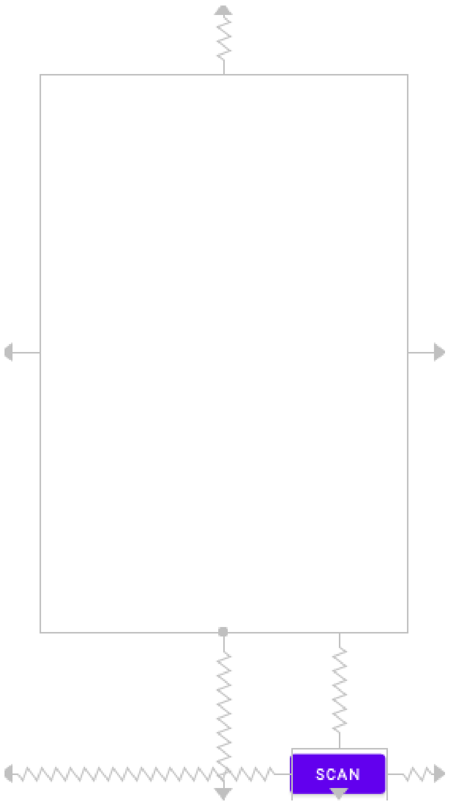
The main activity layout is designed to have a game pad to control the arm.



activity_main.xml layout

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <GridLayout
        android:id="@+id/grid_layout_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@id/grid_layout_2"
        app:layout_constraintHorizontal_chainStyle="spread_inside"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.7">


        <ImageView
            android:id="@+id/button_up"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_row="0"
            android:layout_column="1"
            android:contentDescription="up arrow button"
            android:src="@drawable/ic_button_up" />

        <ImageView
            android:id="@+id/button_left"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_row="1"
            android:layout_column="0"
            android:contentDescription="left arrow button"
            android:src="@drawable/ic_button_left" />

        <ImageView
            android:id="@+id/button_right"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_row="1"
            android:layout_column="2"
            android:contentDescription="right arrow button"
            android:src="@drawable/ic_button_right" />

        <ImageView
            android:id="@+id/button_down"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_row="2"
            android:layout_column="1"
            android:contentDescription="down arrow button"
            android:src="@drawable/ic_button_down" />

    </GridLayout>

    <GridLayout
        android:id="@+id/grid_layout_2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="32dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/grid_layout_1"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.7">

        <ImageView
            android:id="@+id/triangle_button"
            android:layout_width="@dimen/button_size"
            android:layout_height="@dimen/button_size"
            android:layout_row="0"
            android:layout_column="1"
            android:contentDescription="triangle button"
            android:src="@drawable/triangle_button" />

        <ImageView
            android:id="@+id/square_button"
            android:layout_width="@dimen/button_size"
            android:layout_height="@dimen/button_size"
            android:layout_row="1"
            android:layout_column="0"
            android:contentDescription="sqaure button"
            android:src="@drawable/square_button"
            app:layout_constraintEnd_toStartOf="@+id/button_right" />

        <ImageView
            android:id="@+id/circle_button"
            android:layout_width="@dimen/button_size"
            android:layout_height="@dimen/button_size"
            android:layout_row="1"
            android:layout_column="2"
            android:contentDescription="circle button"
            android:src="@drawable/circle_button" />

        <ImageView
            android:id="@+id/cross_button"
            android:layout_width="@dimen/button_size"
            android:layout_height="@dimen/button_size"
            android:layout_row="2"
            android:layout_column="1"
            android:contentDescription="cross button"
            android:src="@drawable/cross_button" />

    </GridLayout>

    <GridLayout
        android:id="@+id/grid_layout_3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="32dp"
        android:layout_marginEnd="32dp"
        android:columnCount="3"

        android:rowCount="1"
        app:layout_constraintBottom_toTopOf="@+id/grid_layout_1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.22000003">

        <ImageView
            android:id="@+id/l_button"
            android:layout_width="@dimen/button_size"
            android:layout_height="@dimen/button_size"
            android:layout_row="0"
            android:layout_column="0"
            android:src="@drawable/l_button" />

        <ImageView
            android:id="@+id/r_button"
            android:layout_width="@dimen/button_size"
            android:layout_height="@dimen/button_size"
            android:layout_row="0"
            android:layout_column="2"
            android:src="@drawable/r_button" />

        <ImageView
            android:id="@+id/connect_button"
            android:layout_width="@dimen/button_size"
            android:layout_height="@dimen/button_size"
            android:layout_row="0"
            android:layout_column="1"
            android:layout_gravity="center_horizontal"
            android:src="@drawable/plug_out" />

    </GridLayout>


</androidx.constraintlayout.widget.ConstraintLayout>
```

6.4 SELECT DEVICE ACTIVITY LAYOUT



activity_select_device.xml layout

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SelectDeviceActivity">

    <Button
        android:id="@+id/button_scan"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="106dp"
        android:onClick="onScanButtonPress"
        android:text="scan"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.829"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/listview" />

    <ListView
        android:id="@+id/listview"
        android:layout_width="338dp"
        android:layout_height="511dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.495"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.299" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

6.5 MAIN ACTIVITY CLASS

MainActivity.java

```
package com.arm.controller;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.os.Message;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.content.res.AppCompatResources;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;

public class MainActivity extends AppCompatActivity {

    int REQ_CODE_EN_BT = 1;
    int REQ_CODE_SELECT_REMOTE_DEVICE = 2;
    final int CONNECTION_IS_ENABLED = 3;
    final int CONNECTION_IS_DISABLED = 4;

    BluetoothAdapter mBlueAdapter;
    BluetoothSocket mSocket;

    ImageView mButton_up;
    ImageView mButton_down;
    ImageView mButton_left;
    ImageView mButton_right;

    ImageView mButton_triangle;
    ImageView mButton_cross;
    ImageView mButton_square;
    ImageView mButton_circle;

    ImageView mButton_l;
    ImageView mButton_r;

    ImageView mButton_connect;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mBlueAdapter = BluetoothAdapter.getDefaultAdapter();
        if (mBlueAdapter == null) {
        } else if(!mBlueAdapter.isEnabled()){
            // enable the bluetooth
            Intent enableBluetooth = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBluetooth, REQ_CODE_EN_BT);
        }

        mButton_up = findViewById(R.id.button_up);
        mButton_down = findViewById(R.id.button_down);
        mButton_left = findViewById(R.id.button_left);
        mButton_right = findViewById(R.id.button_right);

        mButton_triangle = findViewById(R.id.triangle_button);
        mButton_cross = findViewById(R.id.cross_button);
        mButton_square = findViewById(R.id.square_button);
        mButton_circle = findViewById(R.id.circle_button);

        mButton_l = findViewById(R.id.l_button);
        mButton_r = findViewById(R.id.r_button);

        mButton_connect = findViewById(R.id.connect_button);

        /* buttons click event */
        mButton_up.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    write("w".getBytes());
                    return true;
                }

                if (event.getAction() == MotionEvent.ACTION_UP) {
                    write("w".getBytes());
                    return true;
                }

                return false;
            }

        });
        mButton_down.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    write("s".getBytes());
                    return true;
                }

                if (event.getAction() == MotionEvent.ACTION_UP) {
                    write("s".getBytes());
                    return true;
                }
                return false;
            }
        });
        mButton_left.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    write("a".getBytes());
                    return true;

                }

                if (event.getAction() == MotionEvent.ACTION_UP) {
                    write("a".getBytes());
                    return true;

                }
                return false;
            }
        });
        mButton_right.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    Log.i("UP", "down");
                    write("d".getBytes());
                    return true;

                }

                if (event.getAction() == MotionEvent.ACTION_UP) {
                    Log.i("UP", "up");
                    write("d".getBytes());
                    return true;

                }
                return false;
            }
        });
        mButton_triangle.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    write("i".getBytes());
                    return true;

                }

                if (event.getAction() == MotionEvent.ACTION_UP) {
                    write("i".getBytes());
                    return true;

                }
                return false;
            }
        });
        mButton_cross.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    write("k".getBytes());
                    return true;

                }

                if (event.getAction() == MotionEvent.ACTION_UP) {
                    write("k".getBytes());
                    return true;

                }
                return false;
            }
        });
        mButton_square.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    write("j".getBytes());
                    return true;

                }

                if (event.getAction() == MotionEvent.ACTION_UP) {
                    write("j".getBytes());
                    return true;

                }
                return false;
            }
        });
        mButton_circle.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    write("l".getBytes());
                    return true;
                }

                if (event.getAction() == MotionEvent.ACTION_UP) {
                    write("l".getBytes());
                    return true;
                }
                return false;
            }
        });
        mButton_l.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    write("q".getBytes());
                    return true;
                }

                if (event.getAction() == MotionEvent.ACTION_UP) {
                    write("q".getBytes());
                    return true;
                }
                return false;
            }
        });
        mButton_r.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                if (event.getAction() == MotionEvent.ACTION_DOWN) {
                    write("e".getBytes());
                    return true;
                }

                if (event.getAction() == MotionEvent.ACTION_UP) {
                    write("e".getBytes());
                    return true;
                }
                return false;
            }
        });

        mButton_connect.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startSelectActivity();
            }
        });


        registerReceiver(new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                handler.obtainMessage(CONNECTION_IS_DISABLED).sendToTarget();
            }
        }, new IntentFilter(BluetoothDevice.ACTION_ACL_DISCONNECTED));
    }


    public void startSelectActivity() {
        Intent intent = new Intent(this, SelectDeviceActivity.class);
        startActivityForResult(intent, REQ_CODE_SELECT_REMOTE_DEVICE );
    }


    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQ_CODE_EN_BT && resultCode == RESULT_OK) {
            Toast.makeText(getApplicationContext(), "bluetooth is enabled", Toast.LENGTH_SHORT).show();
        }

        if (requestCode == REQ_CODE_SELECT_REMOTE_DEVICE && resultCode == RESULT_OK) {
            String address = data.getStringExtra("EXTRA_DEVICE_ADDRESS");
            new Thread(
                    new Runnable() {
                        @Override
                        public void run() {
                            requestRfConnection(address);
                        }
                    }
            ).start();
        }
    }


    @Override
    public void onBackPressed() {
        super.onBackPressed();
        if (mSocket != null && mSocket.isConnected()) {
            try {
                mSocket.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    public void requestRfConnection(String remoteDeviceAddress) {
        BluetoothDevice remoteDevice = mBlueAdapter.getRemoteDevice(remoteDeviceAddress);
        UUID uuid = remoteDevice.getUuids()[0].getUuid();
        BluetoothSocket temp = null;
        try {
            temp = remoteDevice.createInsecureRfcommSocketToServiceRecord(uuid);
            temp.connect();
            mSocket = temp;
            handler.obtainMessage(CONNECTION_IS_ENABLED).sendToTarget();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }



    public void write(byte... b){
        try {
            if (mSocket != null && mSocket.isConnected()) {
                OutputStream out = mSocket.getOutputStream();
                out.write(b);
            }
        } catch (IOException e) {

        }
    }



    Handler handler = new Handler(Looper.getMainLooper()) {
        @Override
        public void handleMessage(Message msg) {
            switch(msg.what) {
                case CONNECTION_IS_ENABLED:
                    mButton_connect.setImageDrawable(AppCompatResources.getDrawable(getApplicationContext(), R.drawable.plug_in)); break;
                case CONNECTION_IS_DISABLED:
                    mButton_connect.setImageDrawable(AppCompatResources.getDrawable(getApplicationContext(), R.drawable.plug_out)); break;
            }
        }
    };

}
```

6.6 SELECT DEVICE ACTIVITY

SelectDeviceActivity.java

```

package com.arm.controller;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.Set;

public class SelectDeviceActivity extends AppCompatActivity {

    private ListView mListView;
    private Button mScanButton;

    private ArrayAdapter<BluetoothDeviceWrapper> arrayAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_select_device);

        BluetoothAdapter ba = BluetoothAdapter.getDefaultAdapter();
        Set<BluetoothDevice> remoteDevices = ba.getBondedDevices();

        mListView = findViewById(R.id.listview);
        mScanButton = findViewById(R.id.button_scan);

        arrayAdapter = new ArrayAdapter<>(getApplicationContext(), android.R.layout.simple_list_item_1);
        for (BluetoothDevice remoteDevice : remoteDevices) {
            arrayAdapter.add(new BluetoothDeviceWrapper(remoteDevice));
        }
        mListView.setAdapter(arrayAdapter);


        mListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                BluetoothDeviceWrapper item = (BluetoothDeviceWrapper) parent.getItemAtPosition(position);
                Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                intent.putExtra("EXTRA_DEVICE_ADDRESS", item.remoteDevice.getAddress());
                setResult(RESULT_OK, intent);
                finish();
            }
        });
    }


    public void onScanButtonPress(View view) {
//        if (!mBlueAdapter.isDiscovering()) {
//            mBlueAdapter.startDiscovery();
//        }
    }
}
```

6.7 BLUETOOTH DEVICE WRAPPER

BluetoothDeviceWrapper.java

```java
package com.arm.controller;

import android.bluetooth.BluetoothDevice;


public class BluetoothDeviceWrapper {
    public BluetoothDevice remoteDevice;

    public BluetoothDeviceWrapper(BluetoothDevice remoteDevice) {
        this.remoteDevice = remoteDevice;
    }

    @Override
    public String toString() {
        return remoteDevice.getName() +  " " + remoteDevice.getAddress();
    }
}
```

CONCLUSION

This project can be developed further in the future. Better results can be obtained if servo motors have better angle positioning, higher torque and wider angle movement. Servo motor positions can be read by encoder sensors to get better angle positioning. Better and stronger materials can be used in the body to avoid shaking while servo motors are turning. In order to lift high weights, servo motors that have higher torques can be chosen. This is a low budget and low cost project and its main purpose is controlling a system from android application via HC 05 bluetooth module.

REFERENCE

[1] Simon Monk - Arduino + Android Projects for the Evil Genius Control Arduino with Your Smartphone or Tablet (2011, McGraw-HillTab Electronics).

[2] Mohammed, A.A. and Sunar, M., 2015, May. Kinematics modeling of a 4-DOF robotic arm. In Control, Automation and Robotics (ICCAR), 2015 International Conference on (pp. 87-91). IEEE.

[3] E. Punna, R. K. Nenavath, and B. Maloth, "Labview Controlled Robotic ARM."

[4] C. A. Schuler and W. L. McNamee, Industrial electronics and robotics: McGraw-Hill, Inc.,
1986.

[5] chapter 18 [Communicating with Bluetooth, NFC, and Wi-Fi Peer-to-Peer] from “Lake, Ian_ Meier, Reto - Professional Android-John Wiley & Sons (2018)”.

[6] Zhang, Z., 2015. Wearable sensor technologies applied for post-stroke rehabilitation (Doctoral dissertation, RMIT University).

[7] Qassem, M.A., Abuhadrous, I. and Elaydi, H., 2010, March. Modeling and Simulation of 5 DOF educational robot arm. In Advanced Computer Control (ICACC), 2010 2nd International Conference on (Vol. 5, pp. 569-574). IEEE.

[8] Bhuyan, A.I. and Mallick, T.C., 2014, October. Gyro-accelerometer based control of a robotic Arm using AVR microcontroller. In Strategic Technology (IFOST), 2014 9th International Forum on (pp. 409-413). IEEE.

[9] Anon, 2016. The Ninth International Symposium. [online] Robotics Research. Available at: http://Robotics Research: The Ninth International Symposium [Accessed 4 Apr. 2016]

[10] Condit, R. and Jones, D.W., 2004. Stepping motors fundamentals. Microchip Application Note: AN907,[Online]. Available: www. microchip. Com