

**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**

**(Assistant Professor)**

**Project Guide**

**Mr. Basanta Mahato**

**(Assistant Professor)**

**Project Coordinator**

**Prof. (Dr.) Seema  
Nayak**

**(Professor)**

**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

<b>STUDENT'S DECLARATION</b>	<b>I</b>
<b>ACKNOWLEDGEMENT</b>	<b>III</b>
<b>ABSTRACT</b>	<b>VI</b>
<b>CHAPTER 1. 5 DOF ROBOTIC ARM</b>	<b>1</b>
1.1 WHAT IS ROBOTIC ARM?	1
1.2 DEGREE OF FREEDOM	2
1.3 ARM DESIGN & MECHANICAL CONFIGURATION	3
1.4 REQUIRED COMPONENTS	4
1.5 HC 05 BLUETOOTH MODULE	5
<i>HC-05 Specification</i>	6
<i>Pin Description</i>	6
<b>CHAPTER 2. READING DATA FROM POTENTIOMETER</b>	<b>8</b>
2.1 INTRODUCTION OF POTENTIOMETER	8
2.2 WIRING DIAGRAM ( CIRCUIT DIAGRAM )	9
2.3 READING THE VALUE OF THE POTENTIOMETER AND CONVERTING IT INTO AN ANGLE	9
<b>CHAPTER 3. CONTROLLING THE SERVO MOTORS</b>	<b>11</b>
3.1 SERVO MOTOR	11
3.2 THE WORKING PRINCIPLE OF SERVO MOTOR	12
3.3 CONTROLLING THE SERVO MOTOR WITH ARDUINO	13
<i>peace of code to control the servo motor</i>	13
<b>CHAPTER 4. DISPLAYING TEXT ON THE OLED SCREEN</b>	<b>14</b>
4.1 OLED SCREEN	14
4.2 CODE TO DISPLAY TEXT ON OLED SCREEN	15
<b>CHAPTER 5. FINALIZING THE COMPLETE CODE FOR ARDUINO</b>	<b>16</b>
5.1 CLASS ROBOTARM	16
5.2 CLASS KNOBCONTROLLER	18
5.3 CLASS OLEDDISPLAY	19
5.4 COMPLETE CODE FOR ARDUINO	20
<b>CHAPTER 6. ANDROID APPLICATION (ARM CONTROLLER)</b>	<b>25</b>
6.1 WHAT IS ANDROID?	25
6.2 ANDROID PROGRAMMING	25
<i>DEVELOPMENT ENVIRONMENT</i>	25
6.3 MAIN ACTIVITY LAYOUT	25
<i>activity_main.xml layout</i>	26
6.4 SELECT DEVICE ACTIVITY LAYOUT	30
<i>activity_select_device.xml layout</i>	30
6.5 MAIN ACTIVITY CLASS	31
<i>MainActivity.java</i>	31

6.6 SELECT DEVICE ACTIVITY	38
<i>SelectDeviceActivity.java</i>	38
6.7 BLUETOOTH DEVICE WRAPPER	40
<i>BluetoothDeviceWrapper.java</i>	40
<b>CONCLUSION</b>	<b>41</b>
<b>REFERENCE</b>	<b>41</b>

# **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.

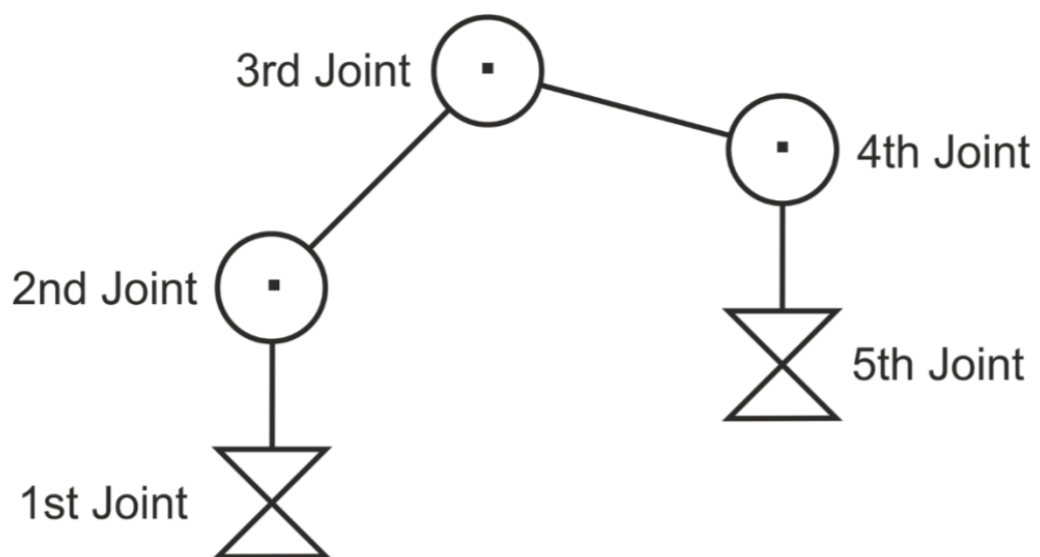
Robot arms, while operating in a similar manner to human arms, can nevertheless have a considerably broader range of motion because their design is entirely up to the creator's creativity. For example, the joint connecting the segments of a robotic arm can rotate as well as move like a hinge. The end 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.



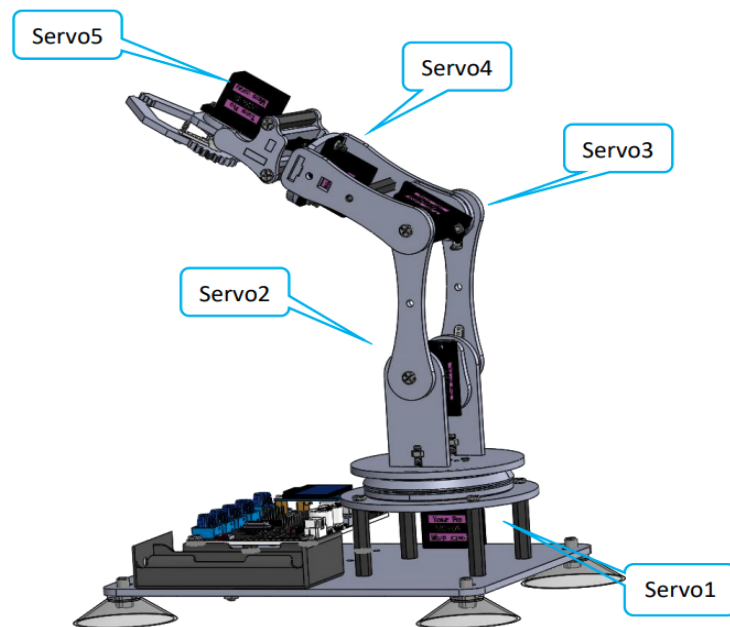
## 1.2 DEGREE OF FREEDOM

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.



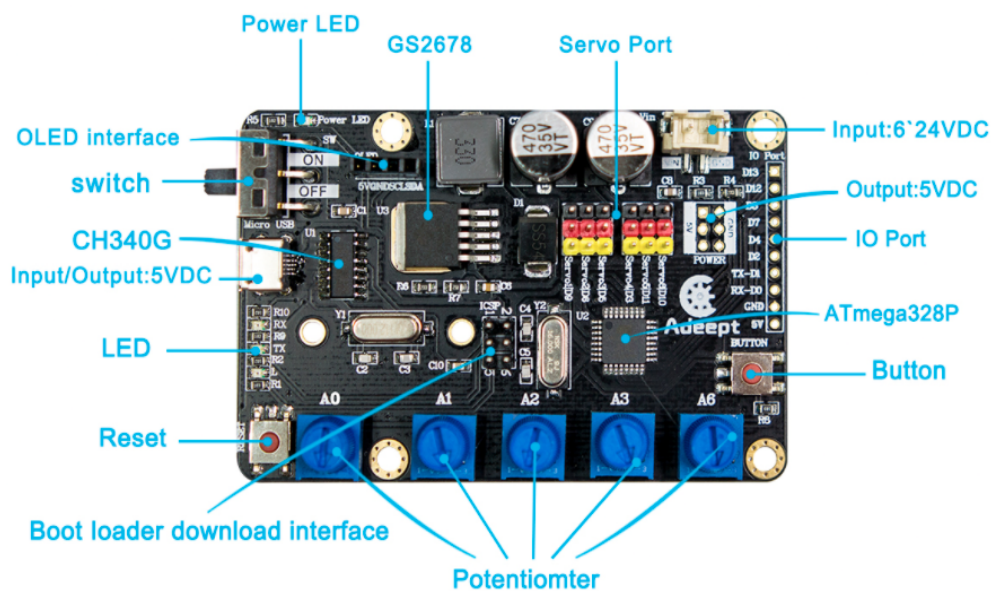
**Figure 1.1 5-DOF Robotic Arm Representation**

### 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.

<b>Actuators</b>	<b>Joints</b>	<b>Arduino pwm pins</b>
servo 1	base	9
servo 2	shoulder	6
servo 3	elbow	5
servo 4	wrist	3
servo 5	gripper	11

## 1.4 REQUIRED COMPONENTS

<i>Components</i>	<i>quantity</i>
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

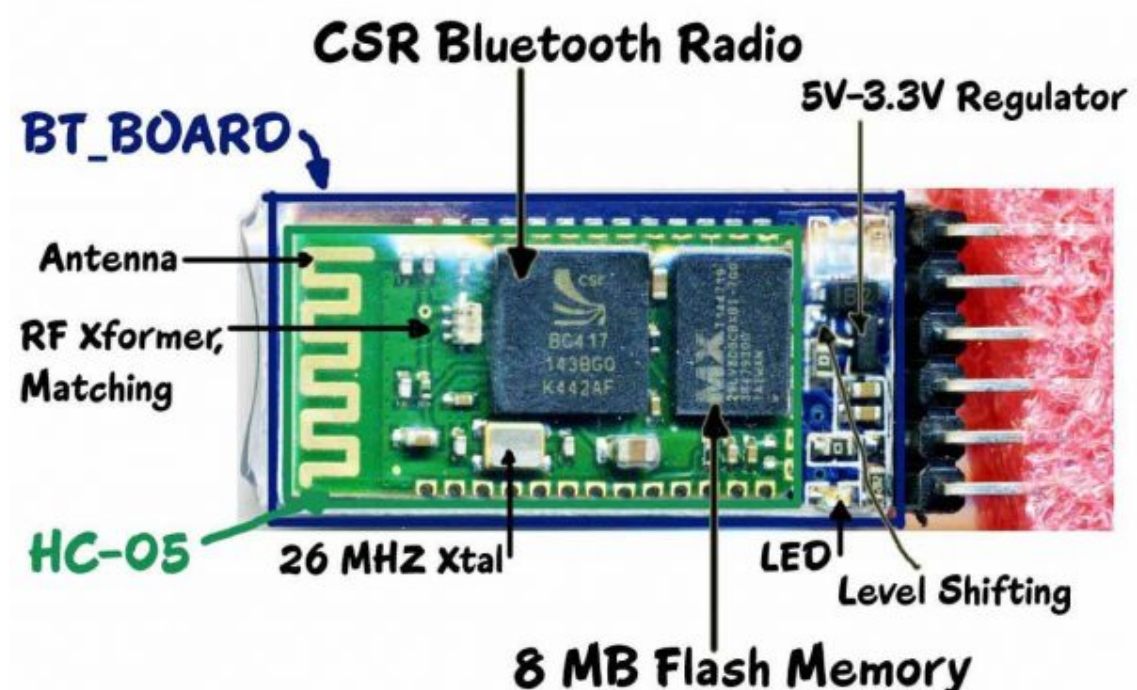
A bluetooth module is installed on the Arduino board which can receive data through bluetooth from android mobile so we can control the arm with mobile.

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:  $\leq 4\text{dBm}$ , Class 2
5. Sensitivity:  $\leq -84\text{dBm}$  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:  $-20 \sim +75\text{Centigrade}$
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

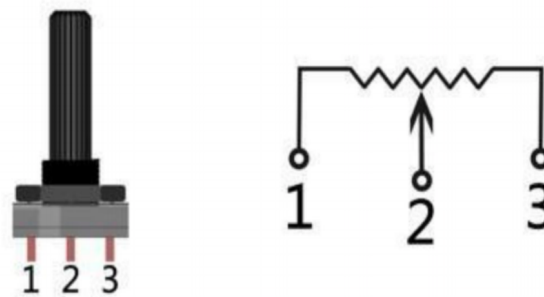
serially on TXD pin).

**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.

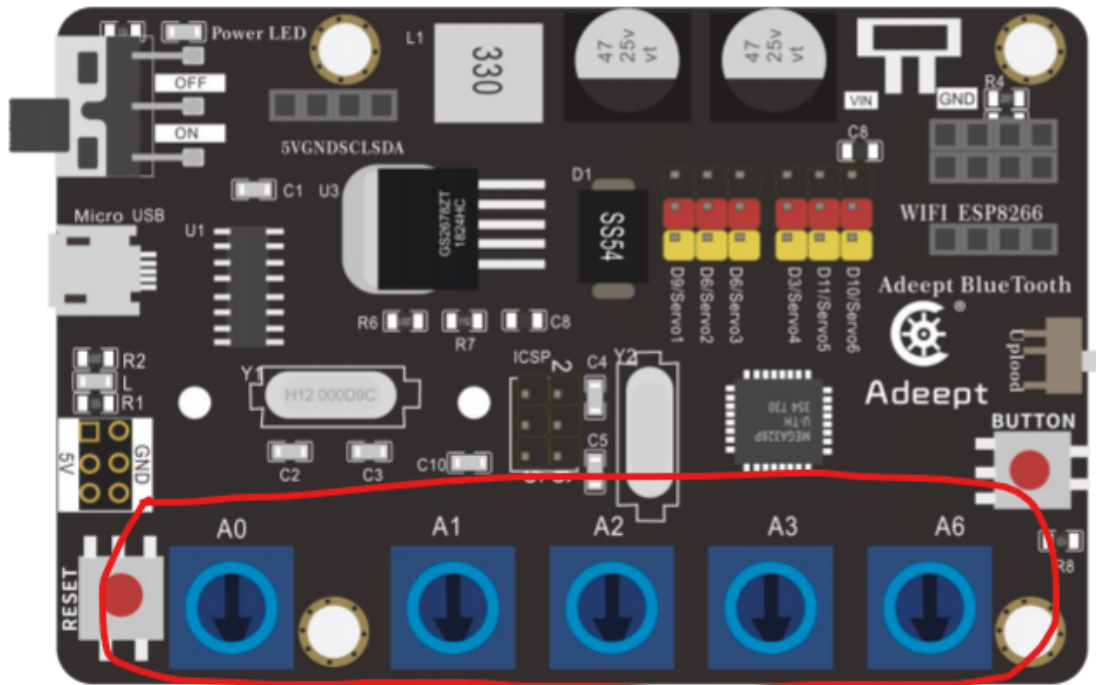


figure 2.2 potentiometers on arm drive board.

## 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.

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

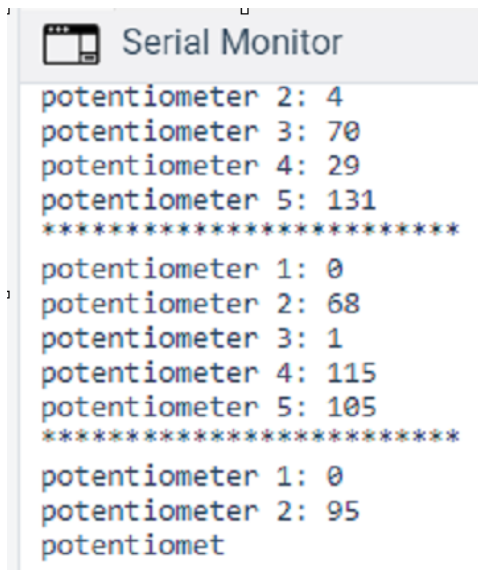


```

3  #define P3_PIN 2
4  #define P4_PIN 3
5  #define P5_PIN 6
6
7  int pot_1, pot_2, pot_3, pot_4, pot_5;
8
9  void setup() {
10   Serial.begin(9600);
11 }
12
13 void loop() {
14   // read data from potentiometers
15   pot_1 = map(analogRead(P1_PIN), 0, 1023, 0, 180);
16   pot_2 = map(analogRead(P2_PIN), 0, 1023, 0, 180);
17   pot_3 = map(analogRead(P3_PIN), 0, 1023, 0, 180);
18   pot_4 = map(analogRead(P4_PIN), 0, 1023, 0, 180);
19   pot_5 = map(analogRead(P5_PIN), 0, 1023, 0, 180);
20
21   // print out read values
22   delay(15);
23
24   Serial.print("potentiometer 1: "); Serial.println(pot_1);
25   Serial.print("potentiometer 2: "); Serial.println(pot_2);
26   Serial.print("potentiometer 3: "); Serial.println(pot_3);
27   Serial.print("potentiometer 4: "); Serial.println(pot_4);
28   Serial.print("potentiometer 5: "); Serial.println(pot_5);

```

## OUTPUT



```

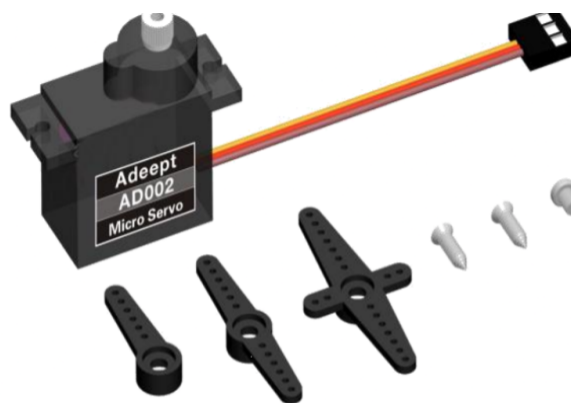
potentiometer 2: 4
potentiometer 3: 70
potentiometer 4: 29
potentiometer 5: 131
*****
potentiometer 1: 0
potentiometer 2: 68
potentiometer 3: 1
potentiometer 4: 115
potentiometer 5: 105
*****
potentiometer 1: 0
potentiometer 2: 95
potentiomet

```

## 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.

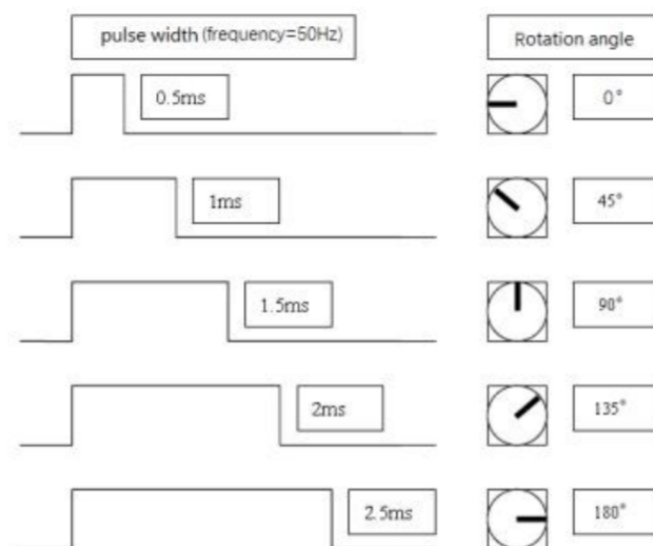


figure 3.2 Servo rotation corresponding to pulse

### 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*

```
1  //...
2  #include<Servo.h>
3  Servo servo;                      // create an object of Servo
4
5  void setup() {
6      //..
7      servo.attach(9);              // attach the servo to pin 9
8  }
9
10 void loop() {
11 // read data from potentiometer connected to pin a0
12 int value = readAnalog(0);
13 int angle = map(0, 1023, 0, 180);
14
15 // update the rotation of servo motor
16 servo.write(angle);
17 }
```

## 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 Electroluminescence 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 Adept Arm Drive Board.

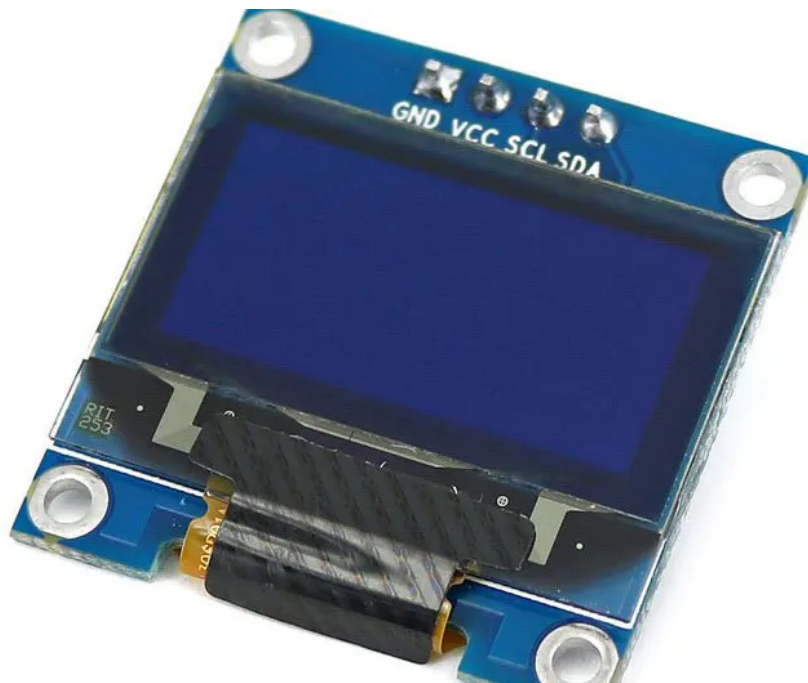


figure 4.1 OLED DISPLAY

## 4.2 CODE TO DISPLAY TEXT ON OLED SCREEN

```
1  #include<Adafruit_SSD1306.h>
2  #include<Adafruit_GFX.h>
3  #include<Wire.h>
4
5  #define OLED_RESET 4
6  #define SCREEN_ADDRESS 0x3c
7  #define OLED_WIDTH 128
8  #define OLED_HEIGHT 64
9
10 Adafruit_SSD1306 oled;
11
12 void setup() {
13     // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V
    internally
14     if (!oled.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
15         Serial.println(F("SSD1306 allocation failed"));
16         for (;;); // Don't proceed, loop forever
17     }
18
19     oled.display();
20     delay(2000);           // display the logo for 2 seconds
21 }
22
23 void loop() {
24     clear(); // clear the display
25     drawString(String("Hello world!"), 1, 1, 2); // // show hello
    world
26 }
27
28 void drawString(String str, int x, int y, int s) {
29     oled.setTextColor(WHITE);
30     oled.setTextSize(s);
31     oled.setCursor(x, y);
32     oled.print(str);
33     oled.display();
34 }
35
36 void clear() {
37     oled.clearDisplay();
38 }
```

## 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.

```
1  class RobotArm {
2      private:
3          Servo base;
4          Servo shoulder;
5          Servo elbow;
6          Servo wrist;
7          Servo grip;
8
9          int baseAngle = 90;
10         int shoulderAngle = 90;
11         int elbowAngle = 90;
12         int wristAngle = 90;
13         int gripAngle = 90;
14
15     public:
16         void setServoPins(int p1, int p2, int p3, int p4, int p5)
17         {
18             base.attach(p1);
19             shoulder.attach(p2);
20             elbow.attach(p3);
21             wrist.attach(p4);
22             grip.attach(p5);
23         }
24
25         void init()
26         {
27             base.write(90);
28             shoulder.write(90);
29             elbow.write(90);
30             wrist.write(90);
31             grip.write(90);
32         }
33
34         void setBaseRot(int a) {
35             base.write(a);
36         }
37
38         void setShoulderRot(int a) {
39             shoulder.write(a);
40         }
```

```

41
42 void setElbowRot(int a) {
43     elbow.write(a);
44 }
45
46 void setWristRot(int a)
47 {
48     wrist.write(a);
49 }
50
51 void setGripRot(int a)
52 {
53     grip.write(a);
54 }
55
56 void rotateLeft(int dr)
57 {
58     baseAngle += dr;
59     if (baseAngle > 180) baseAngle = 180;
60     base.write(baseAngle);
61 }
62
63 void rotateRight(int dr)
64 {
65     baseAngle -= dr;
66     if (baseAngle < 0) baseAngle = 0;
67     base.write(baseAngle);
68 }
69
70 void openShoulder(int dr)
71 {
72     shoulderAngle -= dr;
73     if (shoulderAngle < 0) shoulderAngle = 0;
74     shoulder.write(shoulderAngle);
75 }
76
77 void closeShoulder(int dr)
78 {
79     shoulderAngle += dr;
80     if (shoulderAngle > 180) shoulderAngle = 180;
81     shoulder.write(shoulderAngle);
82 }
83
84
85 // control the elbow at the elbow
86 void openElbow(int dr)
87 {
88     elbowAngle += dr;
89     if (elbowAngle > 180) elbowAngle = 180;
90     elbow.write(elbowAngle);
91 }
92
93 void closeElbow(int dr)
94 {

```



```

95     elbowAngle -= dr;
96     if (elbowAngle < 0) elbowAngle = 0;
97     elbow.write(elbowAngle);
98 }
99
100
101
102 void wristLeft(int dr)
103 {
104     wristAngle += dr;
105     if (wristAngle > 180) wristAngle = 180;
106     wrist.write(wristAngle);
107 }
108
109 void wristRight(int dr)
110 {
111     wristAngle -= dr;
112     if (wristAngle < 0) wristAngle = 0;
113     wrist.write(wristAngle);
114 }
115
116
117
118 void openGrip(int dr)
119 {
120     gripAngle -= dr;
121     if (gripAngle < 0) gripAngle = 0;
122     grip.write(gripAngle);
123 }
124
125 void closeGrip(int dr)
126 {
127     gripAngle += dr;
128     if (gripAngle > 90) gripAngle = 90;
129     grip.write(gripAngle);
130 }
131 };

```

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.

```

1 class KnobController {
2     private:
3         // hold int data in the range from 0 to 180
4         int pot_1;
5         int pot_2;
6         int pot_3;
7         int pot_4;
8         int pot_5;
9
10        RobotArm m_arm;
11
12    public:
13        KnobController(RobotArm robot_arm) : m_arm(robot_arm) { }
14
15        void poll() {
16            pot_1 = map(analogRead(P1_PIN), 0, 1023, 0, 180);
17            pot_2 = map(analogRead(P2_PIN), 0, 1023, 0, 180);
18            pot_3 = map(analogRead(P3_PIN), 0, 1023, 0, 180);
19            pot_4 = map(analogRead(P4_PIN), 0, 1023, 0, 180);
20            pot_5 = map(analogRead(P5_PIN), 0, 1023, 0, 180);
21        }
22
23        void step() {
24            m_arm.setBaseRot(pot_1);
25            m_arm.setShoulderRot(pot_2);
26            m_arm.setElbowRot(pot_3);
27            m_arm.setWristRot(pot_4);
28            m_arm.setGripRot(pot_5);
29        }
30 };

```

### 5.3 class OledDisplay

```

1 class OledDisplay {
2     private:
3         Adafruit_SSD1306 oled;
4
5     public:
6         OledDisplay(): oled(128, 64, &Wire, OLED_RESET) {}
7
8         void init() {
9             // SSD1306_SWITCHCAPVCC = generate display voltage from
            3.3V internally
10            if (!oled.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
11                Serial.println(F("SSD1306 allocation failed"));

```

```

12     for (;;); // Don't proceed, loop forever
13 }
14
15     oled.display();
16     delay(1500);
17 }
18
19 void clear() {
20     oled.clearDisplay();
21 }
22
23 void drawString(String str, int x, int y, int s) {
24     oled.setTextColors(WHITE);
25     oled.setTextSize(s);
26     oled.setCursor(x, y);
27     oled.print(str);
28     oled.display();
29 }
30 };

```

## 5.4 COMPLETE CODE FOR ARDUINO

```

1  #include<Servo.h>
2  #include<SoftwareSerial.h>
3  #include<Adafruit_SSD1306.h>
4  #include<Adafruit_GFX.h>
5  #include<Wire.h>
6
7  // servo pins
8  #define BASE_PIN 9
9  #define SHOULDER_PIN 6
10 #define ELBOW_PIN 5
11 #define WRIST_PIN 3
12 #define GRIPPER_PIN 11
13
14 // pins for potentiometers
15 #define P1_PIN 0
16 #define P2_PIN 1
17 #define P3_PIN 2
18 #define P4_PIN 3
19 #define P5_PIN 6
20
21 // Declaration for an SSD1306 display connected to I2C (SDA, SCL
    pins)
22 // The pins for I2C are defined by the Wire-library.
23 // On an arduino UNO:      A4(SDA), A5(SCL)

```

```

24 #define OLED_RESET 4
25 #define SCREEN_ADDRESS 0x3c
26 #define OLED_WIDTH 128
27 #define OLED_HEIGHT 64
28
29
30
31 //-----
32 //      STARTS FROM HERE
33 //-----
34
35 RobotArm arm;
36 KnobController knobController(arm);
37 OledDisplay scr;
38 SoftwareSerial btSerial(7, 8); // RX | TX
39 int mode = 2;                  // by default
40
41
42 // imaginary (fake) controlling buttons can be pressed and
   released via Serial communication.
43 // -----
44 // buttons      //      when button is down
45 // -----
46 // btn_0        //      rotate left
47 // btn_1        //      rotate Right
48 // btn_2        //      open shoulder
49 // btn_3        //      close shoulder
50 // btn_4        //      open elbow
51 // btn_5        //      close elbow
52 // btn_6        //      rotate wrist clockwise
53 // btn_7        //      rotate wrist anti-clockwise
54 // btn_8        //      open grip
55 // btn_9        //      close grip
56 // -----
57
58
59 boolean btn[10] = {false}; // contains all buttons from 0 to 9,
   boolean true indicates that the button is down, otherwise not.
60 int buttonPressed;          // state for physical button given
   on the board
61
62
63 void setup() {
64   btSerial.begin(9600);
65   Serial.begin(9600);
66
67   pinMode(4, INPUT);
68   pinMode(BASE_PIN, OUTPUT); //Set the servo interface as the

```

```

    output interface
69   pinMode(SHOULDER_PIN, OUTPUT); //Set the servo interface as
    the output interface
70   pinMode(ELBOW_PIN, OUTPUT); //Set the servo interface as the
    output interface
71   pinMode(WRIST_PIN, OUTPUT); //Set the servo interface as the
    output interface
72   pinMode(GRIPPER_PIN, OUTPUT); //Set the servo interface as the
    output interface
73
74   scr.init();
75   arm.setServoPins(BASE_PIN, SHOULDER_PIN, ELBOW_PIN, WRIST_PIN,
    GRIPPER_PIN);
76   arm.init();
77
78   scr.clear();
79   scr.drawString(String("mode: ") + mode, 1, 1, 2);
80
81 }
82
83 void loop() {
84
85   if (digitalRead(4) == LOW) {
86       // if the on board BUTTON is pressed, PIN4 will connect to
87       ground
88       buttonPressed++;
89   } else {
90       buttonPressed = 0;
91   }
92   if (buttonPressed == 1) {
93       // is button down once
94       mode = (mode > 1) ? 1 : ++mode;
95       scr.clear();
96       scr.drawString(String("mode: ") + mode, 1, 1, 2);
97   }
98
99   switch (mode) {
100       case 1:
101           // scrtrolling mode 1 : with potentiometers
102           // read potentiometers at analog pins a0, a1, a2, a3, a6
103           knobsctrroller.poll();
104           knobsctrroller.step();
105           break;
106
107       case 2:
108           // process commands
109           if (Serial.available() > 0) {
110               int command = Serial.read();

```

```

108     if (command == 'a') btn[0] = !btn[0]; else if (command
    == 'd') btn[1] = !btn[1];
109     if (command == 'k') btn[2] = !btn[2]; else if (command
    == 'i') btn[3] = !btn[3];
110     if (command == 's') btn[4] = !btn[4]; else if (command
    == 'w') btn[5] = !btn[5];
111     if (command == 'j') btn[6] = !btn[6]; else if (command
    == 'l') btn[7] = !btn[7];
112     if (command == 'q') btn[8] = !btn[8]; else if (command
    == 'e') btn[9] = !btn[9];
113     if (command == '>') Serial.println('<');
114 }
115
116 while (btSerial.available() > 0) {
117     int command = btSerial.read();
118     if (command == 'a') btn[0] = !btn[0]; else if (command
    == 'd') btn[1] = !btn[1];
119     if (command == 'k') btn[2] = !btn[2]; else if (command
    == 'i') btn[3] = !btn[3];
120     if (command == 's') btn[4] = !btn[4]; else if (command
    == 'w') btn[5] = !btn[5];
121     if (command == 'j') btn[6] = !btn[6]; else if (command
    == 'l') btn[7] = !btn[7];
122     if (command == 'q') btn[8] = !btn[8]; else if (command
    == 'e') btn[9] = !btn[9];
123 }
124
125 if (btn[0]) arm.rotateLeft(1);
126 if (btn[1]) arm.rotateRight(1);
127 if (btn[2]) arm.openShoulder(1);
128 if (btn[3]) arm.closeShoulder(1);
129 if (btn[4]) arm.openElbow(1);
130 if (btn[5]) arm.closeElbow(1);
131 if (btn[6]) arm.wristLeft(1);
132 if (btn[7]) arm.wristRight(1);
133 if (btn[8]) arm.openGrip(1);
134 if (btn[9]) arm.closeGrip(1);
135 }
136 delay(16);
137}

```

- 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**.

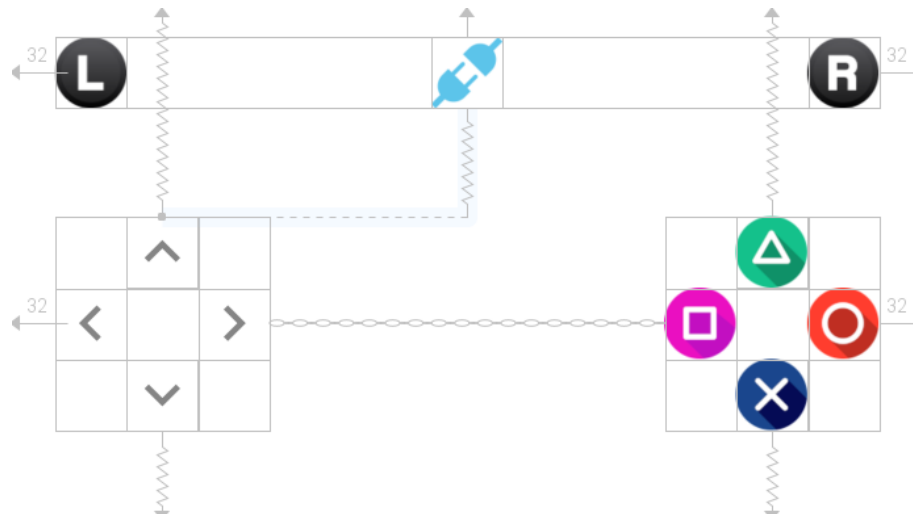
#### ***DEVELOPMENT ENVIRONMENT***

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

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".MainActivity">
9
10     <GridLayout
11         android:id="@+id/grid_layout_1"
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:layout_marginStart="32dp"
15         app:layout_constraintBottom_toBottomOf="parent"
16         app:layout_constraintEnd_toStartOf="@id/grid_layout_2"
17         app:layout_constraintHorizontal_chainStyle="spread_inside"
18         app:layout_constraintStart_toStartOf="parent"
19         app:layout_constraintTop_toTopOf="parent"
20         app:layout_constraintVertical_bias="0.7">
21
22         <ImageView
23             android:id="@+id/button_up"
24             android:layout_width="50dp"
25             android:layout_height="50dp"
26             android:layout_row="0"
27             android:layout_column="1"
28             android:contentDescription="up arrow button"
29             android:src="@drawable/ic_button_up" />
30
31

```

```

32     <ImageView
33         android:id="@+id/button_left"
34         android:layout_width="50dp"
35         android:layout_height="50dp"
36         android:layout_row="1"
37         android:layout_column="0"
38         android:contentDescription="left arrow button"
39         android:src="@drawable/ic_button_left" />
40
41     <ImageView
42         android:id="@+id/button_right"
43         android:layout_width="50dp"
44         android:layout_height="50dp"
45         android:layout_row="1"
46         android:layout_column="2"
47         android:contentDescription="right arrow button"
48         android:src="@drawable/ic_button_right" />
49
50     <ImageView
51         android:id="@+id/button_down"
52         android:layout_width="50dp"
53         android:layout_height="50dp"
54         android:layout_row="2"
55         android:layout_column="1"
56         android:contentDescription="down arrow button"
57         android:src="@drawable/ic_button_down" />
58
59 </GridLayout>
60
61 <GridLayout
62     android:id="@+id/grid_layout_2"
63     android:layout_width="wrap_content"
64     android:layout_height="wrap_content"
65     android:layout_marginStart="16dp"
66     android:layout_marginEnd="32dp"
67     app:layout_constraintBottom_toBottomOf="parent"
68     app:layout_constraintEnd_toEndOf="parent"
69     app:layout_constraintHorizontal_bias="0.5"
70     app:layout_constraintStart_toEndOf="@+id/grid_layout_1"
71     app:layout_constraintTop_toTopOf="parent"
72     app:layout_constraintVertical_bias="0.7">
73
74     <ImageView
75         android:id="@+id/triangle_button"
76         android:layout_width="@dimen/button_size"
77         android:layout_height="@dimen/button_size"
78         android:layout_row="0"
79         android:layout_column="1"

```

```

80         android:contentDescription="triangle button"
81         android:src="@drawable/triangle_button" />
82
83     <ImageView
84         android:id="@+id/square_button"
85         android:layout_width="@dimen/button_size"
86         android:layout_height="@dimen/button_size"
87         android:layout_row="1"
88         android:layout_column="0"
89         android:contentDescription="square button"
90         android:src="@drawable/square_button"
91         app:layout_constraintEnd_toStartOf="@+id/button_right"
92     />
93
94     <ImageView
95         android:id="@+id/circle_button"
96         android:layout_width="@dimen/button_size"
97         android:layout_height="@dimen/button_size"
98         android:layout_row="1"
99         android:layout_column="2"
100        android:contentDescription="circle button"
101        android:src="@drawable/circle_button" />
102
103    <ImageView
104        android:id="@+id/cross_button"
105        android:layout_width="@dimen/button_size"
106        android:layout_height="@dimen/button_size"
107        android:layout_row="2"
108        android:layout_column="1"
109        android:contentDescription="cross button"
110        android:src="@drawable/cross_button" />
111
112    </GridLayout>
113
114    <GridLayout
115        android:id="@+id/grid_layout_3"
116        android:layout_width="match_parent"
117        android:layout_height="wrap_content"
118        android:layout_marginStart="32dp"
119        android:layout_marginEnd="32dp"
120        android:columnCount="3"
121
122        android:rowCount="1"
123        app:layout_constraintBottom_toTopOf="@+id/grid_layout_1"
124        app:layout_constraintEnd_toEndOf="parent"
125        app:layout_constraintHorizontal_bias="0.498"
126        app:layout_constraintStart_toStartOf="parent"
127        app:layout_constraintTop_toTopOf="parent"

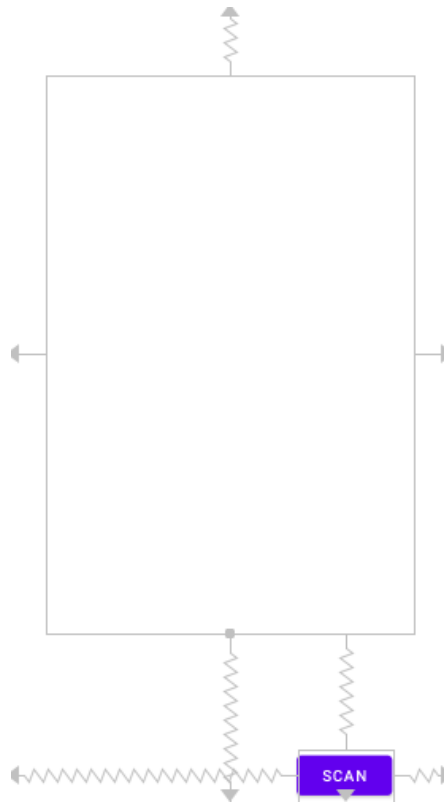
```

```

127         app:layout_constraintVertical_bias="0.22000003">
128
129         <ImageView
130             android:id="@+id/l_button"
131             android:layout_width="@dimen/button_size"
132             android:layout_height="@dimen/button_size"
133             android:layout_row="0"
134             android:layout_column="0"
135             android:src="@drawable/l_button" />
136
137         <ImageView
138             android:id="@+id/r_button"
139             android:layout_width="@dimen/button_size"
140             android:layout_height="@dimen/button_size"
141             android:layout_row="0"
142             android:layout_column="2"
143             android:src="@drawable/r_button" />
144
145         <ImageView
146             android:id="@+id/connect_button"
147             android:layout_width="@dimen/button_size"
148             android:layout_height="@dimen/button_size"
149             android:layout_row="0"
150             android:layout_column="1"
151             android:layout_gravity="center_horizontal"
152             android:src="@drawable/plug_out" />
153
154     </GridLayout>
155
156 </androidx.constraintlayout.widget.ConstraintLayout>

```

## 6.4 SELECT DEVICE ACTIVITY LAYOUT



activity\_select\_device.xml layout

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".SelectDeviceActivity">
9      <Button
10         android:id="@+id/button_scan"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:layout_marginTop="106dp"
14         android:onClick="onScanButtonPress"
15         android:text="scan"
16         app:layout_constraintBottom_toBottomOf="parent"
17         app:layout_constraintEnd_toEndOf="parent"
18         app:layout_constraintHorizontal_bias="0.829"
19         app:layout_constraintStart_toStartOf="parent"
```

```

20         app:layout_constraintTop_toBottomOf="@+id/listview" />
21
22     <ListView
23         android:id="@+id/listview"
24         android:layout_width="338dp"
25         android:layout_height="511dp"
26         app:layout_constraintBottom_toBottomOf="parent"
27         app:layout_constraintEnd_toEndOf="parent"
28         app:layout_constraintHorizontal_bias="0.495"
29         app:layout_constraintStart_toStartOf="parent"
30         app:layout_constraintTop_toTopOf="parent"
31         app:layout_constraintVertical_bias="0.299" />
32
33
34 </androidx.constraintlayout.widget.ConstraintLayout>

```

## 6.5 MAIN ACTIVITY CLASS

### *MainActivity.java*

```

1  package com.arm.controller;
2
3  import android.bluetooth.BluetoothAdapter;
4  import android.bluetooth.BluetoothDevice;
5  import android.bluetooth.BluetoothSocket;
6  import android.content.BroadcastReceiver;
7  import android.content.Context;
8  import android.content.Intent;
9  import android.content.IntentFilter;
10 import android.os.Bundle;
11 import android.os.Handler;
12 import android.os.Looper;
13 import android.os.Message;
14 import android.util.Log;
15 import android.view.MotionEvent;
16 import android.view.View;
17 import android.widget.ImageButton;
18 import android.widget.ImageView;
19 import android.widget.Toast;
20 import androidx.appcompat.app.AppCompatActivity;
21 import androidx.appcompat.content.res.AppCompatResources;
22 import java.io.IOException;
23 import java.io.InputStream;
24 import java.io.OutputStream;
25 import java.util.UUID;
26
27 public class MainActivity extends AppCompatActivity {
28
29     int REQ_CODE_EN_BT = 1;

```

```

30     int REQ_CODE_SELECT_REMOTE_DEVICE = 2;
31     final int CONNECTION_IS_ENABLED = 3;
32     final int CONNECTION_IS_DISABLED = 4;
33
34     BluetoothAdapter mBlueAdapter;
35     BluetoothSocket mSocket;
36
37     ImageView mButton_up;
38     ImageView mButton_down;
39     ImageView mButton_left;
40     ImageView mButton_right;
41
42     ImageView mButton_triangle;
43     ImageView mButton_cross;
44     ImageView mButton_square;
45     ImageView mButton_circle;
46
47     ImageView mButton_l;
48     ImageView mButton_r;
49
50     ImageView mButton_connect;
51
52     @Override
53     protected void onCreate(Bundle savedInstanceState) {
54         super.onCreate(savedInstanceState);
55         setContentView(R.layout.activity_main);
56         mBlueAdapter = BluetoothAdapter.getDefaultAdapter();
57         if (mBlueAdapter == null) {
58             } else if (!mBlueAdapter.isEnabled()){
59                 // enable the bluetooth
60                 Intent enableBluetooth = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
61                 startActivityForResult(enableBluetooth, REQ_CODE_EN_BT);
62             }
63
64             mButton_up = findViewById(R.id.button_up);
65             mButton_down = findViewById(R.id.button_down);
66             mButton_left = findViewById(R.id.button_left);
67             mButton_right = findViewById(R.id.button_right);
68
69             mButton_triangle = findViewById(R.id.triangle_button);
70             mButton_cross = findViewById(R.id.cross_button);
71             mButton_square = findViewById(R.id.square_button);
72             mButton_circle = findViewById(R.id.circle_button);
73
74             mButton_l = findViewById(R.id.l_button);
75             mButton_r = findViewById(R.id.r_button);
76
77             mButton_connect = findViewById(R.id.connect_button);
78
79             /* buttons click event */
80             mButton_up.setOnTouchListener(new View.OnTouchListener() {
81                 @Override
82                 public boolean onTouch(View v, MotionEvent event) {

```

```

83         if (event.getAction() == MotionEvent.ACTION_DOWN) {
84             write("w".getBytes());
85             return true;
86         }
87
88         if (event.getAction() == MotionEvent.ACTION_UP) {
89             write("w".getBytes());
90             return true;
91         }
92
93         return false;
94     }
95
96 });
97 mButton_down.setOnTouchListener(new View.OnTouchListener() {
98     @Override
99     public boolean onTouch(View v, MotionEvent event) {
100         if (event.getAction() == MotionEvent.ACTION_DOWN) {
101             write("s".getBytes());
102             return true;
103         }
104
105         if (event.getAction() == MotionEvent.ACTION_UP) {
106             write("s".getBytes());
107             return true;
108         }
109         return false;
110     }
111 });
112 mButton_left.setOnTouchListener(new View.OnTouchListener() {
113     @Override
114     public boolean onTouch(View v, MotionEvent event) {
115         if (event.getAction() == MotionEvent.ACTION_DOWN) {
116             write("a".getBytes());
117             return true;
118         }
119
120
121         if (event.getAction() == MotionEvent.ACTION_UP) {
122             write("a".getBytes());
123             return true;
124         }
125
126         return false;
127     }
128 });
129 mButton_right.setOnTouchListener(new View.OnTouchListener() {
130     @Override
131     public boolean onTouch(View v, MotionEvent event) {
132         if (event.getAction() == MotionEvent.ACTION_DOWN) {
133             Log.i("UP", "down");
134             write("d".getBytes());
135             return true;
136

```



```

137         }
138
139         if (event.getAction() == MotionEvent.ACTION_UP) {
140             Log.i("UP", "up");
141             write("d".getBytes());
142             return true;
143         }
144     }
145     return false;
146 }
147 });
148 mButton_triangle.setOnTouchListener(new View.OnTouchListener()
149 {
150     @Override
151     public boolean onTouch(View v, MotionEvent event) {
152         if (event.getAction() == MotionEvent.ACTION_DOWN) {
153             write("i".getBytes());
154             return true;
155         }
156
157         if (event.getAction() == MotionEvent.ACTION_UP) {
158             write("i".getBytes());
159             return true;
160         }
161     }
162     return false;
163 }
164 });
165 mButton_cross.setOnTouchListener(new View.OnTouchListener() {
166     @Override
167     public boolean onTouch(View v, MotionEvent event) {
168         if (event.getAction() == MotionEvent.ACTION_DOWN) {
169             write("k".getBytes());
170             return true;
171         }
172     }
173
174     if (event.getAction() == MotionEvent.ACTION_UP) {
175         write("k".getBytes());
176         return true;
177     }
178     return false;
179 }
180 });
181 mButton_square.setOnTouchListener(new View.OnTouchListener() {
182     @Override
183     public boolean onTouch(View v, MotionEvent event) {
184         if (event.getAction() == MotionEvent.ACTION_DOWN) {
185             write("j".getBytes());
186             return true;
187         }
188     }
189 }

```

```

190
191         if (event.getAction() == MotionEvent.ACTION_UP) {
192             write("j".getBytes());
193             return true;
194
195         }
196         return false;
197     }
198 });
199 mButton_circle.setOnTouchListener(new View.OnTouchListener() {
200     @Override
201     public boolean onTouch(View v, MotionEvent event) {
202         if (event.getAction() == MotionEvent.ACTION_DOWN) {
203             write("l".getBytes());
204             return true;
205         }
206
207         if (event.getAction() == MotionEvent.ACTION_UP) {
208             write("l".getBytes());
209             return true;
210         }
211         return false;
212     }
213 });
214 mButton_l.setOnTouchListener(new View.OnTouchListener() {
215     @Override
216     public boolean onTouch(View v, MotionEvent event) {
217         if (event.getAction() == MotionEvent.ACTION_DOWN) {
218             write("q".getBytes());
219             return true;
220         }
221
222         if (event.getAction() == MotionEvent.ACTION_UP) {
223             write("q".getBytes());
224             return true;
225         }
226         return false;
227     }
228 });
229 mButton_r.setOnTouchListener(new View.OnTouchListener() {
230     @Override
231     public boolean onTouch(View v, MotionEvent event) {
232         if (event.getAction() == MotionEvent.ACTION_DOWN) {
233             write("e".getBytes());
234             return true;
235         }
236
237         if (event.getAction() == MotionEvent.ACTION_UP) {
238             write("e".getBytes());
239             return true;
240         }
241         return false;
242     }
243 });

```

```

244
245     mButton_connect.setOnClickListener(new View.OnClickListener() {
246         @Override
247         public void onClick(View v) {
248             startSelectActivity();
249         }
250     });
251
252
253     registerReceiver(new BroadcastReceiver() {
254         @Override
255         public void onReceive(Context context, Intent intent) {
256             handler.obtainMessage(CONNECTION_IS_DISABLED).sendToTarget();
257         }
258     }, new IntentFilter(BluetoothDevice.ACTION_ACL_DISCONNECTED));
259 }
260
261
262 public void startSelectActivity() {
263     Intent intent = new Intent(this, SelectDeviceActivity.class);
264     startActivityForResult(intent, REQ_CODE_SELECT_REMOTE_DEVICE );
265 }
266
267
268 @Override
269 protected void onActivityResult(int requestCode, int resultCode,
    Intent data) {
270     super.onActivityResult(requestCode, resultCode, data);
271     if (requestCode == REQ_CODE_EN_BT && resultCode == RESULT_OK) {
272         Toast.makeText(getApplicationContext(), "bluetooth is
    enabled", Toast.LENGTH_SHORT).show();
273     }
274
275     if (requestCode == REQ_CODE_SELECT_REMOTE_DEVICE && resultCode
    == RESULT_OK) {
276         String address =
    data.getStringExtra("EXTRA_DEVICE_ADDRESS");
277         new Thread(
278             new Runnable() {
279                 @Override
280                 public void run() {
281                     requestRfConnection(address);
282                 }
283             }
284         ).start();
285     }
286 }
287
288
289 @Override
290 public void onBackPressed() {
291     super.onBackPressed();
292     if (mSocket != null && mSocket.isConnected()) {

```

```

293         try {
294             mSocket.close();
295         } catch (IOException e) {
296             e.printStackTrace();
297         }
298     }
299 }
300
301 public void requestRfConnection(String remoteDeviceAddress) {
302     BluetoothDevice remoteDevice =
303         mBlueAdapter.getRemoteDevice(remoteDeviceAddress);
304     UUID uuid = remoteDevice.getUuids()[0].getUuid();
305     BluetoothSocket temp = null;
306     try {
307         temp =
308             remoteDevice.createInsecureRfcommSocketToServiceRecord(uuid);
309         temp.connect();
310         mSocket = temp;
311     } catch (IOException e) {
312         e.printStackTrace();
313     }
314
315     handler.obtainMessage(CONNECTION_IS_ENABLED).sendToTarget();
316 }
317
318 public void write(byte... b){
319     try {
320         if (mSocket != null && mSocket.isConnected()) {
321             OutputStream out = mSocket.getOutputStream();
322             out.write(b);
323         }
324     } catch (IOException e) {
325     }
326 }
327
328
329
330 Handler handler = new Handler(Looper.getMainLooper()) {
331     @Override
332     public void handleMessage(Message msg) {
333         switch(msg.what) {
334             case CONNECTION_IS_ENABLED:
335                 mButton_connect.setImageDrawable(AppCompatResources.getDrawable(getApplicationContext(), R.drawable.plug_in)); break;
336             case CONNECTION_IS_DISABLED:
337                 mButton_connect.setImageDrawable(AppCompatResources.getDrawable(getApplicationContext(), R.drawable.plug_out)); break;
338         }
339     }

```

```
340     };
341
342 }
```

## 6.6 SELECT DEVICE ACTIVITY

### *SelectDeviceActivity.java*

```
1
2 package com.arm.controller;
3
4 import android.bluetooth.BluetoothAdapter;
5 import android.bluetooth.BluetoothDevice;
6 import android.content.Intent;
7 import android.os.Bundle;
8 import android.view.View;
9 import android.widget.AdapterView;
10 import android.widget.AdapterView;
11 import android.widget.Button;
12 import android.widget.ListView;
13
14 import androidx.appcompat.app.AppCompatActivity;
15
16 import java.util.Set;
17
18 public class SelectDeviceActivity extends AppCompatActivity {
19
20     private ListView mListView;
21     private Button mScanButton;
22
23     private ArrayAdapter<BluetoothDeviceWrapper> arrayAdapter;
24
25     @Override
26     protected void onCreate(Bundle savedInstanceState) {
27         super.onCreate(savedInstanceState);
28         setContentView(R.layout.activity_select_device);
29
30         BluetoothAdapter ba = BluetoothAdapter.getDefaultAdapter();
31         Set<BluetoothDevice> remoteDevices = ba.getBondedDevices();
32
33         mListView = findViewById(R.id.listview);
34         mScanButton = findViewById(R.id.button_scan);
35
36         arrayAdapter = new ArrayAdapter<>(getApplicationContext(),
37             android.R.layout.simple_list_item_1);
38         for (BluetoothDevice remoteDevice : remoteDevices) {
39             arrayAdapter.add(new BluetoothDeviceWrapper(remoteDevice));
40         }
41         mListView.setAdapter(arrayAdapter);
42
43     }
44 }
```

```

43         mListView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
44             @Override
45             public void onItemClick(AdapterView<?> parent, View view,
int position, long id) {
46                 BluetoothDeviceWrapper item = (BluetoothDeviceWrapper)
parent.getItemAtPosition(position);
47                 Intent intent = new Intent(getApplicationContext(),
MainActivity.class);
48                 intent.putExtra("EXTRA_DEVICE_ADDRESS",
item.remoteDevice.getAddress());
49                 setResult(RESULT_OK, intent);
50                 finish();
51             }
52         });
53     }
54
55
56     public void onScanButtonPress(View view) {
57 //         if (!mBlueAdapter.isDiscovering()) {
58 //             mBlueAdapter.startDiscovery();
59 //         }
60     }
61 }

```

## 6.7 BLUETOOTH DEVICE WRAPPER

### *BluetoothDeviceWrapper.java*

```
1 package com.arm.controller;
2
3 import android.bluetooth.BluetoothDevice;
4
5
6 public class BluetoothDeviceWrapper {
7     public BluetoothDevice remoteDevice;
8
9     public BluetoothDeviceWrapper(BluetoothDevice remoteDevice)
10    {
11        this.remoteDevice = remoteDevice;
12    }
13
14    @Override
15    public String toString() {
16        return remoteDevice.getName() + " " +
17        remoteDevice.getAddress();
18    }
19 }
```

## 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](http://www.microchip.com)