Robotic Arm

Project Submitted for the

Award

of

Diploma

in

Electronics and Telecommunication Engineering

By

Abhishek Sanjay Chinchole	**1903012**
Akash Ananda Lohar	**1903039**
Srushti Dattatray Pophale	**1903050**
Aditi Chetan Vanikar	**1903061**

Under the Guidance of

Prof. A.D. Vikhankar

Government Polytechnic, Pune

JUNE 2022

Government Polytechnic, Pune - 16

(An Autonomous Institute of Government of Maharashtra)

CERTIFICATE

This is to Certify That

Abhishek Sanjay Chinchole	1903012
Akash Ananda Lohar	1903039
Srushti Dattatray Pophale	1903050
Aditi Chetan Vanikar	1903061

Has completed the necessary project work and prepared for the Bonafide on

ROBOT ARM

In satisfactory manner as a partial fulfilment of requirement of the

Third Year Diploma in

Electronics and Telecommunication Engineering

For Academic Year

2021 – 2022

Prof. A.D. Vikhankar	**Dr. S.P Narote**	**Dr. Vitthal S. Bandal**
Guide	**H.O.D E&TC**	**Principal**

Place

Date:

ABSTRACT

A robotics is the most lucrative and rewarding career. Being an engineering student, youare not only developing advanced robotic devices but also making the lives of people easy by making innovative solutions that enhance and help humans. Robotic hands that appear and act like human hands are constructed in a way that makes them very similar to the real thing. This simple human-like hand uses multiple motors with one long tendon roped through the fingers to close and relax the hand, and move the fingers independently.

Here we propose to build a robotic arm controlled by MIT app. Robot arms are designed to manipulate and transport parts, tools, or special manufacturing elements through reprogrammable movements to perform a variety of tasks. This is why they are commonly used in industrial environments, although there are also scale versions dedicated to STEM education and hobby robotics.

The development of this arm based on ATMEGA-328 through an Arduino uno with a personal computer for controlling this arm with an invented application. Finally this robot arm may be expected to overcome the problem such as picking or placing hazardous or non-hazardous objects from one place to another place required in many industries. A microcontroller that drives the servo motors with the capability of modifying position The programming is done on ATMEGA-328p Microcontroller using Arduino programming.

ACKNOWLEDGEMENT

This project is carried out at the department of Electronics and Telecommunications Engineering, under the supervision of Mr. A.D. Vikhankar sir. We wish to express our sincere gratitude to our Honorable HOD Mr. Narote sir and staff members for their competent guidance support throughout this project. We would like to express our deepest gratitude to the staff for their timely help, support and everlasting patience andwe would like to thank our beloved sir.

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them. We are highly indebted to Mr. A. D. Vikhankar sir for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express my gratitude towards my parents & friends for their kind co- operation and encouragement which help me in completion of this project. We would like to express my special gratitude and thanks to Hon. HOD. Dr. Narote sir for givingus such attention and time.

Abhishek Chinchole

Akash Lohar

Srushti Pophale

Aditi Vanikar

Contents

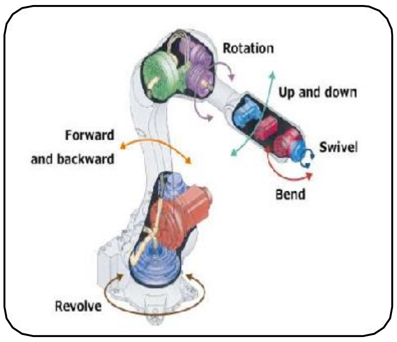
LIST OF FIGURES

LIST OF TABLES

Chapter 1

1. Introduction

Robotic Arm is a mechanical arm like structured robot that is capable of lifting an object and keeping it in desired Place within the limit of arms. As manual labour is being reduced at big scale industries and factories to increase efficiency and gain profit by installing robotsthat can do repetitive works. A simple robotic arm is one of the most commonly installed machines. We are introducing the basic concepts of an Arduino controlled robotic arm project.



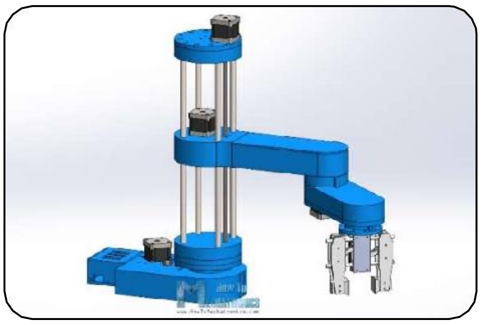


Fig 1.1(a): Movement of Robot Arm

Fig 1.1(b): Pick & Place Robot Arm

- A typical industrial robot arm includes a series of joints, articulations and manipulators that work together to closely resemble the motion and functionality of a human arm (at least from a purely mechanical perspective). A programmable robotic arm can be a complete machine in and of itself, or it can function as an individual robot part of a larger and more complex piece of equipment.
- A great many smaller robotic arms used in countless industries and workplace applications today are benchtop-mounted and controlled electronically. Larger versions might be floor-mounted, but either way they tend to be constructed from sturdy and durable metal (often steel or cast iron), and most will feature between 4-6 articulating joints.

- Again, from a mechanical perspective, the key joints on a robotic arm are designed to closely resemble the main parts of its human equivalent including the shoulder, elbow, forearm and wrist.
- Such is the speed and power that industrial robot arms can work at, there's a pressing need to be extremely safety-conscious when programming and using the .However, when deployed appropriately, they can vastly increase production rates and accuracy of placement and picking tasks, as well as performing heavy-duty
- As technology has advanced and the manufacturing costs of robotic componentshas fallen over the years, the past decade or so has seen a very rapid expansion in the availability and affordability of robots and robotic arms across a very wide rangeof industries. This means that they're far more commonly encountered in smaller- scale operations than they once were, because they're no longer only an economically viable option for large-scale production lines outputting very high volumes of product.

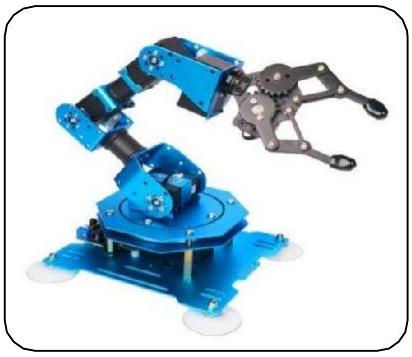


Fig 1.2(a):Side view of Robot Arm

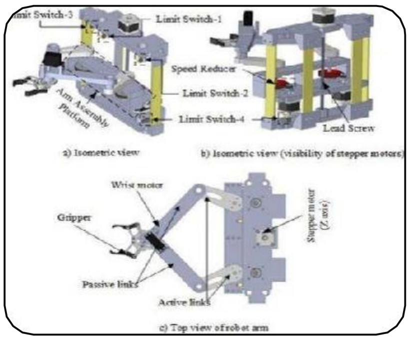


Fig 1.2(b):Top view of Robot Arm

- Robotic arms are fast, accurate and reliable, and can collectively be programmed to perform an almost infinite range of different operations.
- Robotic arms can be used for all manner of industrial production, processing and manufacturing roles - any task in which extremely precise, fast and repeatablemovements are required.
- At times humans may tend to error or get tired or may not be competent to work at certain levels and hence the use of Robotic Arm becomes Mandatory.

1.1 Project Motivation

Lucrative & rewarding career: A robotics is the most lucrative and rewarding career. Being an engineering student, you are not only developing advanced robotic devices but also making the lives of people easy by making innovative solutions that enhance and help humans. Robotic hands that appear and act like human hands are constructed in a way that makes them very similar to the real thing. This simple human-like hand uses multiple motors with one long tendon roped through the fingers to close and relax the hand, and move the fingers independently**.**

Robots are used in science and industry to replace or entertain humans and although theydo not have to look like us nor have to perform tasks in a humane manner, many of themdo. In fact, some of the most popular in the industry are robot manipulators which resemble the human arm. Robot arms are designed to manipulate and transport parts, tools, or special manufacturing elements through reprogrammable movements to perform a variety of tasks. This is why they are commonly used in industrial environments, although there are also scale versions dedicated to STEM education and hobby robotics.

1.2 Problem Definition

i. The control task is to move the robot arm from an initial position to a final position.
ii. To achieve that we require prior knowledge of either desired position or angle ofeach joint, where using the angels is called forward kinematic
iii. Arm which can be wirelessly controlled and programmed using a custom-buildAndroid application.
iv. And building own application through MIT application

1.3 Evolution

Unimate introduced the first industrial robotic arm in 1961, it has subsequently evolved into the PUMA arm. In 1963 the Rancho arm was designed; Minsky's Tentacle arm appeared in 1968, Scheinman's Stanford arm in 1969, and MIT's Silver arm in 1974. Aird became the first cyborg human with a robotic arm in 1993.

The first Unimation robotic arm was sold to the GM plant in Ewing Township, New Jersey, where it was used for stacking hot metal. From there, other automotive makers licensed to the technology to aid their production. The earliest robots as we know them were created in the early 1950s by George C. Devol, an inventor from Louisville, Kentucky. Universal Robots CTO and co-founder Esben Østergaard tells IEEE Spectrum that it took three years to develop the UR3. The biggest technical challenge, he said, was miniaturizing their technology, which was already highly compact and integrated

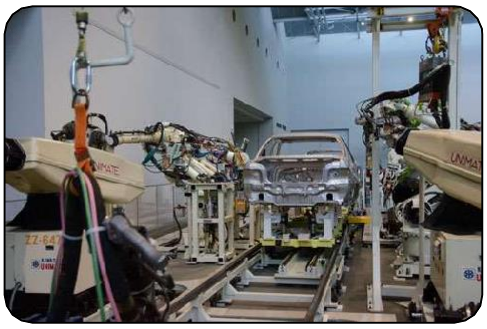


Fig 1.3(a): Industrial Robot Arm

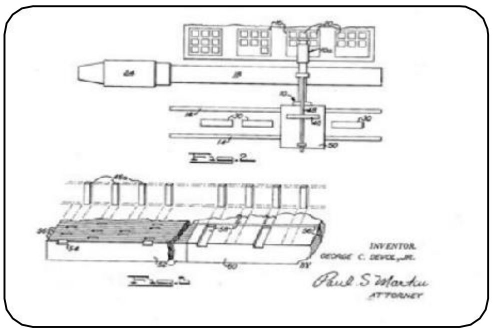


Fig 1.3(b): Internal Robot Arm Parts

Eventually, U.S. companies bought in as well, and the robots redefined the auto industryin particular—even if, like Ford, they had to take great pains to soft-pedal the concept. The slow movement by corporate America, whatever the reason, cost the American robotics market its first-mover advantage.

1.4 Famous Robotic Arms Manufactures

FANUC Robotics

FANUC Robotics offers over one hundred models of industrial robots. Renowned for their easy-to-use, versatile products, FANUC is a leader in innovation. FANUC is also well- known for their large and powerful M-2000iA series robotic arms. This “Ultra Heavy Payload” class has a working capacity of up to 2300 kg! Of course, they make arms of all sizes in between as well. Their Paint Series robotic arm uses a top-of-the-line hydraulic system that is powerful enough for automobile painting but delicate enough for small powder-coating jobs. Finally, their mid-range arms can do everything from pick-and-place to welding and machine tending. FANUC has a robot for virtually every automation needed.

Universal Robots

Robots are more popular now than ever, and they are quickly becoming a ubiquitous fixture in many factories. When you look back at robot evolution, you will find Universal Robotics at the start of it all. Universal Robots introduced robots to the industrial market. If you’re searching for a robotic arm for your factory, robots may be on your list. If they are, it’s worth taking a look at the creator of these revolutionary and now commonplace machines.

Yaskawa Electric

Almost one hundred years ago, Yasakawa Electric was producing basic induction motors and magnetic conductors. Today, their U1000 Industrial Matrix inverter drive is revolutionizing the industry. Additionally, Yasakawa offers several welding robots, robots, and a variety of material handling robotic arms. Today, Yawaska is known for its high-quality and energy-efficient designs. Their Motoman line, in particular, is one ofthe most versatile lines of robot arms available. Available for everything from the assembly line to the food industry, these arms have reaches ranging from less than a meter to well over three meters. Customers can also choose between multi-axis, robot, and delta robot

ABB

Swiss-based ABB has over 130 years of technological innovation. With a foothold in more than 100 countries, ABB has sold over 400,000 robots worldwide. Today, ABB is a leaderin industrial digitization and a forerunner in Industry 4.0. ABB specializes in single-arm and dual-arm designs that make their robots versatile, adaptable, and durable.

Kuka Robotics

When Kuka’s Quantec robotic arm came out, it went viral with commercial videos of it playing ping-pong. It’s an impressive machine that showcases unparalleled speed and precision. Since then, anyone who didn’t know the Kuka brand is now sure to be familiar with their line-up. From the small and quick Agilus line to the KR 1000 Titan (with a 1300 kg payload capacity and 3.6 m reach), Kuka produces a wide range of robotic arms. Start here, and you may not need to look any further.

Chapter 2

Literature Survey/Related work

2.1 Robotic Arm Control System Based on AI Wearable Acceleration Sensor.

With the development of science and technology, many practical production requirements for the function of the manipulator are more and more refined, especially in the high-end research field. This paper mainly introduces the research of manipulator control system based on AI wearable acceleration sensor, aiming to provide some ideas and directions for the research of wearable manipulator.

2.2 Design of a 4 DOF parallel robot arm

This paper presents a firmware design and its implementation on a real time embedded system for driving a 4DOF parallel robot arm. The firmware primarily comprised of two components to produce motion of the robot arm:

a) generation of continuous position coordinates and

b) generation of actuating signals.

c) The kinematic equations were solved with the dual-core capability of the microcontroller using real time operating system(RTOS),

2.3 How to mechatronics

- An Arduino Robot Arm which can be wirelessly controlled and programmed usinga custom-build Android application.
- Using the sliders in the app we can manually control the movement of each servo or axis of the robot arm. Also using the “Save” button we can record each position or step and then the robot arm can automatically run and repeat these steps. With the same button we can pause the automatic operation as well as reset or delete all steps so that we can record new ones.

2.4 Survey of Robotic Arm and Parameters

- This is a working field of research in which there are a number of outstanding open problems and an area of exploration. Nowadays, a different variety of robotic arms are commercially available. Some of them are excellent in accuracyand repeatability.
- The paper concludes with research gaps and proposed work. Robotic arm uses inthe different fields like a household, workplace, and working station. Keywords- Robotic arm, axis, degree of freedom, working envelope and space, kinematics, payload, speed and acceleration, accuracy and repeatability.
- In today's world there is an increasing need to create artificial arms for different inhuman situations where human interaction is difficult or impossible.

Chapter 3
System Development

3.1 Hardware Requirements

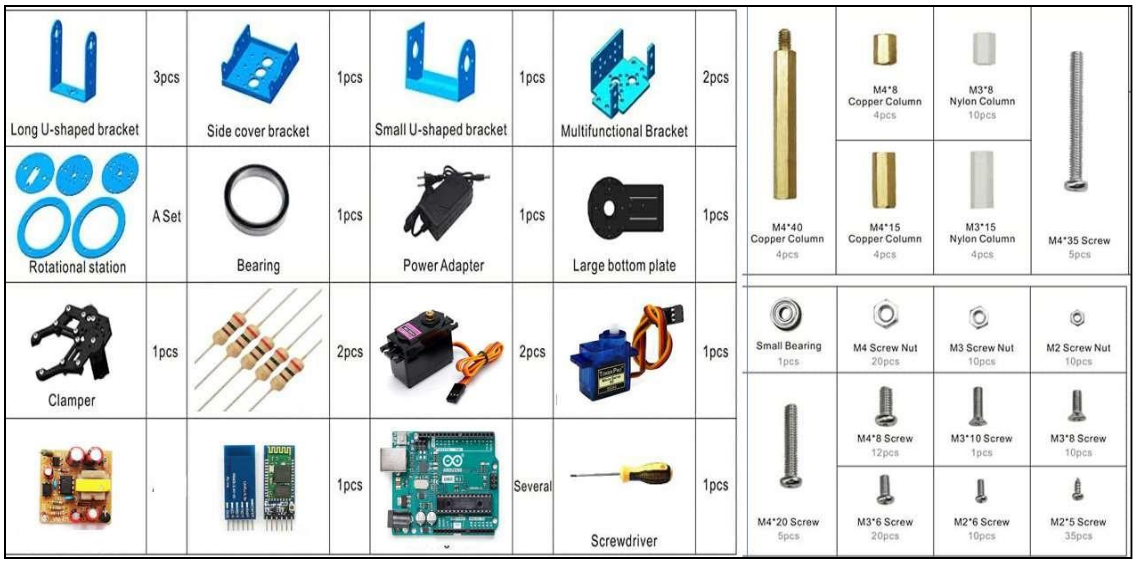
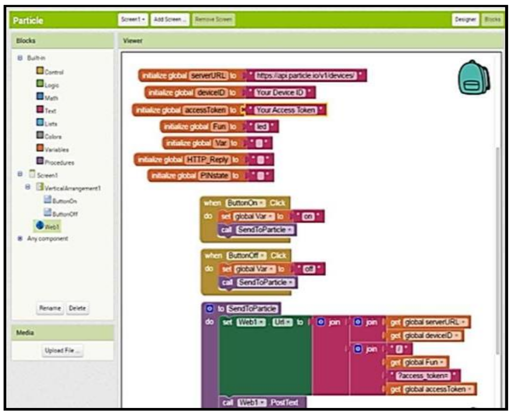


Fig 3.1: Hardware Requirements

- Arduino UNO
- HC-06 Bluetooth Module
- MG996 Servo Motor
- SG90 Servo Motor
- Power Adapter
- 1k resistors
- Arm Mechanical Parts
- Jumper Cables

3.2 Software Requirements

- Arduino IDE
- MIT APP Inventor



```
initialize global serverURL to "https://api.particle.io/v1/devices/"
initialize global deviceID to "Your Device ID"
initialize global accessToken to "Your Access Token"
initialize global Fun to "led"
initialize global Var to ""
initialize global HTTP_Reply to ""
initialize global PINstate to ""

when ButtonOn.Click
do  set global Var to "on"
    call SendToParticle

when ButtonOff.Click
do  set global Var to "off"
    call SendToParticle

to SendToParticle
do  set Web1.Url to join  join  get global serverURL
                                get global deviceID
                          join  "/"
                                get global Fun
                                "?access_token="
                                get global accessToken
    call Web1.PostText
```

AN OPEN PROJECT WRITTEN, DEBUGGED, AND SUPPORTED BY ARDUINO.CC AND THE ARDUINO COMMUNITY WORLDWIDE

LEARN MORE ABOUT THE CONTRIBUTORS OF ARDUINO.CC on arduino.cc/credits

Fig 3.2: Software Requirements

3.3 Components Specification

3.3.1 Arduino UNO

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects. Arduino senses the environment by receiving inputs from many sensors, and affects its surroundings by controlling lights, motors, and other actuators. You can tell your Arduino what to do by writing code in the Arduino programming language and using the Arduino development environment.

The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software. The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases. The ATmega328 on the board comes preprogramed with a bootloader that rammer.

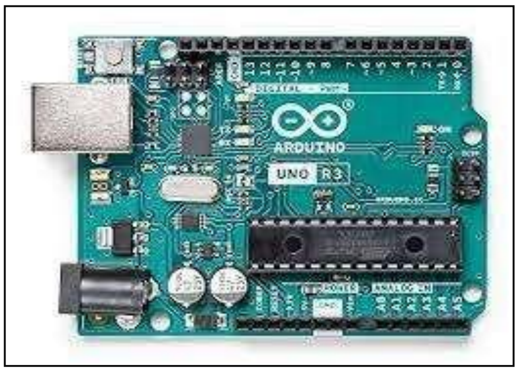


Fig 3.3:Arduino Uno

3.3.2 Arduino Uno Specification

- Microcontroller: ATmega328P
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limit): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- PWM Digital I/O Pins: 6
- Analog Input Pins: 6
- DC Current per I/O Pin: 20 mA
- DC current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB (ATmega328P) of which 0.5 KB used by bootloader
- SRAM: 2 KB (ATmega328P)
- EEPROM: 1 KB (ATmega328P)
- Clock Speed: 16 MHz
- LED_BUILTIN: 13
- Length: 68.6 mm

3.3.3 General pin functions

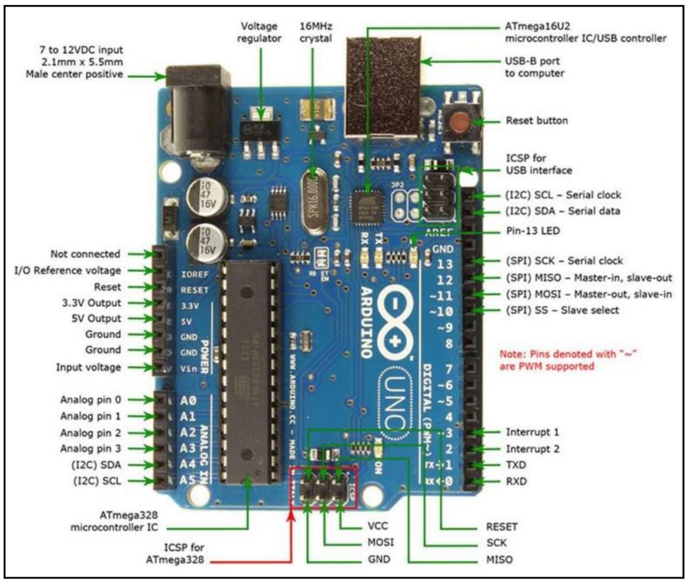


Fig 3.4:Arduino Pin Diagram

LED: There is a built-in LED driven by digital pin 13. When the pin is high value, the LED is on, when the pin is low, it is off.

VIN: The input voltage to the Arduino board when it is using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supplyvoltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin ofthe board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board.

3V3: A 3.3-volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND: Ground pins.

IOREF: This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and selectthe appropriate power source, or enable voltage translators on the outputs to work with the 5V or 3.3V.

Reset: Typically used to add a reset button to shields that block the one on the board.

3.3.4 Special pin functions

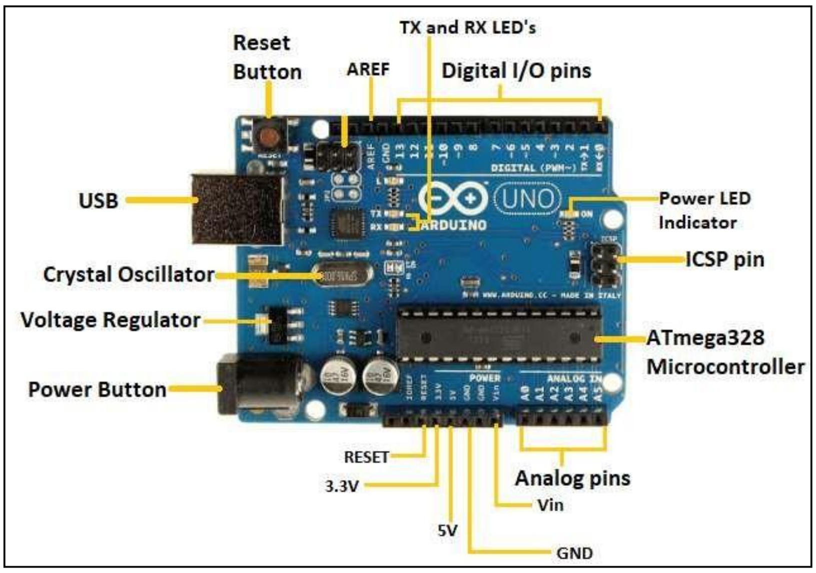


Fig 3.5:Arduino Special pin features

Each of the 14 digital pins and 6 analog pins on the Uno can be used as an input or output, under software control (using pinMode(), digitalWrite(), and digitalRead() functions). They operate at 5volts. Each pin can provide or receive 20 mA as the recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50K ohm. A maximum of 40mA must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller. The Uno has 6 analog inputs, labelled A0 through A5; each provides 10 bits of resolution (i.e., 1024 different values). By default, they measure from ground to 5 volts, though it is possible to change the upperend of the range using the AREF pin and the analogReference() function.

In addition, some pins have specialized functions:

i. **Serial** / UART: pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL serial chip.

ii. **External interrupts**: pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.

iii. **PWM** (pulse-width modulation): pins 3, 5, 6, 9, 10, and 11. Can provide 8-bit PWM output with the analogWrite() function.

iv. **SPI** (Serial Peripheral Interface): pins 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK). These pins support SPI communication using the SPI library.

v. **TWI** (two-wire interface) / I²C: pin SDA (A4) and pin SCL (A5). Support TWI communication using the Wire library.

vi. **AREF** (analog reference): Reference voltage for the analog inputs

3.3.5 Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. Arduino Software (IDE) includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A Software Serial library allows serial communication on any of the Uno's digital pins

3.3.6 MG996R Servo Motor

A **servomotor** (or **servo motor**) is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors. A servomotor is a closed- loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft.

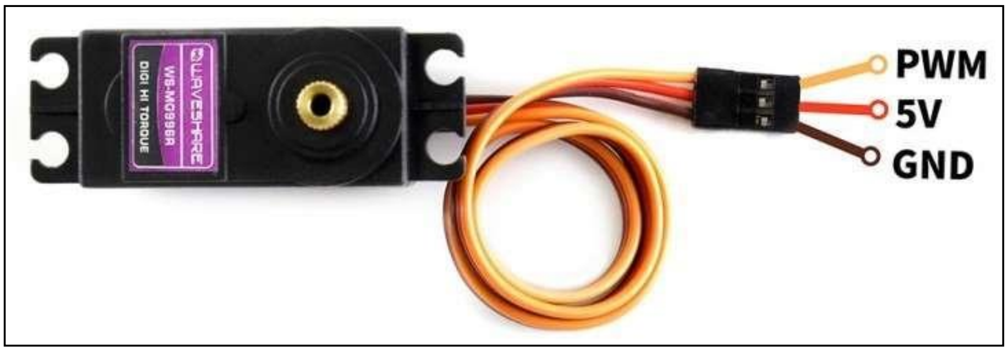


Fig 3.6: MG996R Servo Motor

Servomotors are generally used as a high-performance alternative to the stepper motor. Stepper motors have some inherent ability to control position, as they have built-in output steps. This often allows them to be used as an open-loop position control, without any feedback encoder, as their drive signal specifies the number of steps of movement to rotate, but for this the controller needs to 'know' the position of the stepper motor on power up. Therefore, on first power up, the controller will have to activate the stepper motor and turn it to a known position, A servomotor will immediately turn to whatever angle the controller instructs it to, regardless of the initial position at power up.

3.3.7 Specifications

Table 3.1: MG996 Servo Motor Specifications

Operating Voltage (VDC)	4.8 ~ 6.6
Temperature Range	0- 55℃
Stall Torque 4.8V (Kg-Cm)	9.40
Stall Torque 6.6V (Kg-Cm)	11
Operating Speed 4.8V	0.19sec/60°
Operating Speed 6.6V	0.15sec/60°
Dead Bandwidth	1uS
Gear Type	Metal
No. of teeth	25
Length (mm)	40.7
Width (mm)	19.7
Height (mm)	42.9
Weight (gm)	55
Shipment Weight	0.059 kg
Shipment Dimensions	5 × 3 × 5 cm

Features:

1. Gear Type: Metal gear
2. Servo Plug: JR (Fits JR and Futaba)
3. Servo arms & screws included, and fit with Futaba servo arm
4. It’s universal “S” type connector that fits most receivers, including Futaba, JR, Hitec

   ,GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum.
5. CE & RoHS approved

3.3.8 SG90 Servo Motor

There are lots of servo motors available in the market and each one has its own specialty and applications. The following two paragraphs will help you identify the right type of servo motor foryour project/system.

Most of the hobby Servo motors operates from 4.8V to 6.5V, the higher the voltage higher the torque we can achieve, but most commonly they are operated at +5V. Almost all hobby servo motors can rotate only from 0° to 180° due to their gear arrangement so make sure you project canlive with the half circle if no, you can prefer for a 0° to 360° motor or modify the motor to make afull circle. The gears in the motors are easily subjected to wear and tear, so if your application requires stronger and long running motors you can go with metal gears or just stick with normal plastic gear.

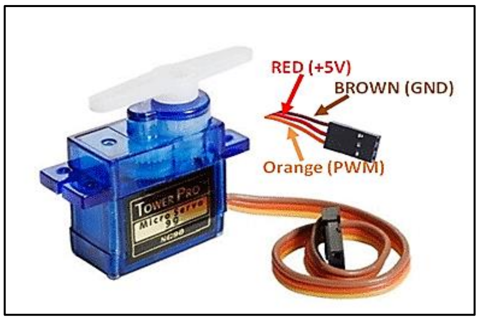


Fig 3.7: (a) SG90 Servo Motor

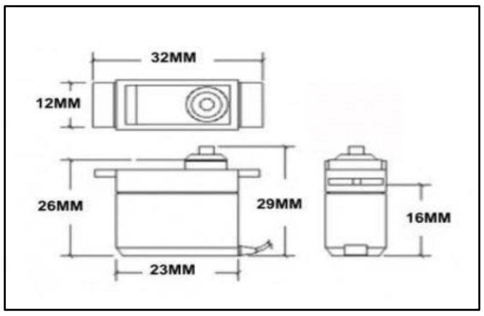


Fig 3.7: (b) SG90 Dimensions

Table 3.2: SG90 Servo Motor Pin Description

Wire Number	**Wire Color**	**Description**
1	Brown	Ground wire connected to the ground of system
2	Red	Powers the motor typically +5V is used
3	Orange	PWM signal is given in through this wire to drive the motor

3.3.9 SG-90 Features

I. Operating Voltage is +5V typically

II. Torque: 2.5kg/cm

III. Operating speed is 0.1s/60°

IV. Gear Type: Plastic

V. Rotation : 0°-180°

VI. Weight of motor : 9gm

VII. Package includes gear horns and screws

3.3.10 HC-06 Bluetooth Module

HM-06 is a Bluetooth module designed for establishing short range wireless data communication between two microcontrollers or systems. The module works on Bluetooth 2.0 communication protocol and it can only act as a slave device. This is cheapest method for wireless data transmission and more flexible compared to other methods and it even can transmit files at speed up to 2.1Mb/s.

HC-06 uses frequency hopping spread spectrum technique (FHSS) to avoid interference with other devices and to have full duplex transmission. The device works on the frequency range from 2.402 GHz to 2.480GHz.

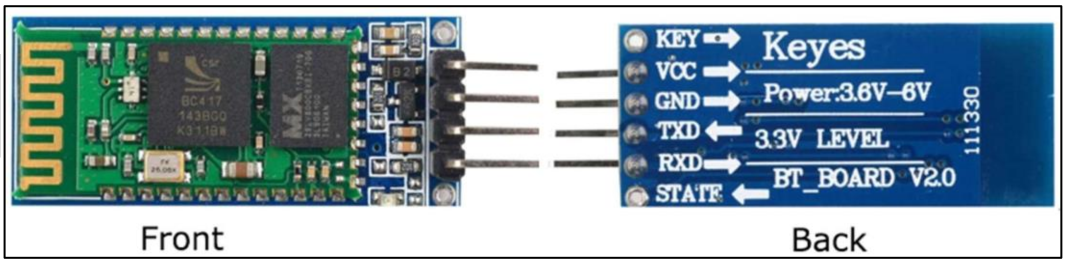


Fig 3.8 :HC06 Bluetooth module

Table 3.3: HC06 Bluetooth module Pin Description

Pin	**Name**	**Function**
1	Key	The pin state determines whether the module works in AT command mode or normal mode [High=AT commands receiving mode(Commands response mode), Low or NC= Bluetooth module normally working]
2	Vcc	+5V Positive supply needs to be given to this pin for powering the module
3	Gnd	Connect to ground
4	TXD	Serial data is transmitted by module through this pin (at 9600bps by default), 3.3V logic
5	RXD	Serial data is received by module through this pin (at 9600bps by default),3.3V logic
6	State	The pin is connected to the LED on the board to represent the state of the module

3.3.11 HC-06 Features and Electrical characteristics

i. Bluetooth protocol: Bluetooth V2.0 protocol standard

ii. Band: 2.40GHz—2.48GHz, ISM Band

iii. Receiver sensitivity: -85dBm

iv. USB protocol: USB v1.1/2.0

v. Safety feature: Authentication and encryption

vi. Operating voltage range:+3.3V to +6V

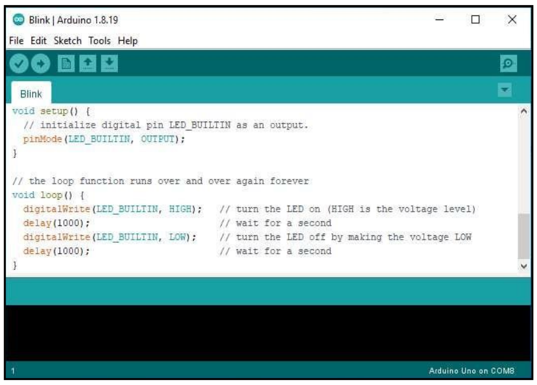
vii. Operating Current: 40Ma

3.4 Software Description

3.4.1 Arduino IDE

Arduino IDE is an open-source software, designed by Arduino.cc and mainly used for writing, compiling & uploading code to almost all Arduino Modules. It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process. The main code,also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board. The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module. This environment supports both C and C++ languages.

Fig 3.9(a): Arduino IDE Overview



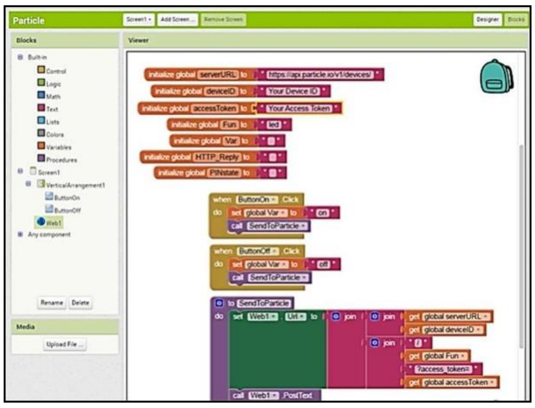
```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Fig 3.9(b): Arduino IDE Sketch

3.4.2 MIT APP Inventor

MIT App Inventor is a web application integrated development environment originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT). It allows newcomers to computer programming to create application software(apps) for two operating systems (OS): Android, and iOS, which, as of 8 July 2019, is in final beta testing. It is free and open-source software released under dual licensing: a Creative Commons Attribution Share Alike 3.0 Unported license, and an Apache License 2.0 for the source code. It uses a graphical user interface (GUI) very similar to the programming

languages Scratch (programming language) and the Star Logo, which allows users to dragand drop visual objects to create an application that can run on Android devices, while an App-Inventor Companion (The program that allows the app to run and debug on) that works on iOS running devices are still under development. In creating App Inventor, Google drew upon significant prior research in educational computing, and work done within Google on online development environments.



```
initialize global serverURL to "https://api.particle.io/v1/devices/"
initialize global deviceID to "Your Device ID"
initialize global accessToken to "Your Access Token"
initialize global Fun to "led"
initialize global Var to " "
initialize global HTTP_Reply to " "
initialize global PINstate to " "

when ButtonOn .Click
do  set global Var to "on"
    call SendToParticle

when ButtonOff .Click
do  set global Var to "off"
    call SendToParticle

to SendToParticle
do  set Web1 . Url to  join  join  get global serverURL
                                   get global deviceID
                             join  "/"
                                   get global Fun
                                   "?access_token="
                                   get global accessToken
    call Web1 .PostText
```

Fig 3.10(a): MIT App inventor **Fig 3.10(b): MIT App Overview**

3.5 System Architecture & Mechanical Design

i. The arm is the main section of the robotic arm and consists of three parts: the shoulder, elbow and wrist. These are all joints, with the shoulder resting at the base of the arm, typically connected to the controller, and it can move forward, backwardor spin.

ii. A typical robotic arm is made up of seven metal segments, joined by six joints. The computer controls the robot by rotating individual step motors connected to each joint (some larger arms use hydraulics or pneumatics). The robot uses motionsensors to make sure it moves just the right amount.

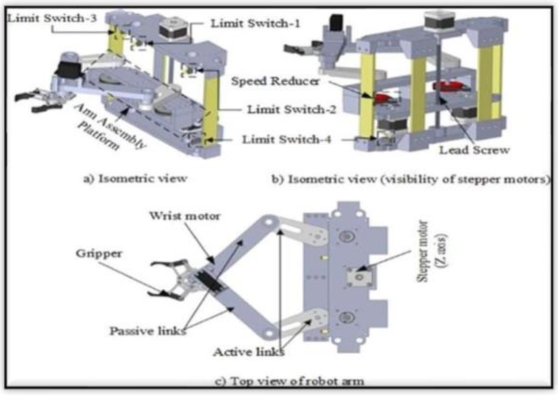


Fig 3.11(a): Isometric View of Robot Arm Fig 3.11(b): Right view of Robot Arm

The Robot Arm will have 5 outputs which consist of the base, grip, wrist, elbow, shoulder and waist. The robot has a combination of a round and rectangular base capable of housing the Arduino UNO microcontroller and the body of the robot arm. Chipboard 20 pt was used as the main material for the constructionof the robot arm because it is easy to be formed, cheap and can bear the motor weight and movement. The robot degree- of- freedom mechanism is powered by two different type ofservo motors. The robot gripper is also made of plastic filament.

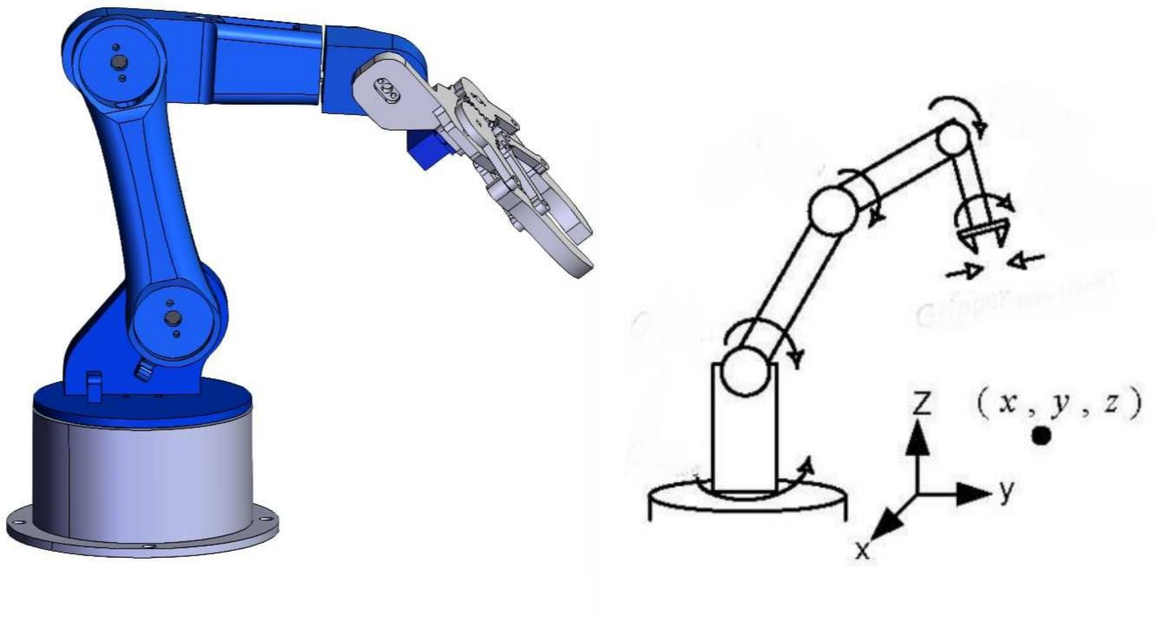


Fig 3.12: Mechanical design of robot arm

The robot arm moves in 4 axes with 6 servo motors at a specified position according to the mobile app specifications. The robot is designed to have a stationary base and movable arm body with servo motors attached to move the arm. Each servo motor (6)
is assigned as joints of the robot arm which then corresponds, each of it, on the mobile application namely for the; (s1) waist, (s2) shoulder, (s3) elbow, (s4) wrist roll, (s5) wristpitch and (s6) grip.
A software used for planning, visual ideation, and feasibility assessment, in finalizing the design. Using 6 servo motors, the robotic arm could move in different directions and could hold or release things with its gripper. For the optimal control of the robotic arm, the microcontroller Arduino UNO is used. The program code was written in c+ + language which is one of the most popular and fundamental programming languages. We also created a mobile application that will be connected to the microcontroller Arduino UNO via Bluetooth module.

3.6 Block Diagram

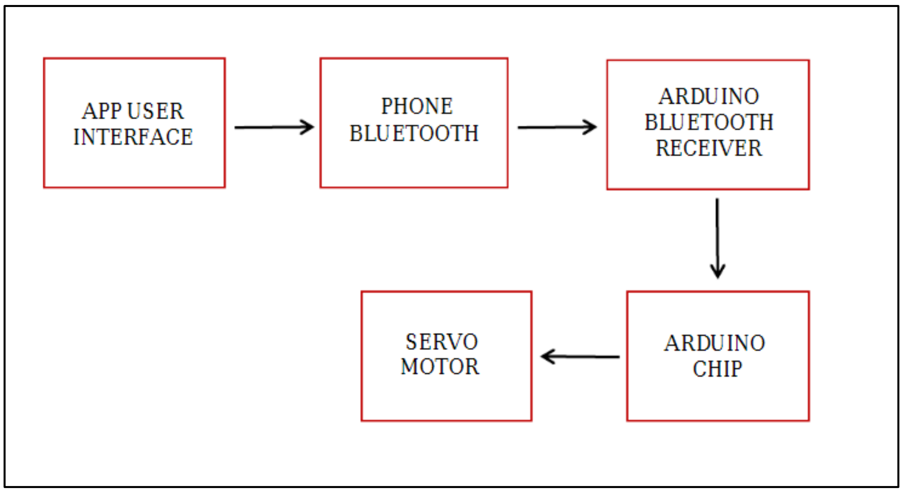


Fig 3.13: Block diagram of Robot Arm

Block diagram consists of 6 motors.

i. The three motors are MG996 and other three motors are MG90.
ii. A Bluetooth module through which the MIT app will control arm.
iii. Arduino uno is the CPU of the Robotic Arm
iv. External power source

3.7 Interfacing Diagram

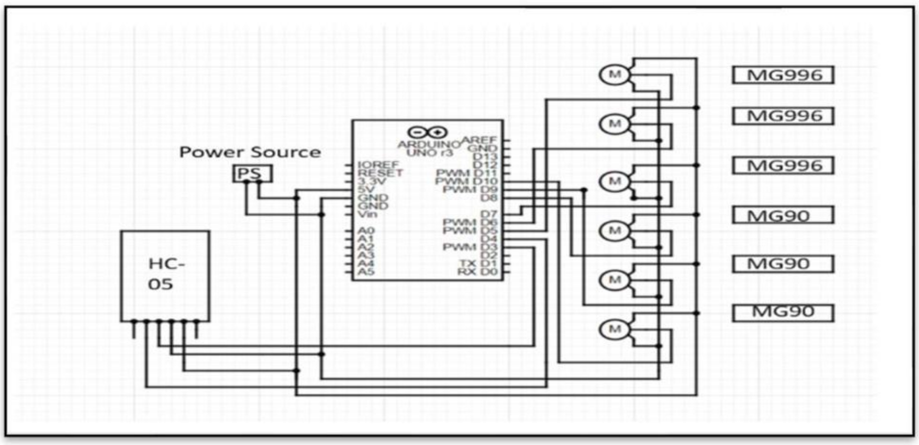


Fig 3.14:Interfacing Diagram

3.8 Circuit Diagram

i. Six servo motors are connected to the digital pins, D5, D6, D7, D8, D9, D10 of Arduino UNO

ii. Power source is connected to 5V(+ve) and to the ground

iii. HC-06 is connected to the Arduino for the communication with phone through MIT appbuilde

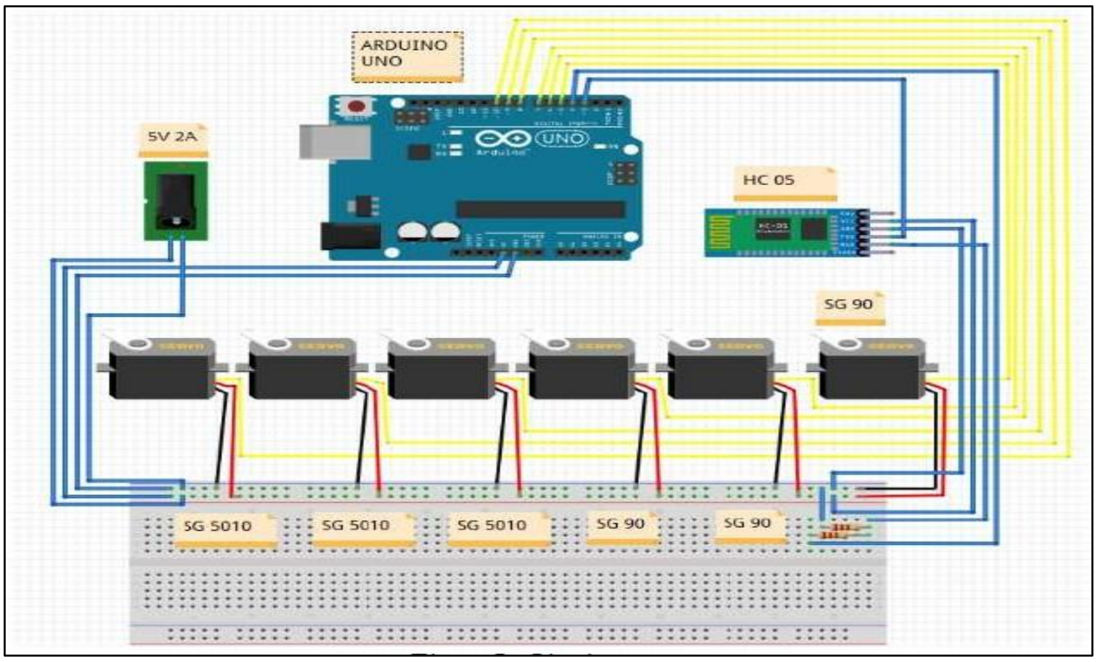


Fig 3.15: Circuit Diagram

i. Arduino UNO is the CPU for the Robotic Arm

ii. Bluetooth module HC-06 module is used for communication purpose with smartphone

iii. The control pins of six servo motors are connected to the six digital pins of ArduinoUNO.

iv. For powering the servos, we need 5V, but this must come from an external power source because the Arduino is not able to handle the amount of current that all of them can draw. The power source must be able to handle at least 2A of current. So once we have connected everything together we can move on to programming the Arduino and make the Android app.

3.9 Timeline

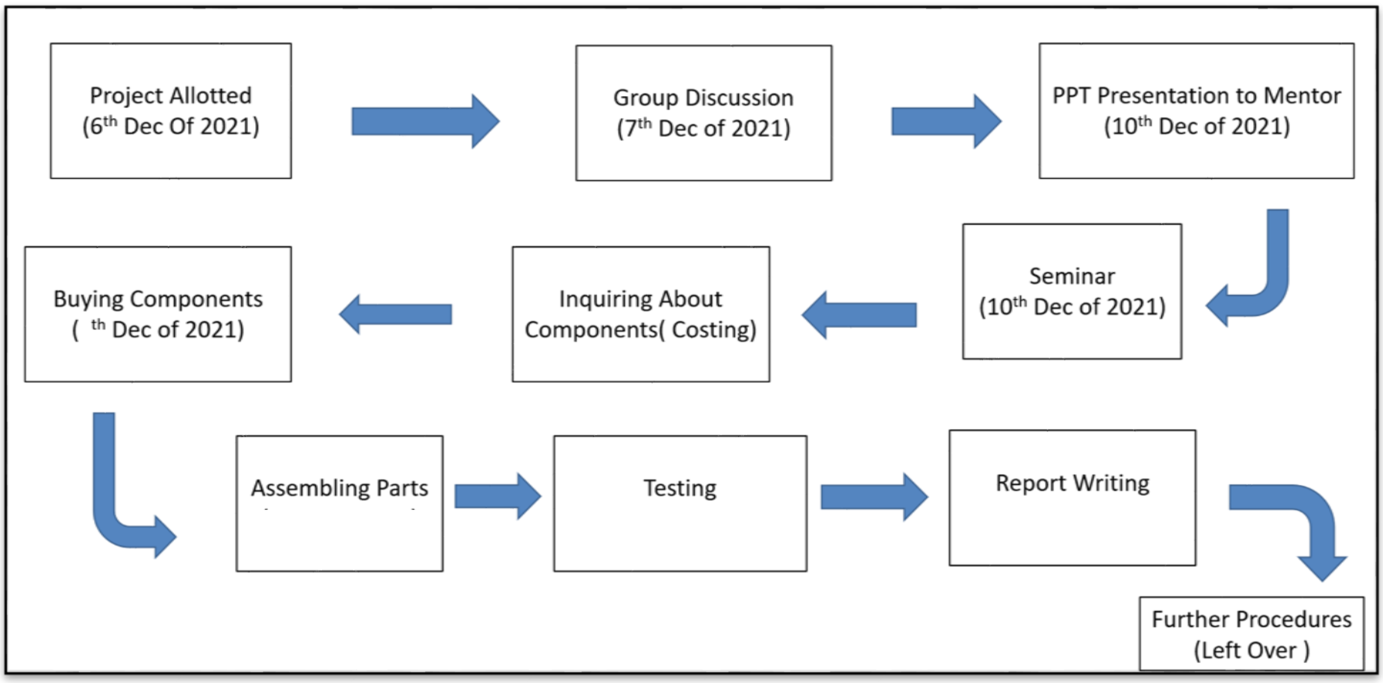

Project Allotted (6th Dec Of 2021)
Group Discussion (7th Dec of 2021)
PPT Presentation to Mentor (10th Dec of 2021)
Seminar (10th Dec of 2021)
Inquiring About Components(Costing)
Buying Components (th Dec of 2021)
Assembling Parts
Testing
Report Writing
Further Procedures (Left Over)

Fig 3.16:Timeline

Chapter 4

4.1Algorithms Development

4.1.1 Arduino robot ARM programming logic

Step 1- The Logic of this code is fairly simple the values of potentiometers are stored in an array the records are then traversed using a for loop and the servos do the steps asper the values.

Step 2- we need to define the six servos, the HC-05 Bluetooth module and some variables for storing the current and previous position of the servos, as well as arrays for storing the positions or the steps for the automatic mode.

Step 3- Initialize the servos and the Bluetooth module and move the robot arm to its initial position. We do that using the write() function which simply moves the servo to any position from 0 to 180 degrees.

Step 4- Sing the Bluetooth. Available() function , we constantly check whether there is any incoming data from the Smartphone. If true, using the readString() function we read the data as string an store it into the dataIn variable. Depending on the arrived data we will tell robot arm what to do.

Step 5- MIT App Inventor online application and here's how it works. At the top we have two buttons for connecting the smartphone to the HC-05 Bluetooth module. Then on the left side we have an image of the robot arm, and on the right side we have the six sliders for controlling the servos and one slider for the speed control.

Step 6- Each slider has different initial, minimum and maximum value that suits the robot arm joints. At the bottom of the app, we have three buttons, SAVE, RUN and RESET through which we can program the robot arm to run automatically. There is also a label below which shows the number of steps we have saved.

4.2 Flowchart

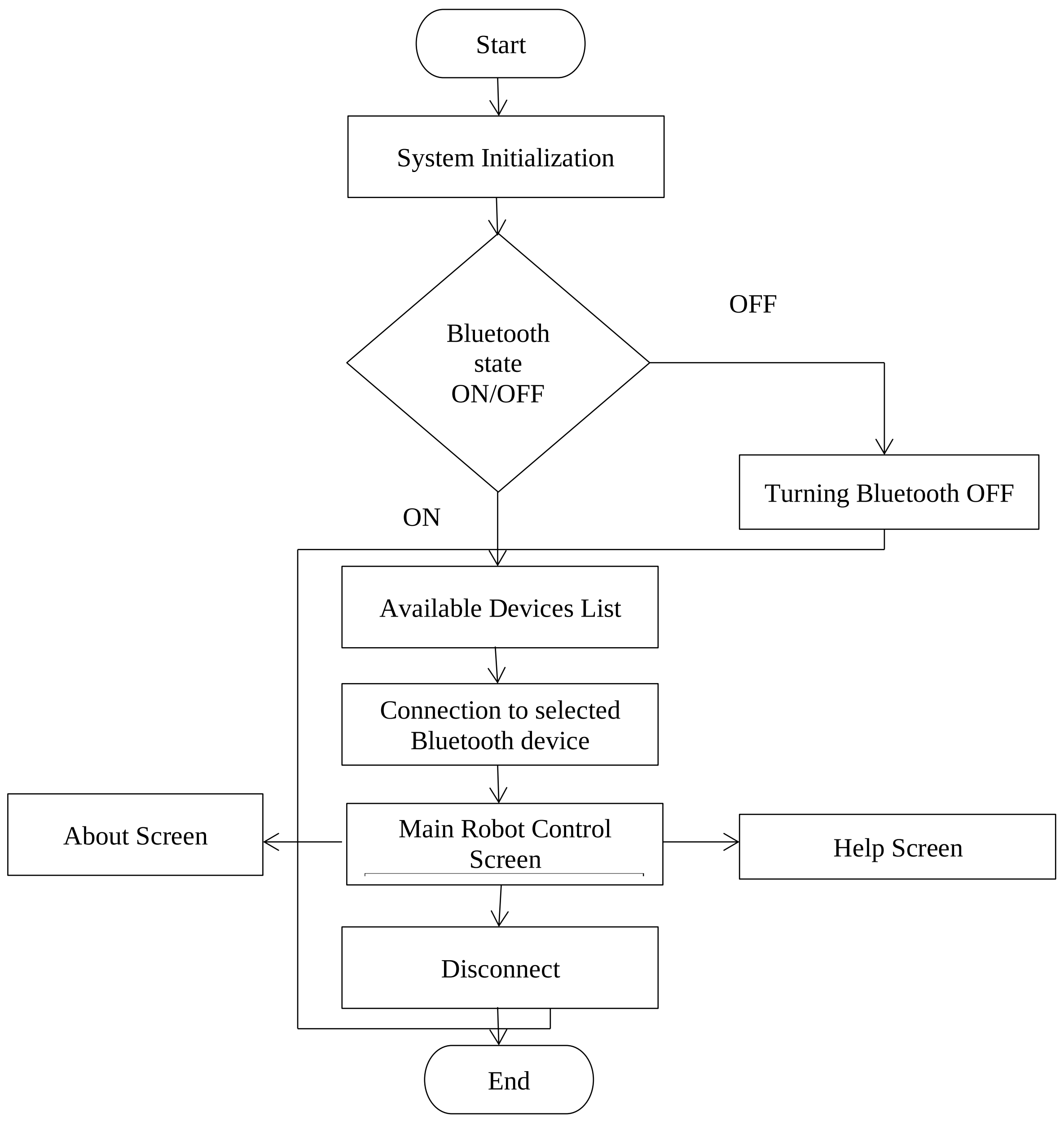


Fig 4.1: Flowchart

4.3 Arm Coding

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

Servo
servo01;
Servo
servo02;
Servo
servo03;
Servo
servo04;
Servo
servo05;
Servo
servo06;

SoftwareSerial Bluetooth(3, 4); // Arduino(RX, TX) - HC-05 Bluetooth (TX, RX)

int servo1Pos, servo2Pos, servo3Pos, servo4Pos, servo5Pos, servo6Pos; // current
positionint servo1PPos, servo2PPos, servo3PPos, servo4PPos, servo5PPos, servo6PPos;
// previous position
int servo01SP[50], servo02SP[50], servo03SP[50], servo04SP[50], servo05SP[50],
servo06SP[50]; // for storing positions/steps
int speedDelay =
20;int index = 0;
String dataIn = "";

void setup() {
  servo01.attach(5);
  servo02.attach(6);
  servo03.attach(7);
  servo04.attach(8);
  servo05.attach(9);
  servo06.attach(10);
  Bluetooth.begin(38400); // Default baud rate of the Bluetooth
  moduleBluetooth.setTimeout(1);
  delay(20);
  // Robot arm initial position
  servo1PPos = 90;
  servo01.write(servo1PPos);
  servo2PPos = 150;
  servo02.write(servo2PPos);
  servo3PPos = 35;
  servo03.write(servo3PPos);
  servo4PPos = 140;
  servo04.write(servo4PPos);
  servo5PPos = 85;
  servo05.write(servo5PPos); servo6PPos = 80; servo06.write(servo6PPos);
}


void loop() {
  // Check for incoming data
  if (Bluetooth.available() > 0) {
    dataIn = Bluetooth.readString(); // Read the data as string

    // If "Waist" slider has changed value - Move Servo 1 to
    positionif (dataIn.startsWith("s1")) {
      String dataInS = dataIn.substring(2, dataIn.length()); // Extract only the number.
E.g.from "s1120" to "120"
      servo1Pos = dataInS.toInt(); // Convert the string into integer
      // We use for loops so we can control the speed of the servo
      // If previous position is bigger then current
  positionif (servo1PPos > servo1Pos) {
    for ( int j = servo1PPos; j >= servo1Pos; j--) { // Run servo down
      servo01.write(j);
      delay(20);    // defines the speed at which the servo rotates
    }
  }
  // If previous position is smaller then current position
  if (servo1PPos < servo1Pos) {
    for ( int j = servo1PPos; j <= servo1Pos; j++) { // Run servo up
      servo01.write(j);
      delay(20);
    }
  }
  servo1PPos = servo1Pos;  // set current position as previous position
}
// Move Servo 2
if (dataIn.startsWith("s2")) {
  String        dataInS       =       dataIn.substring(2,
  dataIn.length());servo2Pos = dataInS.toInt();

  if (servo2PPos > servo2Pos) {
    for ( int j = servo2PPos; j >= servo2Pos; j-
      -) {servo02.write(j);
      delay(50);
    }
  }
  if (servo2PPos < servo2Pos) {
    for ( int j = servo2PPos; j <= servo2Pos;
      j++) {servo02.write(j);
      delay(50);
    }
  }
  servo2PPos = servo2Pos;
}
```

```
// Move Servo 3
if (dataIn.startsWith("s3")) {
 String       dataInS       =       dataIn.substring(2,
 dataIn.length());servo3Pos = dataInS.toInt();
 if (servo3PPos > servo3Pos) {
  for ( int j = servo3PPos; j >= servo3Pos; j-
   -) {servo03.write(j);
  delay(30);
  }
 }
  if (servo3PPos < servo3Pos) {
  for  (  int  j  =  servo3PPos;  j  <=  servo3Pos;
   j++) {servo03.write(j);
   delay(30);
  }
 }
 servo3PPos = servo3Pos;
}
// Move Servo 4
if (dataIn.startsWith("s4")) {
 String       dataInS       =       dataIn.substring(2,
 dataIn.length());servo4Pos = dataInS.toInt();
 if (servo4PPos > servo4Pos) {
  for ( int j = servo4PPos; j >= servo4Pos; j-
   -) {servo04.write(j);
   delay(30);
  }
 }
 if (servo4PPos < servo4Pos) {
  for  (  int  j  =  servo4PPos;  j  <=  servo4Pos;
   j++) {servo04.write(j);
   delay(30);
```

```
    }
  }
  servo4PPos = servo4Pos;
}
// Move Servo 5
if (dataIn.startsWith("s5")) {
  String       dataInS       =       dataIn.substring(2,
  dataIn.length());servo5Pos = dataInS.toInt();
  if (servo5PPos > servo5Pos) {
  for ( int j = servo5PPos; j >= servo5Pos; j--) {servo05.write(j);
      delay(30);
    }
  }
  if (servo5PPos < servo5Pos) {
    for  (  int  j  =  servo5PPos;  j  <=  servo5Pos;
      j++) {servo05.write(j);
      delay(30);
    }
  }
  servo5PPos = servo5Pos;
}
// Move Servo 6
if (dataIn.startsWith("s6")) {
  String       dataInS       =       dataIn.substring(2,
  dataIn.length());servo6Pos = dataInS.toInt();
  if (servo6PPos > servo6Pos) {
    for ( int j = servo6PPos; j >= servo6Pos; j-
      -) {servo06.write(j);
      delay(30);
    }
  }
  if (servo6PPos < servo6Pos) {
    for  (  int  j  =  servo6PPos;  j  <=  servo6Pos;
      j++) {servo06.write(j);
```

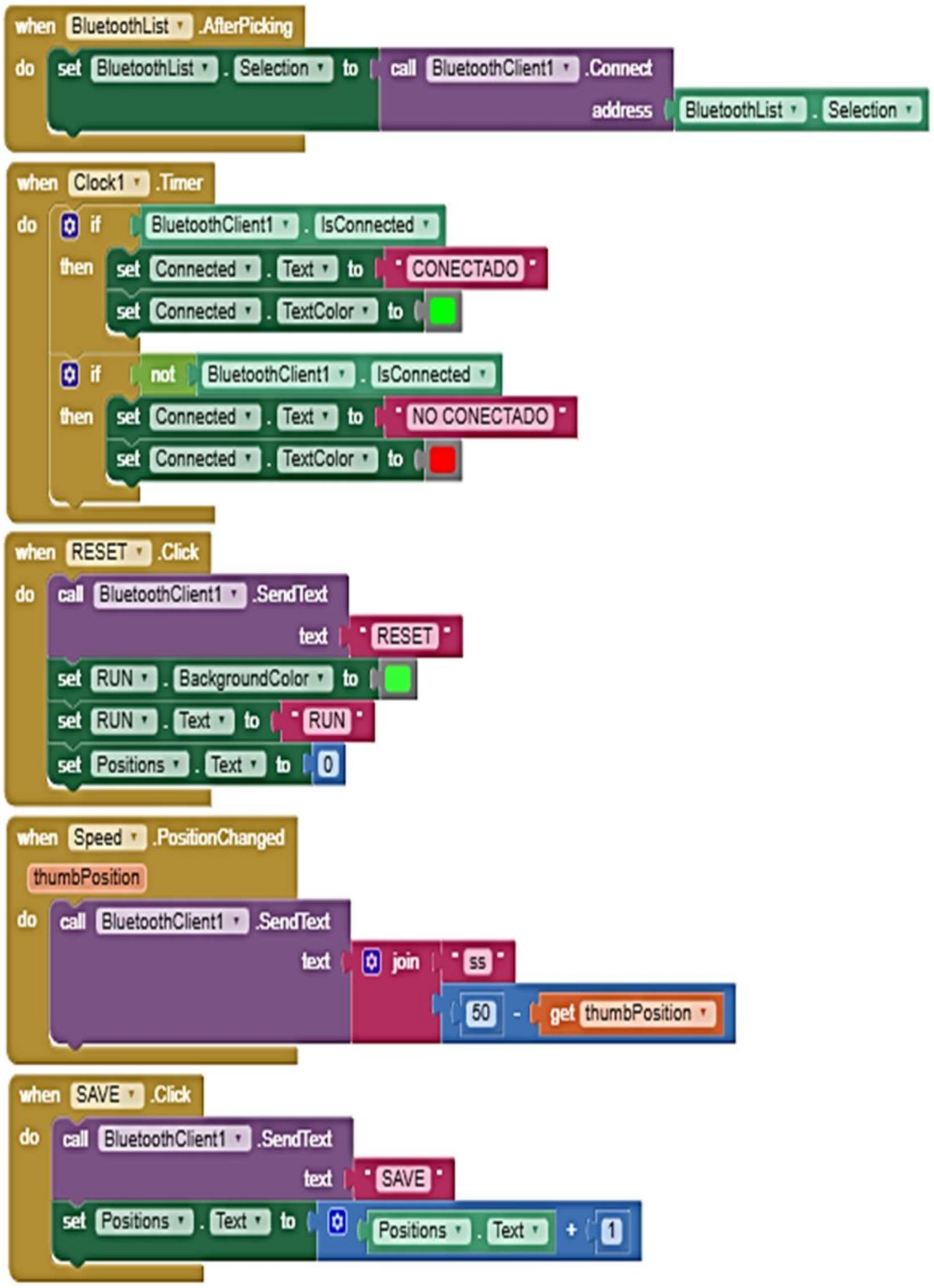
```
        delay(30);
       }
      }
      servo6PPos = servo6Pos;
    }
    // If button "SAVE" is pressed
    if (dataIn.startsWith("SAVE")) {
      servo01SP[index] = servo1PPos; // save position into the array
      servo02SP[index] = servo2PPos;
      servo03SP[index] = servo3PPos;
      servo04SP[index] = servo4PPos;
      servo05SP[index] = servo5PPos;
      servo06SP[index] = servo6PPos;
      index++;                    // Increase the array index
    }
    // If button "RUN" is pressed
    if (dataIn.startsWith("RUN")) {
      runservo(); // Automatic mode - run the saved steps
    }
    // If button "RESET" is
    pressed if ( dataIn ==
    "RESET") {
      memset(servo01SP, 0, sizeof(servo01SP)); // Clear the array data to 0
      memset(servo02SP, 0, sizeof(servo02SP));
      memset(servo03SP, 0, sizeof(servo03SP));
      memset(servo04SP, 0, sizeof(servo04SP));
      memset(servo05SP, 0, sizeof(servo05SP));
      memset(servo06SP,               0,
      sizeof(servo06SP));index = 0; // Index
      to 0
    }
  }
}
```

```
// Automatic mode custom function - run the saved
stepsvoid runservo() {
  while (dataIn != "RESET") { // Run the steps over and over again until "RESET"
button is pressed
    for (int i = 0; i <= index - 2; i++) { // Run through all
      steps(index)if (Bluetooth.available() > 0) {       // Check for
      incomding data dataIn = Bluetooth.readString();
        if ( dataIn == "PAUSE") {          // If button "PAUSE" is pressed
          while (dataIn != "RUN") {        // Wait until "RUN" is pressed
          againif (Bluetooth.available() > 0) {
        dataIn = Bluetooth.readString();if ( dataIn == "RESET") { break;
              }
            }
            }
        }
        // If speed slider is
        changed                if
        (dataIn.startsWith("ss"))
        {
          String dataInS = dataIn.substring(2, dataIn.length());
          speedDelay = dataInS.toInt(); // Change servo speed (delay time)
        }
      }
      // Servo 1
      if (servo01SP[i] == servo01SP[i + 1]) {
      }
      if (servo01SP[i] > servo01SP[i + 1]) {
        for ( int j = servo01SP[i]; j >= servo01SP[i + 1]; j-
          -) {servo01.write(j);
          delay(speedDelay);
        }
      }
      if (servo01SP[i] < servo01SP[i + 1]) {
        for ( int j = servo01SP[i]; j <= servo01SP[i + 1]; j++)
          {servo01.write(j);
          delay(speedDelay);
        }
      }
// Servo 2
      if (servo02SP[i] == servo02SP[i + 1]) {
      }
      if (servo02SP[i] > servo02SP[i + 1]) {
        for ( int j = servo02SP[i]; j >= servo02SP[i + 1]; j-
          -) {servo02.write(j);

        delay(speedDelay);

        }
      }
      if (servo02SP[i] < servo02SP[i + 1]) {
        for ( int j = servo02SP[i]; j <= servo02SP[i + 1]; j++)
          {servo02.write(j);
          delay(speedDelay);
        }
      }


      // Servo 3
      if (servo03SP[i] == servo03SP[i + 1]) {
      }
      if (servo03SP[i] > servo03SP[i + 1]) {
        for ( int j = servo03SP[i]; j >= servo03SP[i + 1]; j-
          -) {servo03.write(j);
          delay(speedDelay);
        }
      }
      if (servo03SP[i] < servo03SP[i + 1]) {
```

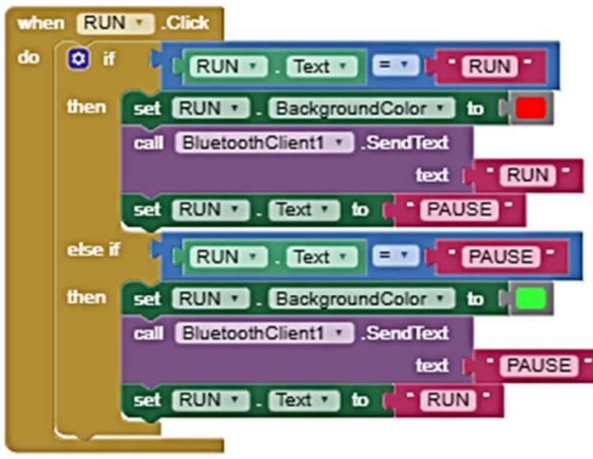
```
    for ( int j = servo03SP[i]; j <= servo03SP[i + 1]; j++)
      {servo03.write(j);
      delay(speedDelay);
    }
  }

  // Servo 4
  if (servo04SP[i] == servo04SP[i + 1]) {
  }
  if (servo04SP[i] > servo04SP[i + 1]) {
    for ( int j = servo04SP[i]; j >= servo04SP[i + 1]; j-
      -) {servo04.write(j);
      delay(speedDelay);
    }
    }
  if (servo04SP[i] < servo04SP[i + 1]) {
    for ( int j = servo04SP[i]; j <= servo04SP[i + 1]; j++)
      {servo04.write(j);
      delay(speedDelay);
    }
  }
// Servo 5
  if (servo05SP[i] == servo05SP[i + 1]) {
  }
  if (servo05SP[i] > servo05SP[i + 1]) {
    for ( int j = servo05SP[i]; j >= servo05SP[i + 1]; j-
      -) {servo05.write(j);
      delay(speedDelay);
    }
  }
  if (servo05SP[i] < servo05SP[i + 1]) {
    for ( int j = servo05SP[i]; j <= servo05SP[i + 1]; j++)
      {servo05.write(j);
      delay(speedDelay);
    }
  }
```

```
      // Servo 6
      if (servo06SP[i] == servo06SP[i + 1]) {
      }
      if (servo06SP[i] > servo06SP[i + 1]) {
        for ( int j = servo06SP[i]; j >= servo06SP[i + 1]; j-
          -) {servo06.write(j);
          delay(speedDelay);
        }
      }
      if (servo06SP[i] < servo06SP[i + 1]) {
        for ( int j = servo06SP[i]; j <= servo06SP[i + 1]; j++) {
        servo06.write(j);
        delay(speedDelay);
           }
      }
    }
  }
}
```

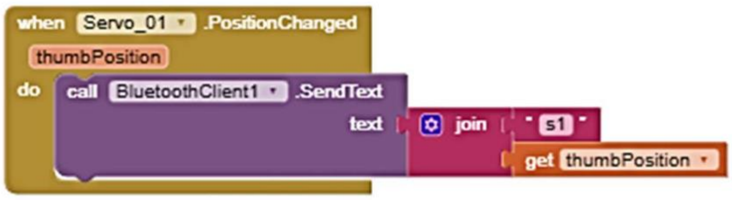
4.4 Mobile Application Code



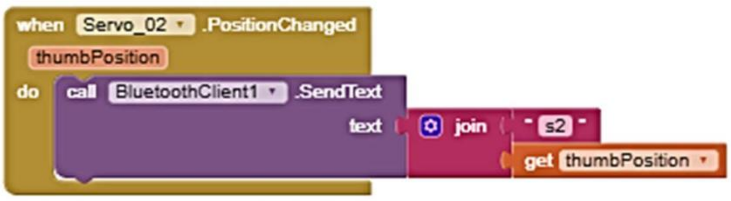
```
when BluetoothList .AfterPicking
do  set BluetoothList . Selection to  call BluetoothClient1 .Connect
                                                          address  BluetoothList . Selection

when Clock1 .Timer
do  if    BluetoothClient1 . IsConnected
    then  set Connected . Text to " CONECTADO "
          set Connected . TextColor to 🟩
    if    not BluetoothClient1 . IsConnected
    then  set Connected . Text to " NO CONECTADO "
          set Connected . TextColor to 🟥

when RESET .Click
do  call BluetoothClient1 .SendText
                          text " RESET "
    set RUN . BackgroundColor to 🟩
    set RUN . Text to " RUN "
    set Positions . Text to 0

when Speed .PositionChanged
 thumbPosition
do  call BluetoothClient1 .SendText
                          text  join  " ss "
                                      50 - get thumbPosition

when SAVE .Click
do  call BluetoothClient1 .SendText
                          text " SAVE "
    set Positions . Text to  Positions . Text + 1
```



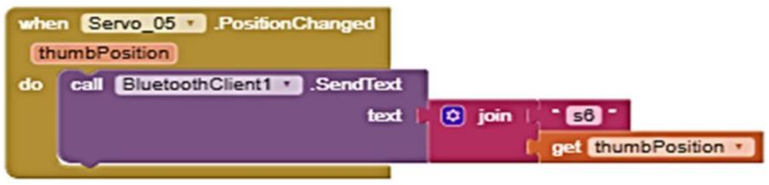
```
when RUN .Click
do  if   RUN . Text  =  "RUN"
    then set RUN . BackgroundColor to 🟥
         call BluetoothClient1 .SendText
                               text "RUN"
         set RUN . Text to "PAUSE"
    else if RUN . Text  =  "PAUSE"
    then set RUN . BackgroundColor to 🟩
         call BluetoothClient1 .SendText
                               text "PAUSE"
         set RUN . Text to "RUN"
```



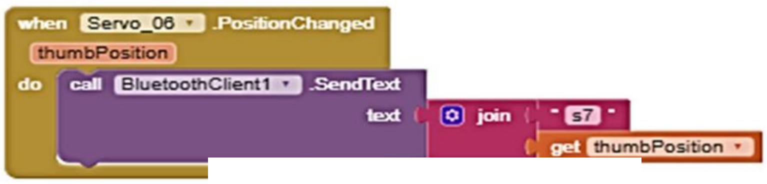
```
when Servo_01 .PositionChanged
 thumbPosition
do  call BluetoothClient1 .SendText
                          text  join  "s1"
                                      get thumbPosition
```



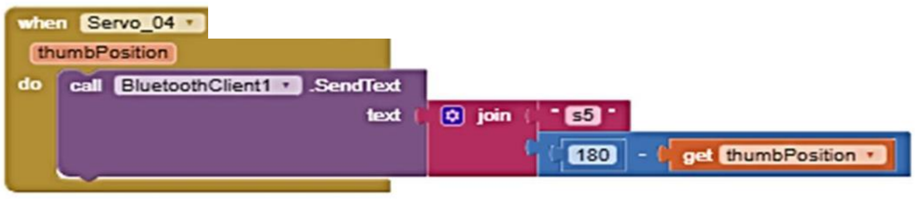
```
when Servo_02 .PositionChanged
 thumbPosition
do  call BluetoothClient1 .SendText
                          text  join  "s2"
                                      get thumbPosition
```



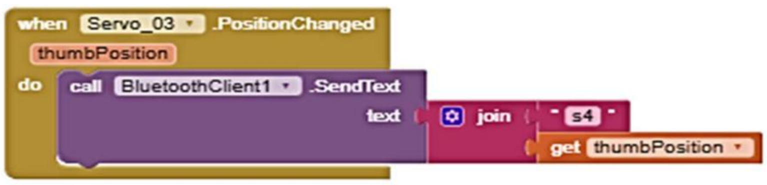
```
when Servo_05 .PositionChanged
 thumbPosition
do  call BluetoothClient1 .SendText
                          text  join  "s6"
                                      get thumbPosition
```



```
when Servo_06 .PositionChanged
 thumbPosition
do  call BluetoothClient1 .SendText
                          text  join  "s7"
                                      get thumbPosition
```



```
when Servo_04
 thumbPosition
do  call BluetoothClient1 .SendText
                          text  join  "s5"
                                      180 - get thumbPosition
```



```
when Servo_03 .PositionChanged
 thumbPosition
do  call BluetoothClient1 .SendText
                          text  join  "s4"
                                      get thumbPosition
```

4.5 Advantages of Robotic Arm

1. Cost Effectiveness: There will be no lunchbreaks, holidays, sick leave or shifttime allocated for robotic
2. Improved Quality Assurance: Robotic automation eliminates the risks of vigilancedecrement by accurately producing and checking items meet the required standard without fail.
3. Increased Productivity: Due to continuous and stress less work the production willtake place continuously and will boost the production.

   Work In Hazardous Environments: If a high level of chemicals is present, roboticautomation offers the ideal solution, as it will continue to work without harm, even in areas that have extremely high or low temperatures Robotics will prove themselves the best

4.6 Disadvantages of Robotic Arm

1. Potential Job Losses: One of the biggest concerns surrounding the introduction ofrobotic automation is the impact of jobs for workers. If a robot can perform at a faster, more consistent rate, then the fear is that humans may not be needed at all.
2. Initial Investment Costs: This is typically the biggest obstacle that will decide whether or not a company will invest in robotic automation, or wait until a later stage. The cash flow must be sustainable in the meantime and the stability of thecompany is by no means worth the risk if the returns are only marginal.
3. Hiring Skilled Staff: Over the past decade manufacturers have found it harder tosource skilled staff members to fill the specialised roles in their factories

Chapter 5

5.1 Conclusion

1. We conclude that the main parts of the structure were checked and analysed in orderto better understand the importance of correct assembly, observing the function (according to the structure where it was fitted) of each component.
2. Therefore, from the development of the Robotic Arm project, it was possible to analyze its mechanical structure, optimizing the available spaces. The robot's structure was modelled with M3 fittings and screws for its fixation, and thus,creating better mechanical stability, favoured by the rear and front structures, which connect the other structures.
3. All its development was done with the purpose of making it as didactic as possible for its subsequent assembly with the electronic part.

5.2 Area of future improvement

1. Developing an arm with absorbing the solar power (working on solar energy),Inmedical field a robot arm can play the major role Page no.43
2. For proper disposal of biomedical waste
3. An overall a nurse kit robot arm
4. A robot arm that can check blood pressure and do all the secondary work Developing a robot with 3 arm extension that can do work simultaneously

References

- Robotic Arm Control System Based on AI Wearable Acceleration Sensor. 03 Mar2021.Page no.44
- Design of a 4 DOF parallel robot arm and the firmware implementation onembedded system to transplant pot seedlings. 2020
- Howtomechatronics
- https://www.rhydolabz.com/
- https://robu.in/
- https://en.wikipedia.org/wiki/Robotic_arm
- https://www.intel.com/content/www/us/en/robotics/robotic-arm.html
- ARDUINO BASED ROBOT ARM WITH SMARTPHONE CONTROL (researchgate.net)