DESIGN AND IMPLEMENTATION OF AN AUTOMATED ROBOTIC ARM USING IOT

A PROJECT REPORT

Submitted by

ADHIBAN SIDDARTH. V	**(510418107001)**
SATHISH. S	**(510418107002)**
SURYA. M	**(510418107003)**

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND INSTRUMENTATION ENGINEERING

ARUNAI ENGINEERING COLLEGE, THIRUVANNAMALAI

ANNA UNIVERSITY: CHENNAI 600 025

JUNE 2022

VIVA – VOCE EXAMINATION

The viva voce examination of this project was submitted by

NAME OF THE CANDIDATE	**REG. NO**
ADHIBAN SIDDARTH. V	510418107001
SATHISH. S	510418107002
SURYA. M	510418107003

The project report submitted for the viva voce held on …………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"DESIGN AND IMPLEMENTATION OF AN AUTOMATED ROBOTIC ARM USING IOT"** is the bonafide work of **"ADHIBAN SIDDARTH. V, SATHISH. S and SURYA. M"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. S. SIVAKUMAR, M.E., Ph.D.,	**Mr. V. VELMURUGAN, M.E.,**
HEAD OF THE DEPARTMENT,	**SUPERVISOR,**
Department of Electrical and Electronics Engineering,	Department of Electronics and Instrumentation Engineering,
Arunai Engineering College, Thiruvannamalai,	Arunai Engineering College, Thiruvannamalai,
Tamil Nadu.	Tamil Nadu.

ACKNOWLEDGEMENT

We consider it is our privilege and primary duty to express our gratitude and request to all those who guided and inspired us in the successful completion of the project.

We express our solemn gratitude and thanks to our Honourable Vice Chairman **Er.E.V.KUMARAN,M.E.,** our Principal **Dr.R.RAVICHANDRAN,Ph.D.,** and Head of the Department **Dr. S. SIVAKUMAR, Ph.D.,** for having provided us with adequate facilities to build our project in a smooth manner.

We are really thankful to our Internal guide **M.r.V.VELMURUGAN, M.E.,** Assistant Professor, Department of Electronics and Instrumentation Engineering, Arunai Engineering College, Tiruvannamalai for his enthusiastic motivation and support, which helped us a lot in successful completion of this project.

We also extend our sincere thanks to our Department other Faculty Members and Technical Assistants who helped us at different stages in completing this project successfully.

We finally thank our parents and friends for their support for the entire duration of course without them hitting the success is a layman's work.

ADHIBAN SIDDARTH. V

SATHISH. S

SURYA. M

ABSTRACT

A robotic arm sometimes referred to as an industrial robot, is a device that runs in an equivalent way to a human arm, with several joints that either move along an axis or can rotate in certain directions. Some robotic arms are anthropomorphic and try and imitate the exact movements of human arms. They are, in most cases programmable and used to perform specific tasks, most commonly for manufacturing, fabrication, and industrial applications. Robotic arms were originally designed to help in mass production factories, most famously in the manufacturing of cars. They were also implemented to mitigate the risk of injury for workers, undertake monotonous tasks, and free workers to concentrate on the more complex production elements.

Keywords — 6 DOF, Robotic arm, IoT, Servo motor.

TABEL OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

S. NO	ABBREVATION	ACRONYM
1	SCARA	Selective Compliance Articulated Robot Arm.
2	IOT	Internet of Things
3	DOF	Degrees of Freedom
4	IP	Internet Protocol
5	MQTT	Message Queuing Telemetry Transport
6	Wi-Fi	Wireless Fidelity
7	GUI	Graphical User Interface
8	LAN	Local Area Network
9	PWM	Pulse Width Modulation

CHAPTER 1

INTRODUCTION

1.1 ROBOTIC ARM

In today's world, human work is increasing, and become tough to manage a lot of tasks in less time, this is the reason robotics and automation come into usage which reduces the human effect. Robotics is the joint work of mechanical engineering, electrical engineering, and computer of science. A robot is a machine designed to perform a particular job coded with some programming language. The Robotic arm is a copy of the human arm that can do rotational motion and translation motion as the human arm based on IoT used for doing dangerous tasks such as Welding, Gripping, Spinning, etc.

The Robotic arm types are a Cartesian robot, cylindrical robot, Spherical robot, Articulated robot, Parallel robot, SCARA robot. The number of degrees of freedom is equal to the total number of independent displacements or aspects of motion. A machine may work in two or three dimensions but have more than three degrees of freedom. The term is widely used to define the motion capabilities of robots. Our robotic arm consists of Six-DOF which increases the motion capabilities. The main goal of this project is to control the Robotic Arm manually and automatically by using the ESP32 Microcontroller in IOT to pick, move & rotate the object by giving its 3D coordinates. In Industries highly advanced robots are used and controlled manually or processors like Arduino, Raspberry Pi, Microprocessors, etc.

Here we are using C++ to program ESP32. This project focuses to create and build more compact, useful, and cheaper robotic arm to perform various functions where human is proven too dangerous to perform a specific task and to eliminate human errors to get more precise work and to do repeating tasks. This robotic arm can be controlled by using Buttons and IoT. We can control this robotic arm from a remote location wirelessly. Anyone can operate this robotic arm easily. It can work using battery power or else AC power.

1.2 INTERNET OF THINGS

The Internet of things (IoT) refers to the wide variety of physical objects increasingly connected to the Internet. IoT devices range from common household items, such as thermostats, light bulbs, and door locks, to medical products, wearables, and industrial sensors. For example, a “smart” (IoT) thermostat may be able to receive location data from a smart car and automatically adjust home temperature when a user leaves work. Although IoT devices provide unprecedented convenience and efficiency, they also raise concerns about security and privacy.

Users report privacy fears associated with constant connectivity and always-on environmental sensors, and researchers have identified insecurities in IoT device software and network communications. These issues, combined with the increasing popularity of consumer IoT devices, are motivating many studies analysing IoT network traffic to detect and prevent privacy and security vulnerabilities. Collecting IoT network traffic for these studies typically involves researchers manually interacting with devices in a laboratory setting. Researchers press buttons, touchscreens, or other user interface elements on the devices in attempt to mimic real-world user behaviours or collect traffic from as many device states as possible. For example, a researcher may attempt to record network traffic while pressing every Manuscript received September 30, 2021 button on a device in sequence or by pressing patterns of buttons (or other user interface elements) in order to examine the full scope of device behaviour. However, these types of manual device interactions are time consuming and are unlikely comprehensive, posing a significant challenge to IoT network research. IoT studies have also utilized crowdsourcing to acquire consumer IoT network traffic. This can provide large quantities of realistic data; however, crowdsourcing typically requires extensive data collection platform development, and user reports of device behaviour during traffic collection are prone to inaccuracy. IoT (Internet of Things) is a network which integrates sensors, a communication network, Internet and other transmission technologies, intelligent operations and processing. In other words, a network which is connected to all the things in the physical world. Robotics is a young field of modern technology across the engineering field. There are many ways to develop a kinematic model of a robotic arm. The fundamentals of robotics include kinematic, motion planning, Jacobian interface and control. In this project, a 6 DOF robotic arm is used with digital servo motors. The robotic arm is composed of 6 servo motors and a gripper. The moving angles of each servo is 180 degrees. The first joint represents the rotation around Z axis, the remaining motors are working according to the kinematics formula. This project works in Python and C++ programming languages. ESP32 is programmed in ESP32 and App is built in Python programming language, this App is used to control the robotic arm. This python program can run in mobile also to control the robotic arm.

A robotic arm is a widely used efficient and preferred electromechanical machine for many applications in most of the industrial and dangerous areas. The robotic arm is the product of high-tech automation in the new era. Its advantage is that it can accomplish expected operation through programming and realize remote control. The robotic arm not only has intelligence but also has anti-pressure ability. A robotic arm has similar functions of a human arm, so that can be instead of workers in high-risk environments and highly repetitive work items. Although the robotic arm is not as flexible as the human hand, it can repeat work, and snatch weights, which is far greater than the human hand. It fears neither exhaustion nor danger. Therefore, the robotic arm has been widely used in many fields. IoT connects the perception layer with the application layer through wireless communication technology. The Internet of Things injects life into things, links anything to the Internet, exchanges information and communicates to achieve control and monitoring. The Internet of Things not only brings convenience to life but also plays a vital role in various fields. The main focus of the work is to apply the Internet of Things technology to the control of the robotic arm and to realize data transmission. Manipulator control is the direction that people have been working on. More than ten years ago, when the Internet came into being, people began to study how to control the manipulator through the network. However, due to the limitations of technology at that time, the control of the manipulator was limited to the wired control.

A manipulator is an automatic machine with high flexibility and mobility. The control system of the manipulator mainly includes the main control system, server, and web application. The main control system is realized by ESP32, an open-source hardware board, to realize servo control and data transmission. As a convenient and flexible circuit board, ESP32 can realize flexible control of the manipulator by making full use of its advantages of open technology and secondary development. In the Internet field, the application of the web page is compelling. As long as there is internet access, many platform devices can achieve instant web browsing. By triggering data updates through user’s operation events. In our project web page is not created, but App is created which send information to IP address of ESP32.

CHAPTER 2

LITERATURE SURVEY

2.1 Summary

Internet of things research often involves collecting network traffic sent or received by IoT devices, in [1] they used Arduino that has no built-in Wi-Fi, we are using Esp32, it has built-in Wi-Fi. Python script is employed in the Raspberry Pi to develop a program that will be able to control the robotic arm and to receive the commands from the smart phone [2], C++ is used to program ESP32, there are 4 DOF is used in [2], in our project we are using 6 DOF. A six degree of freedom robotic arm is the most widely used mechanical device in the field of robotics. It can work in the various complex environment and has the accuracy and precision that the human arm cannot achieve [3], they used ESP8266 microcontroller, but we are using ESP32, ESP32 has more features than ESP8266. The program was loaded into ESP32 hardware because of its integrated Wi-Fi and dual-mode Bluetooth [4], so we are using ESP32 for Wi-Fi, [4] is the robot used for writing on the paper, not pick and drop. [5] This also the 6 DOF Robotic Arm, but each motor is controlled individually, but our servo motors are working simultaneously. [6] is the pick and place robotic arm, it has 5 DOF, it uses Arduino, they used MATLAB. In [7] they used keypad to control the robotic arm, we referred to use keypad from here, but they used 8051 microprocessor and stepper motor. In [8] they used Wi-Fi to control the robotic arm, so we referred it from them, but they used PS2 controller. In [9] they also used servo motor to control the robotic arm, but they used Arduino UNO and Ethernet Shield to get internet connection. In [10] they built robotic arm with 6 DOF and using servo motor, but they don’t use internet to control their robotic arm.

2.2 IOT NETWORK TRAFFIC COLLECTION WITH ROBOTIC ARM

Consumer Internet of things research often involves collecting network traffic sent or received by IoT devices. These data are typically collected via crowdsourcing or while researchers manually interact with IoT devices in a laboratory setting. However, manual interactions and crowdsourcing are often tedious, expensive, inaccurate, or do not provide comprehensive coverage of possible IoT device behaviours. We present a new method for generating IoT network traffic using a robotic arm to automate user interactions with devices. This eliminates manual button pressing and enables permutation-based interaction sequences that rigorously explore the range of possible device behaviours. We test this approach with an Arduino controlled robotic arm, a smart speaker and a smart thermostat, using machine learning to demonstrate that collected network traffic contains information about device interactions that could be useful for network, security, or privacy analyses. We also provide source code and documentation allowing researchers to easily automate IoT device interactions and network traffic collection in future studies [1].

2.3 ROBOT ASSISTED SURGERY BASED ON IOT (GYROSCOPE)

A controllable robotic arm via the use of the Internet of Things (IoT). A nurse could control the robotic arm in a hospital to assist a doctor during a surgery and take care of the patient with the picking and placing tasks using the robotic arm. In this system, a smart phone will be used to control the robotic arm. An accelerometer and a gyroscope are used to capture the gestures and postures of the smart phone. The signals of the accelerometer and the gyroscope will be captured by an Android application and sent to a

Raspberry Pi to control the robotic arm. By integrating the Internet of Things (IoT), a worldwide controllable robotic arm is achieved. The Android application is developed by using Android Studio. Python script is employed in the Raspberry Pi to develop a program that will be able to control the robotic arm and to receive the commands from the smart phone. A system with a kinematic model will be used in the Raspberry Pi to control the robotic arm. Besides, a video streaming from computer is implemented to monitor the robotic arm. The performance of the robot is investigated in term of latency of the system and IoT platform [2].

2.4 ROBOTIC ARM CONTROL BASED ON IOT WITH ARDUINO

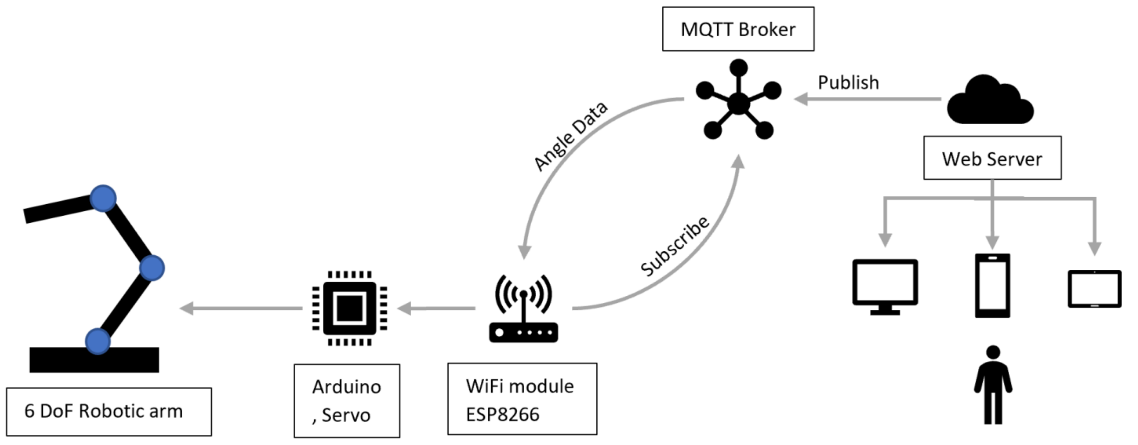


Fig. 2.1 Robotic Arm Control Based on IoT with Arduino

A six degree of freedom robotic arm is the most widely used mechanical device in the field of robotics. It can work in the various complex environment and has the accuracy and precision that the human arm cannot achieve. Generally, the motion of a robotic arm needs to be controlled by a wired teach pendant or computer control. However, in some complex environments, it is impossible to achieve short- distance control wireless control. Real-time remote data transmission is also the key to achieve precise control of the robotic arm, and if the control terminal can be realized anytime and anywhere, the robotic arm can be controlled remotely.

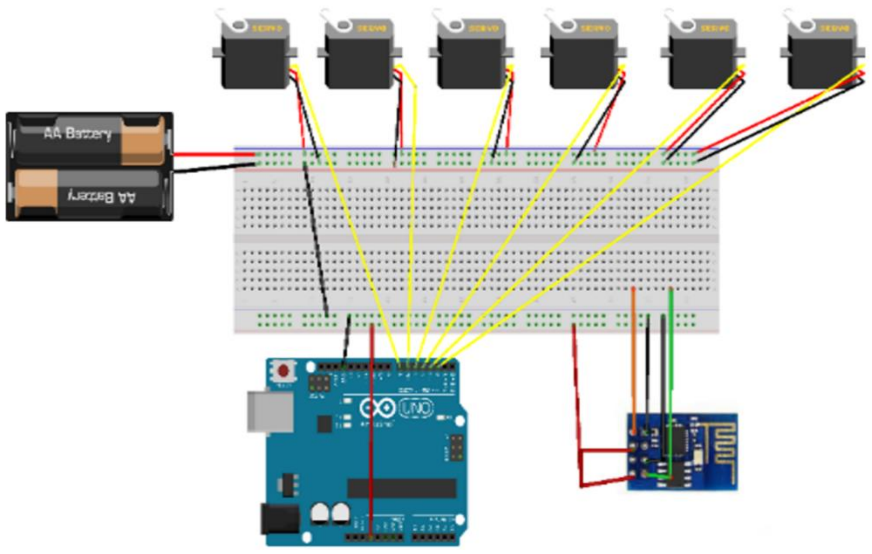


Fig. 2.2 Servo control using Arduino

Internet of Things (IoT) and web applications can solve these problems of near real-time data transmission as well as multi-platform realization at the control end. In this paper we present a design and implementation of a web-based control of the robotic arm using on MQTT (Message Queuing Telemetry Transport) communication protocol and ESP8266 (a network data transmission module). Through these technologies a platform independent web-based control of robotic devices can be realized, for accurate and real-time manipulator from remote environment [3].

2.5 ROBOT ARM FOR ONLINE TEACHING OF ROBOTICS COURSES

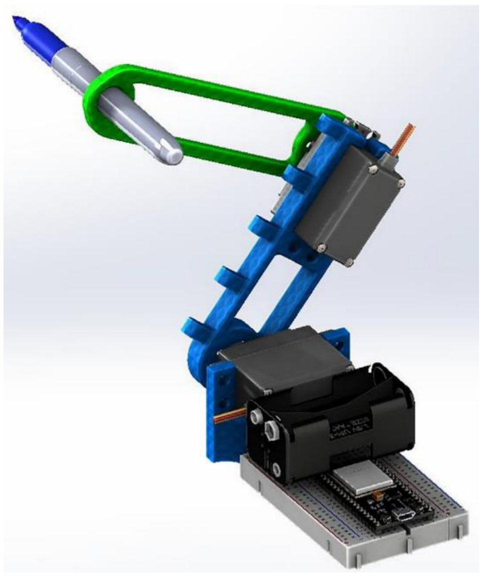


Fig. 2.3 Writing robot

This article explains the design and construction of an affordable, open-source robot arm for online teaching of robotics courses. The main goal of the proposed robotic prototype is to deal with the current situation of pandemic contingency, where students and instructors cannot attend laboratory facilities in person. The robotic system has four main components: an electromechanical robot arm structure, a control system, a Wi-Fi communications module, and a human-machine interface. The IoT (Internet of Things) robot arm can be used to demonstrate important robotics topics such as direct and inverse kinematics, which are shown by programming simple and complex motions using the Denavit-Hartenberg (DH) methodology. The capabilities of the robotic system are empowered by IoT technology, which is demonstrated with an HMI interface deployed in a smartphone using wireless Wi-Fi communication through an ESP32 microcontroller. The arm's purpose is to be a low-cost and replicable robot that aids the comprehension of robotics design through project-based learning, from the theoretical aspects to the actual coding and construction of a prototype [4]. In the current scenario machines and robots are playing an important role in the automation industry. This paper is presenting the process through which robotic arm is made with the help of Arduino and Potentiometer for controlling and coordinating the industrial processes. Here, we realize that the robotic arm has the ability to move in four directions with the help of servo motors. The movement of servo motor, UNO is responsible which converts the analog signal into the digital signal which is further received by servo motor. This project discusses about the technical imputation, the issue related with the implications and application of robotic arm in the field of automation of industries. It can be used as an artificial arm for a human who have lost his hand in any accident [5]. In this paper, the design and development of a robotic arm controller that enables use of a robotic arms as practical laboratory model for teaching and learning of robot arm algorithms is presented. The proposed system consists of OWI robotic arm with 5-Degrees of Freedom (DOF) with DC motors for each joint movement, Arduino based control unit for control of arm motion and a MATLAB based Graphical User Interface (GUI) with kinematics algorithm at the backend providing user friendly environment for controlling the end effector of robot arm remotely. The robotic arm is equipped with accelerometers to provide feedback to the microcontroller. The kinematics algorithm is implemented in MATLAB. The MATLAB based GUI, Arduino controller and accelerometers together with the robotic arm provide the practical laboratory model of teaching learning robotic arm platform. This robot arm also provides standalone operation for pick and place application with the use of Arduino controller and accelerometers [6].

2.6 MICROCONTROLLER BASED ROBOTIC ARM 6 DOF

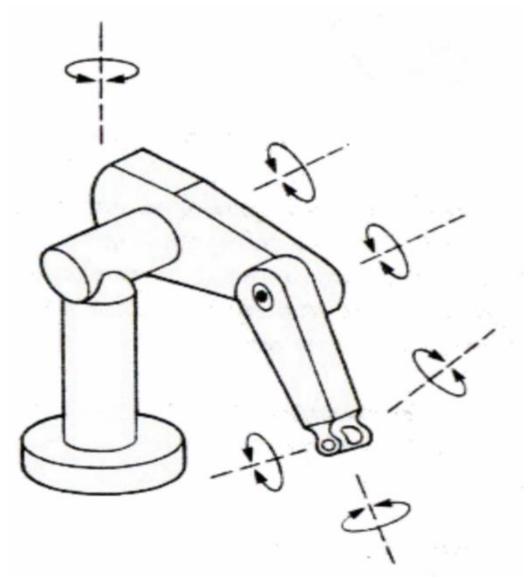


Fig. 2.4 six DOF Robot

Robotic arm has become popular in the world of robotics. The essential part of the robotic arm is a programmable microcontroller-based brick capable of driving basically three stepper motors design to form an anthropomorphic structure. The first design was for experimental use on a human-size industrial robot arm called PUMA 560 used to explore issues in versatile object handling and compliance control in grasp actions (Bejczy & Jau, 1986). This paper explains the method of interfacing the robotic arm stepper motors with the programmed 8051-based microcontroller which are used to control the robot operations. We have employed the assembly language in programming our microcontroller. A sample robot which can grab (by magnetizing) and release small objects (by demagnetizing) is built for demonstrating the method explained [7]. In recent year, with the increase usage of wireless application, the demand for a system that could easily connect devices for transfer of data over a long distance - without cables, grew stronger. This paper presents the development of a wireless mobile robot arm. A mobile robot that functional to do pick and place operation and be controlled by using wireless PS2 controller. It can move forward, reverse, turn right and left for a specific distance according to the controller specification. The development of this robot is based on Arduino Mega platform that will be interfaced with the wireless controller to the mobile robotic arm. Analysis such as speed, distance, load that can be lifted of the robot has been done in order to know its performance. Finally, this prototype of the robot is expected to overcome the problem such as placing or picking object that far away from the user, pick and place hazardous object in the fastest and easiest way [8].

2.7 CONTROL ROBOTIC ARM USING IP ADDRESS

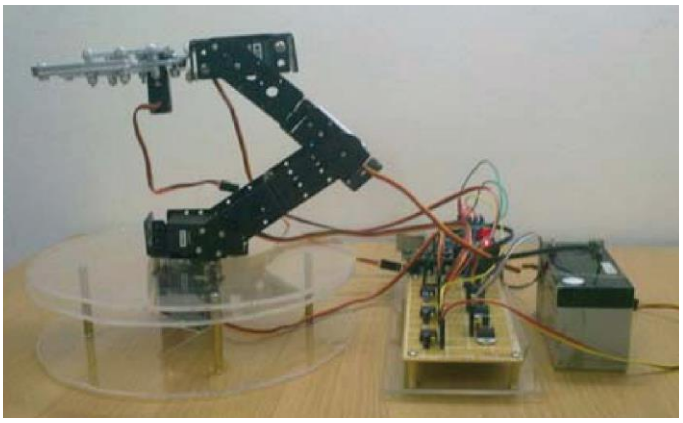
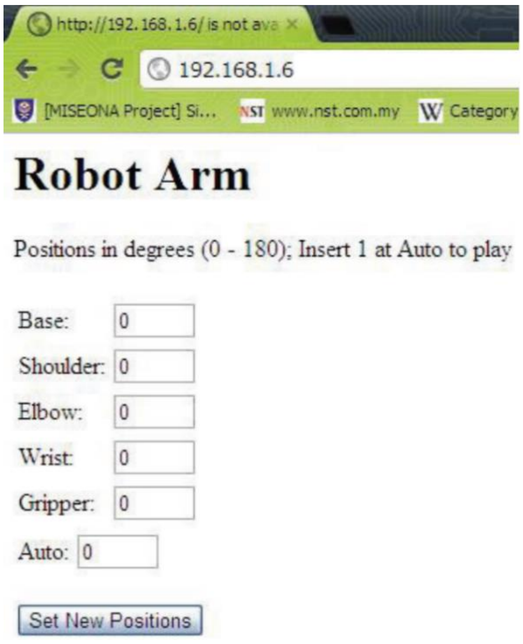


Fig. 2.5 IP Address control robot

When we talk about robots, people tend to think that robots are only suitable to use in the industry or just for the scientist to test about new technologies. However, the main function of robots is to help humans in doing work either in the industries or just helping out doing normal household chores. To bridge the gap of the normal perception of “robots are for the industries only”, internet will be use. This paper presents the development of an internet controlled robotic arm. The movement of the robot arm can be controlled by a computer via the internet. This robot can be used to demonstrate that a robot can be used inside a home for daily human chores.

The robot is controlled by Arduino Uno that interfaced with the internet using Arduino Ethernet Shield.


http://192.168.1.6/ is not ava

192.168.1.6

[MISEONA Project] Si... www.nst.com.my Category

Robot Arm

Positions in degrees (0 - 180); Insert 1 at Auto to play

Base:	0
Shoulder:	0
Elbow:	0
Wrist:	0
Gripper:	0
Auto:	0

Set New Positions

Fig. 2.6 IP address control using Browser

Two types of analysis were done for this project that is servo motor analysis and accuracy test. The accuracy test shows that the results of the actual output of the servo motor as compared to the input send to Arduino Uno via internet is between 97% to 99%. This prototype of the robot showed that the operational was successful. This user-friendly robot is expected to bridge the gap between robot and household chores [9]. The behaviour of physical systems in many situations may better be expressed with an analytical model. Robot modeling and analysis essentially involve its kinematics. For robotic manipulators having high Degrees of Freedom (DOF) with multiple degrees in one or more joints, an analytical solution to the inverse kinematics is probably the most important topic in robot modeling. This paper develops the kinematic models a 6 DOF robotic arm and analyses its workspace. The proposed model makes it possible to control the manipulator to achieve any reachable position and orientation in an unstructured environment.

2.8 ROBOTIC ARM OF 6 DOF MATLAB

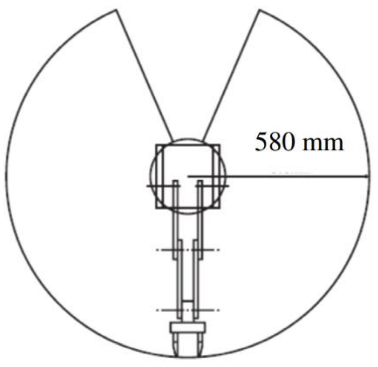


Fig. 2.7 Six DOF MATLAB Robot

The forward kinematic model is predicated on Denavit Hartenberg (DH) parametric scheme of robot arm position placement. Given the desired position and orientation of the robot end-effector, the realized inverse kinematics model provides the required corresponding joint angles. The forward kinematic model has been validated using Robotics Toolbox for MATLAB while the inverse kinematic model has been implemented on a real robotic arm. Experimental results demonstrate that using the developed model, the end-effector of robotic arm can point to the desired coordinates within precision of ±0.5cm. The approach presented in this work can also be applicable to solve the kinematics problem of other similar kinds of robot manipulators [10].

CHAPTER 3

EXISTING SYSTEM

3.1 SUMMARY

“Modeling and Analysis of a 6 DOF Robotic Arm Manipulator” is the previous existing system. Here, the behaviour of physical systems in many situations may better be expressed with an analytical model. Robot modeling and analysis essentially involve its kinematics.

For robotic manipulators having high Degrees of Freedom (DOF) with multiple degrees in one or more joints, an analytical solution to the inverse kinematics is probably the most important topic in robot modeling. This paper develops the kinematic models a 6 DOF robotic arm and analyses its workspace. The proposed model makes it possible to control the manipulator to achieve any reachable position and orientation in an unstructured environment.

The forward kinematic model is predicated on Denavit Hartenberg (DH) parametric scheme of robot arm position placement. Given the desired position and orientation of the robot end-effector, the realized inverse kinematics model provides the required corresponding joint angles. The forward kinematic model has been validated using Robotics Toolbox for MATLAB while the inverse kinematic model has been implemented on a real robotic arm. Experimental results demonstrate that using the developed model, the end-effector of robotic arm can point to the desired coordinates within precision of ±0.5cm.

The approach presented in this work can also be applicable to solve the kinematics problem of other similar kinds of robot manipulators. Analytical prediction of the behaviour of physical systems in many key situations is either extremely complicated or even impossible. Driven with the constraints to prototype a physical system, modeling finds enormous motivations to study and investigate the performance of a system. Modeling a robot involves study of its kinematic behaviour. A kinematic model is concerned with the robot's motion without considering forces producing the motions.

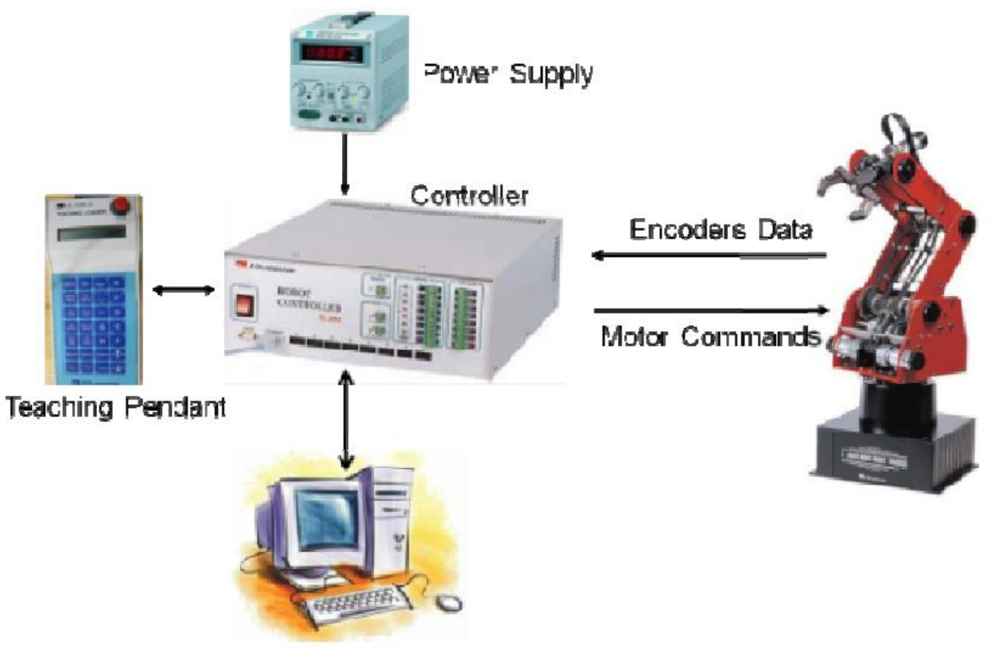


Fig. 3.1 Modeling and Analysis of a 6 DOF Robotic Arm Manipulator

3.2 KINEMATICS

The kinematics of a robotic arm deals with the study of the geometric and time-based properties of the motion and in particular how various links of a robot move with respect to one another and with time. It provides an analytical description of the spatial movements of a robot i.e., a relationship between position and orientation of robot end effector and its joint variables. The problem of kinematic modeling is usually categorized into two sub-problems. First is the forward or direct kinematics, which is the problem of solving the Cartesian position and orientation of a mechanism, given the knowledge of the kinematic structure and the joint coordinates.

CHAPTER 4

PROPOSED WORK

4.1 Internet of Things

The Internet of Things (IoT) describes the network of physical objects— “things”—that are embedded with sensors, software, and other technologies for the purpose of connecting and exchanging data with other devices and systems over the internet.

4.2 Local area network

A local area network (LAN) is a collection of devices connected together in one physical location, such as a building, office, or home. A LAN can be small or large, ranging from a home network with one user to an enterprise network with thousands of users and devices in an office or school. Regardless of size, a LAN's single defining characteristic is that it connects devices that are in a single, limited area. In contrast, a wide area network (WAN) or metropolitan area network (MAN) covers larger geographic areas. Some WANs and MANs connect many LANs together. The robotic arm is going to be controlled by using WiFi, here we are using FOG computing.

4.3 Different computing

There is Cloud, FOG, Edge computing available in Internet of things. Cloud computing is that the data is sent through internet to the cloud server. Mainly cloud is used to process and store the large amount of data. Cloud has another advantage that is the data can be obtained from any where from the world where there is internet available.

4.4 FOG computing

FOG computer is that there is no need of WAN, only LAN is enough for this. In FOG computing the data is stored in the local storage. It has some advantages, when internet is slow the data transfer from server to the client will be slow. It makes some delay; it causes some disadvantages. But in FOG computing the data storage is in local, data transfer is faster than cloud, then the process is done faster. FOG computing is used in LAN like WiFi, Ethernet etc. The local server is connected to the client using WiFi, then the data is transferring between client and server. In our project data transfer speed is important and low storage only required. So, FOG computing is best for this project. Fog computing is a computing architecture in which a series of nodes receives data from IoT devices in real time. These nodes perform real-time processing of the data that they receive, with millisecond response time. Edge computing, a strategy for computing on location where data is collected or used, allows IoT data to be gathered and processed at the edge, rather than sending the data back to a datacentre or cloud. Together, IoT and edge computing are a powerful way to rapidly analyze data in real-time. But in Edge computing the data storage must be placed nearer to the place where the data is getting or giving to them. In our project robotic arm is controlled and automation data is stored in wirelessly so Edge computing is not used in this project. Edge computing gives more speed data transfer but less flexibility. FOG computing has enough speed and flexibility for the robotic arm and it is one of the best methods also.

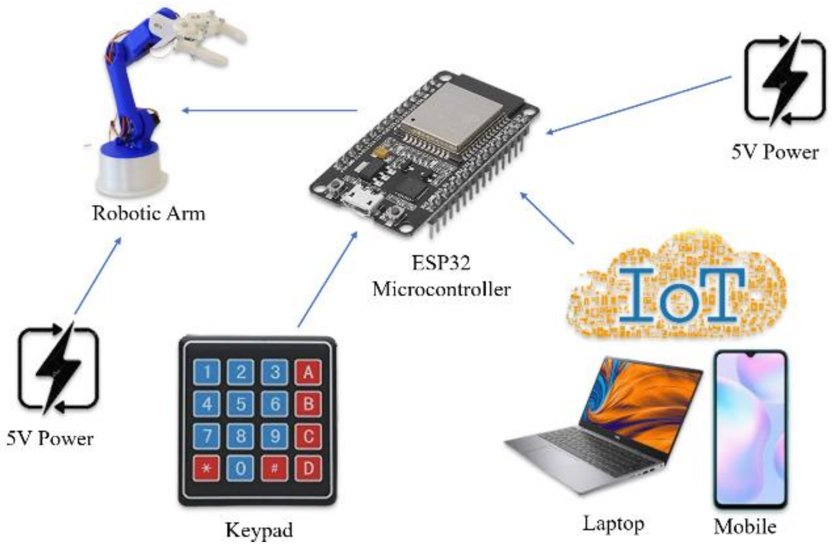


Fig. 4.1 Project diagram

The robotic arm can be controlled by using IoT through WiFi and Keypad. The servo motors of the robotic arm require 5V and ESP32 microcontroller also consumes 5v. ESP32 gives the PWM signal to each servo of the robotic arm.

4.5 Workflow

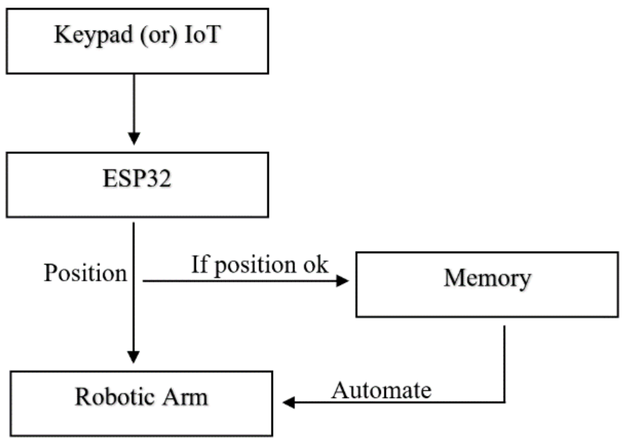


Fig. 4.2 Workflow

We give input to ESP32 using Keypad or IoT as changing the 3D Coordinate position. Then ESP32 converts the input into the position and then positions into the angles for the servo motors of the robotic arm. Then by pressing the 'Record' button we can store the desired positions in the ESP32.

After all the positions are stored, we should press the 'Automate’ button to automate the Robotic arm. Then Robotic arm moves to the stored positions repeatedly in a continuous loop. Pause the automated Robotic arm, we should press the ‘Pause’ button. After that, we should press the 'Run' button to resume the automation. Again, we press the 'Record' button to erase the stored positions, and then we can record a new position.

CHAPTER 5

CIRCUIT DIAGRAM

5.1 Introduction

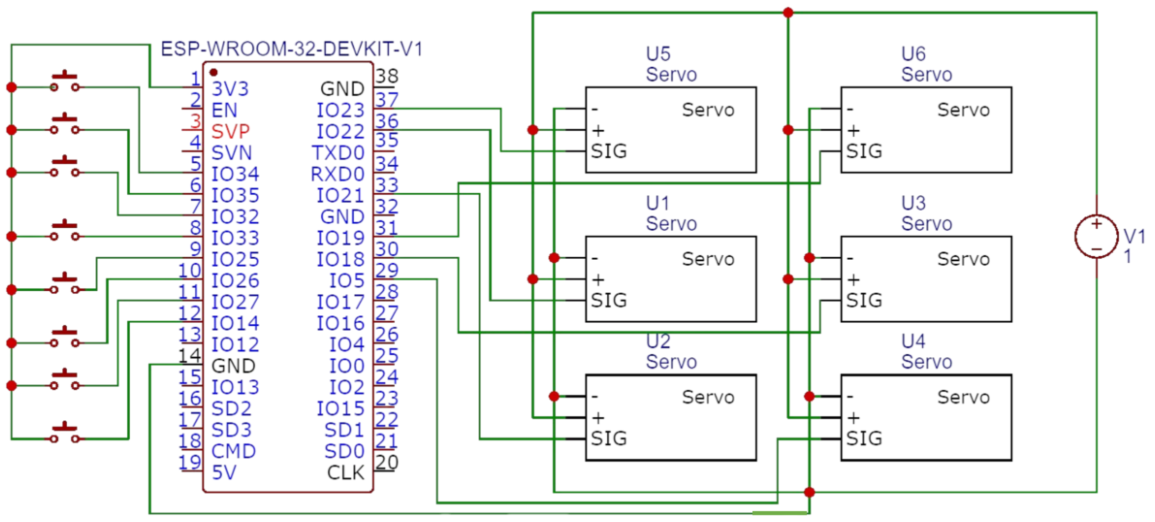


Fig. 5.1 Circuit Diagram

The above diagram is the sample diagram of our circuit diagram. 6 servo motors are connected to the PWM pins of ESP32. In our project base servo is connected to 13th pin, shoulder servo is connected to 12th pin, Elbow servo connected to 14th pin, Wrist Rotation motor is connected to 27th pin, Wrist Up Down servo motor is connected to 26th pin and gripper is connected to 25th pin. The Keypad has 8 pins each pin is connected to GPIO pins of the ESP32. Both ESP32 and servo motors operated in 5V. Total of all servo motors and ESP32 consume 6A. So 5V/6A adopter is used to power up this robotic arm.

5.2 Pulse Width Modulation

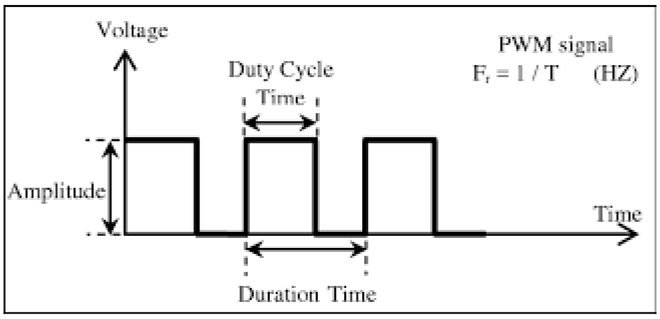


Fig. 5.2 Pulse Width Modulation

Pulse-width modulation (PWM), or pulse-duration modulation (PDM), is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts. The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast rate. The longer the switch is on compared to the off periods, the higher the total power supplied to the load. Along with maximum power point tracking (MPPT), it is one of the primary methods of reducing the output of solar panels to that which can be utilized by a battery. PWM is particularly suited for running inertial loads such as motors, which are not as easily affected by this discrete switching, because their inertia causes them to react slowly. The PWM switching frequency has to be high enough not to affect the load, which is to say that the resultant waveform perceived by the load must be as smooth as possible.

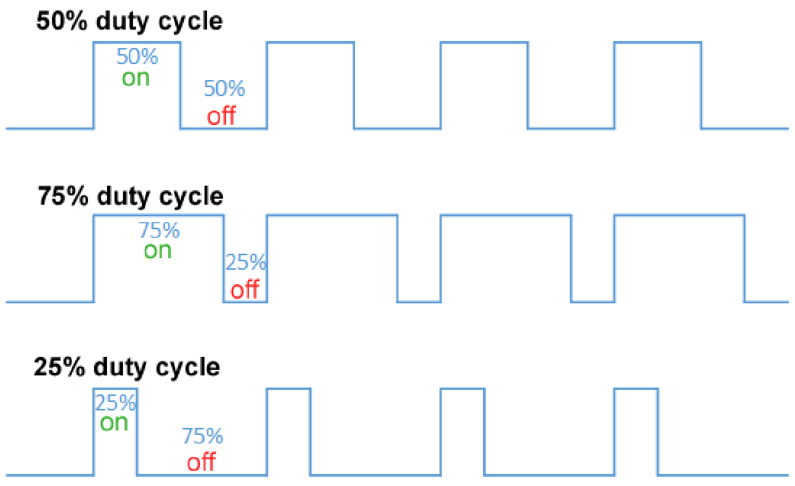


Fig. 5.3 Duty Cycle

The rate (or frequency) at which the power supply must switch can vary greatly depending on load and application. For example, switching has to be done several times a minute in an electric stove; 100 or 120 Hz (double of the utility frequency) in a lamp dimmer; between a few kilohertz (kHz) and tens of kHz for a motor drive; and well into the tens or hundreds of kHz in audio amplifiers and computer power supplies. The main advantage of PWM is that power loss in the switching devices is very low. When a switch is off there is practically no current, and when it is on and power is being transferred to the load, there is almost no voltage drop across the switch. Power loss, being the product of voltage and current, is thus in both cases close to zero. PWM also works well with digital controls, which, because of their on/off nature, can easily set the needed duty cycle. PWM has also been used in certain communication systems where its duty cycle has been used to convey information over a communications channel.

In electronics, many modern microcontrollers (MCUs) integrate PWM controllers exposed to external pins as peripheral devices under firmware control by means of internal programming interfaces. These are commonly used for direct current (DC) motor control in robotics and other applications.

5.3 Servo motor control

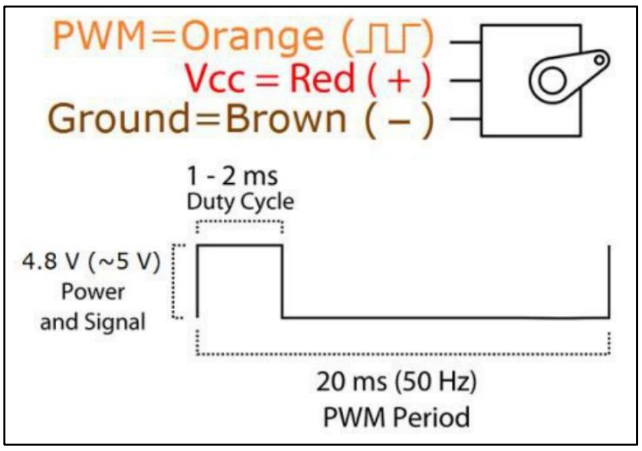


Fig. 5.4 Servo Duty Cycle

Sg90 and MG995 servo motors are used in this robotic arm. These servo motors have PWM (Orange), VCC (Red), Ground (Brown) wires. The PWM wire is used to control the servo motor in specific angle. The operating voltage is 4.8 (~5V). Duty Cycle is 1-2ms. If you send 1ms duty cycle to the servo, it will move to the 0 degree. If you send 1.5ms duty cycle to servo, it will move to 90 degree and so on.

CHAPTER 6

KINEMATICS FORMULA

6.1 Important Equation

Kinematics is the study of motion of a system of bodies without directly considering the forces or potential fields affecting the motion. In other words, kinematics examines how the momentum and energy are shared among interacting bodies. Forward kinematics refers to the use of the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters. Inverse kinematics is the use of kinematic equations to determine the motion of a robot to reach a desired position. For example, to perform automated bin picking, a robotic arm used in a manufacturing line needs precise motion from an initial position to a desired position between bins and manufacturing machines.

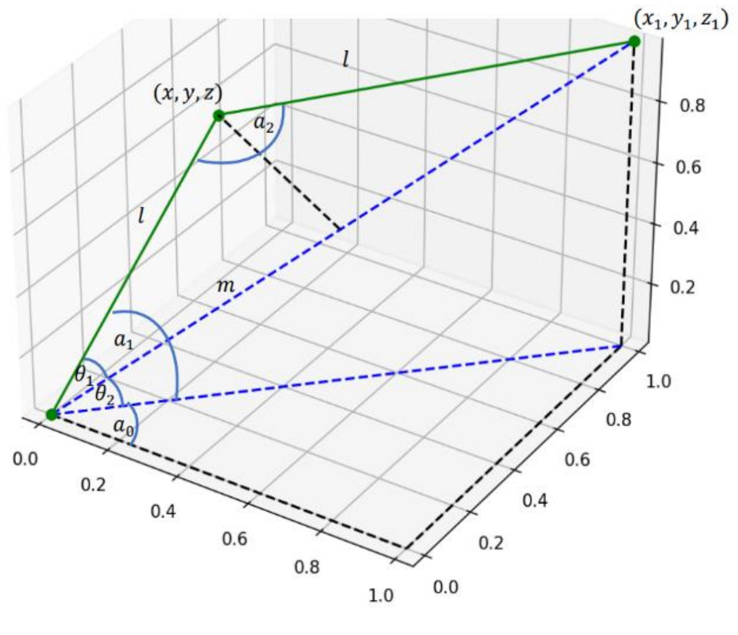


Fig. 6.1 3D graph of Kinematics

In the above diagram, the green line is represented as the links of the robotic arm. (x_1, y_1, z_1) is the given point. Then we have to find the angles a_0, a_1, a_2.

d_1 is the distance between (0,0,0) and (1,1,0)

$$d_1 = \sqrt{x_1^2 + y_1^2}$$

$$a_0 = \cos^{-1}\left(\frac{x_1}{d_1}\right)$$

d_2 is the distance between (0,0,0) and (1,1,1)

$$d_2 = \sqrt{x_1^2 + y_1^2 + z_1^2}$$

$$m = \frac{d_2}{2}$$

$$\theta_1 = \cos^{-1}\left(\frac{m}{l}\right)$$

$$\theta_2 = \cos^{-1}\left(\frac{d_1}{d_2}\right)$$

$$a_1 = \theta_1 + \theta_2$$

$$a_2 = \pi - 2 * \theta_1$$

$$a_0 = cos^{-1}\left(\frac{x_1}{\sqrt{x_1^2+y_1^2}}\right)$$

$$a_1 = cos^{-1}\left(\frac{\sqrt{x_1^2+y_1^2+z_1^2}}{2*l}\right) + cos^{-1}\left(\frac{\sqrt{x_1^2+y_1^2}}{\sqrt{x_1^2+y_1^2+z_1^2}}\right)$$

$$a_2 = \pi - 2 * cos^{-1}\left(\frac{\sqrt{x_1^2+y_1^2+z_1^2}}{2*l}\right)$$

Here $a_0, a_1, and\ a_2$ are the angles of servo motors in the Robotic arm. a_0 is the angle of the servo motor which is placed in the basement of the Robotic arm, this servo rotates the robotic arm along the Z-axis at the range of 0 to 180 angles (anti-clockwise).

6.2 Wrist Parallel to ground

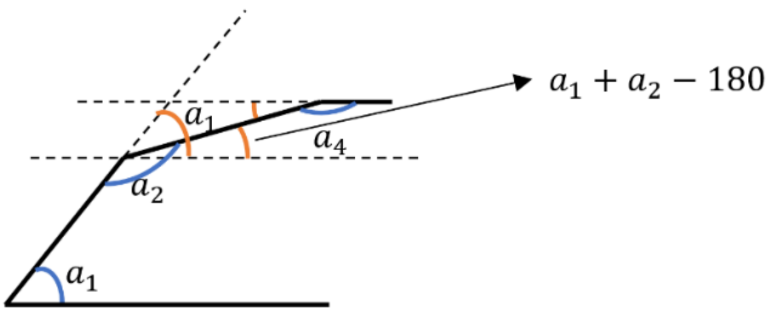


Fig. 6.2 Wrist parallel to ground

$\because Here,\ 2\pi\ radian = 360\ degree$

$\pi\ radian = 180\ degree$

$a_4 = \pi - (a_1 + a_2 - \pi)$

$a_4 = 2\pi - a_1 - a_2$

For our project, the wrist angle is calibrated for our convenience by removing $\pi/2$ in a_4.

$a_4 = 2\pi - a_1 - a_2 - \pi/2$

$a_4 = 3\pi/2 - a_1 - a_2$

This a_4 angle is used to keep the wrist parallel to the ground surface for all the movement of the robotic arm. The Robotic arm moves anywhere in its range and the wrist is always parallel to the ground surface. We can change the angle of the Robotic Arm's wrist at any angle in between the manual control or in automation.

The Robotic arm can reach any point within its range using the above equations. But the robotic arm needs a smooth move from one point to another point. So, the points in between the two points are used to move the robotic arm smoothly.

6.3 Straight line equation

We must find points in between two points.

Assume (x_1, y_1, z_1) *and* (x_2, y_2, z_2) are the two points.

$$\frac{x-x_1}{x_2-x_1} = \frac{y-y_1}{y_2-y_1} = \frac{z-z_1}{z_2-z_1} \ldots (1)$$

(1) is the Equation of straight line in 3D.

$$(1) \Rightarrow y = (y_2 - y_1)\frac{x-x_1}{x_2-x_1} + y_1 \ldots (2)$$

$$(1) \Rightarrow z = (z_2 - z_1)\frac{x-x_1}{x_2-x_1} + z_1 \ldots (3)$$

Now, we take 10 to 100 numbers between x_1 and x_2. Apply these numbers in (2) and (3).

Then we can get the points in between (x_1, y_1, z_1) *and* (x_2, y_2, z_2).

When $x_1 = x_2$, we cannot find numbers between x_1 and x_2.

Then replace x instead of y and y instead of x in (2) and (3).

When $y_1 = y_2$, we cannot find numbers between y_1 and y_2.

Then replace y instead of z and z instead of y in the earlier two equations.

6.4 Workspace Analysis

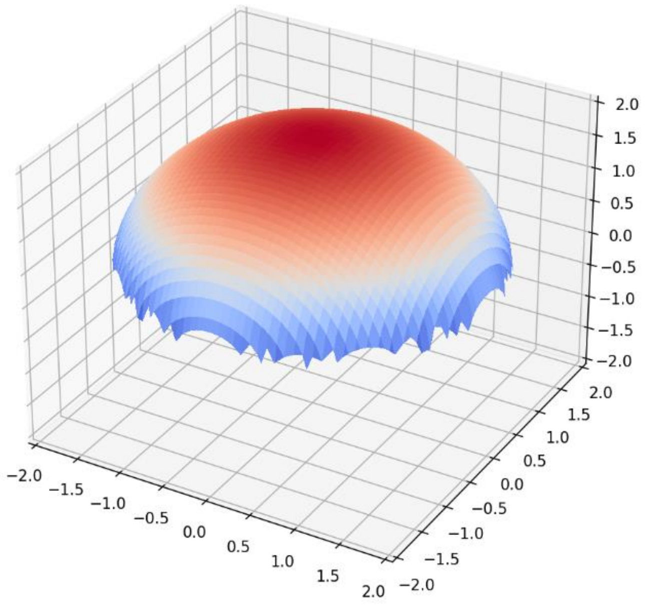


Fig. 6.3 Workspace

A robotics' workspace is the space in which the robot operates on the production line or in a work cell. Each industrial articulated robot has a very specific workspace for which it can operate and move around within.

Equation of sphere: $x^2 + y^2 + z^2 = r^2$

r value is the radius of the sphere. Here $r = 2.l$.

$\Rightarrow x^2 + y^2 + z^2 = (2.l)^2$

$\Rightarrow x^2 + y^2 + z^2 = 4.l^2$

The robotic arm range is within this sphere above the XY plane.

6.5 Workspace Examples

Considering $(x_1, y_1, z_1) = (1,0,1)$ the position and orientation of the end-effector will be shown below.

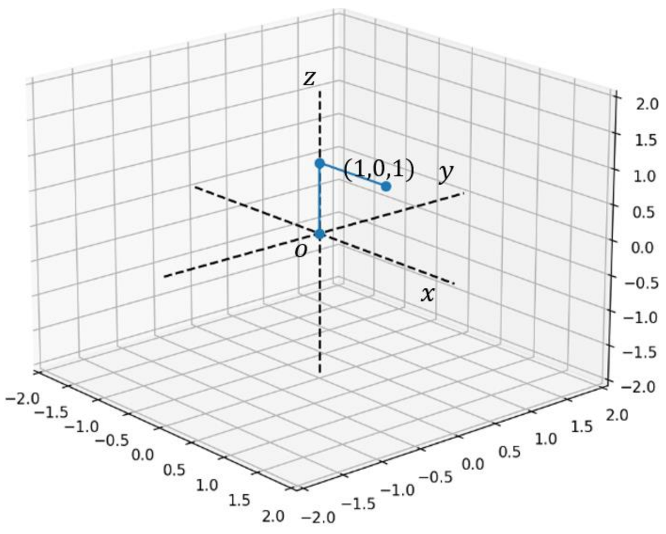
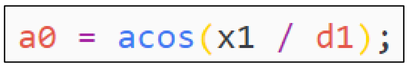


Fig. 6.4 Workspace example 1

Here we consider $l = 1$. The angles $a_0, \ a_1$ & a_2 will be 0.0, 1.570, 1.570 in radian. Then these angles are converted into degrees as 0, 90, and 90 degrees. The return value of the acos() in C++ will be radian only. Here acos means cos^{-1}.



```
a0 = acos(x1 / d1);
```

Fig. 6.5 acos() in C++ and Python

The value given to the servos should be in degrees, so we should convert the radian into degrees. If we consider $(x_1, y_1, z_1) = (2,2,2)$ the result will be invalid, because (2,2,2) is outside the sphere $x^2 + y^2 + z^2 = 4$.

Considering $(x_1, y_1, z_1) = (1,1,1)$ the position and orientation of the end-effector will be shown below.

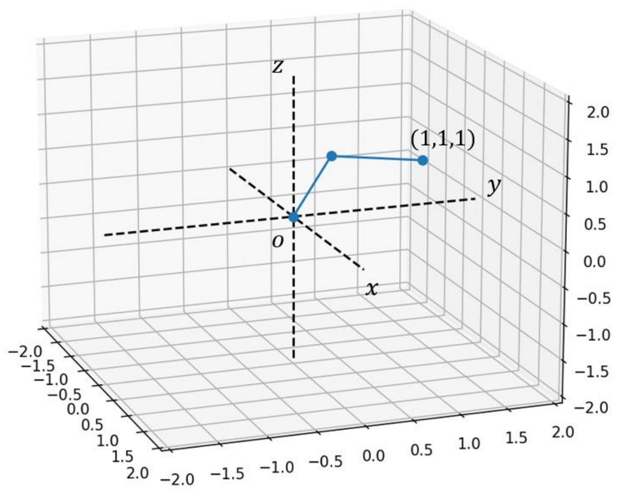


Fig. 6.6 Workspace example 2

The angles a_0, a_1 & a_2 will be 0.785, 1.139, 2.094 in radian, 45, 65.26, 119.99 degrees.

Considering $(x_1, y_1, z_1) = (-1,1,1)$ the position and orientation of the end-effector will be shown in below.

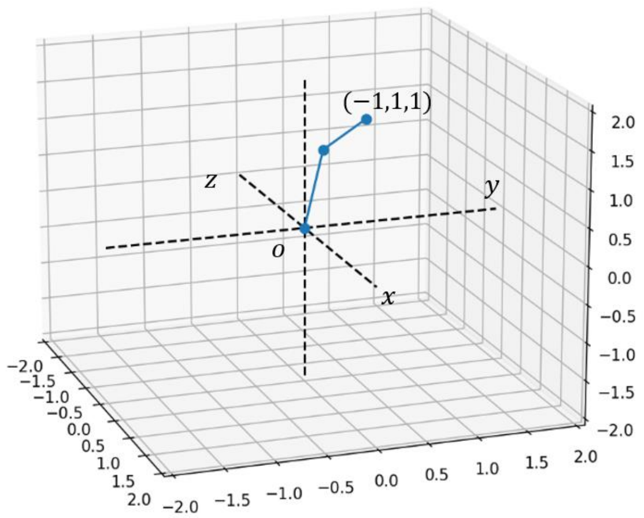


Fig. 6.7 Workspace example 3

The angles a_0, a_1 & a_2 will be 2.356, 1.139, 2.094 in radian, 135.0, 65.264, 119.999 degrees.

CHAPTER 7

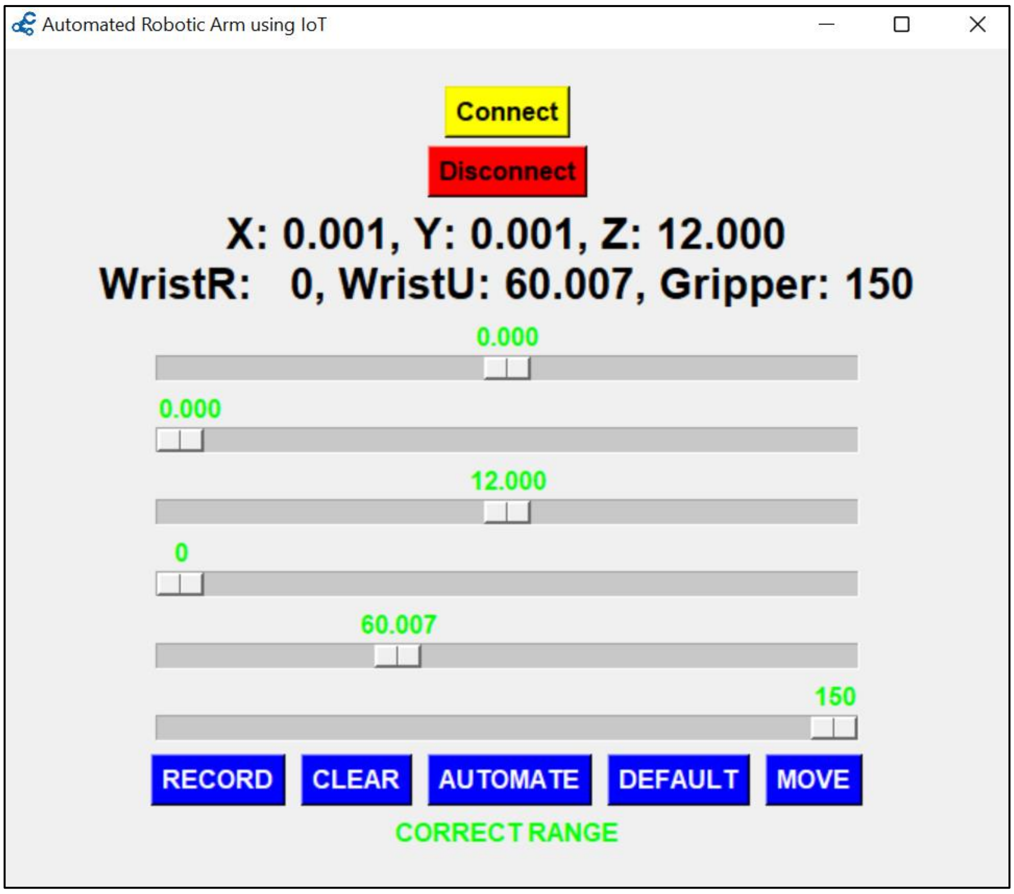
SOFTWARE DETAILS

7.1 Introduction

Here we are created two Apps:

- ✓ Robotic Arm
- ✓ Test

7.2 Robotic Arm App



X: 0.001, Y: 0.001, Z: 12.000
WristR: 0, WristU: 60.007, Gripper: 150

0.000
0.000
12.000
0
60.007
150

CORRECT RANGE

Fig. 7.1 Robotic arm app

Robotic arm app is made with the Kinematics principle of the robotic arm. When the X, Y, Z values are changed in this app, then a0, a1, a2, a4 angles are obtained from X, Y, Z through formula. This app is used to control the robotic arm in X, Y, Z coordinates using WiFi connected with ESP32 microcontroller.

7.3 Steps to use this app

- First powerup the ESP32, then only WiFi Access Point is enabled.
- After that connect your PC or Laptop to ESP32 Access Point WiFi and enter the password.
- After WiFi is connected to the PC or Laptop. Open the Robotic Arm App.
- Click Connect Button.
- If connection is successful, then '---CONNECTED---' is displayed in the app.

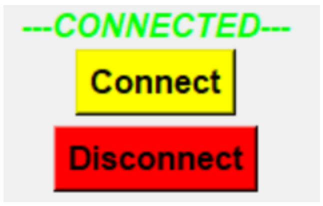


Fig. 7.2 App connected

- If connection is not success, then 'TimeOutError' is displayed in the app. ('TimeOutError' comes after some time delay so wait after press 'Connect' Button when this Button do not come back its normal position.)

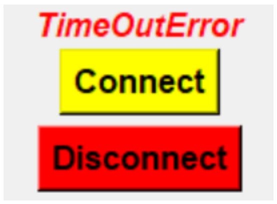
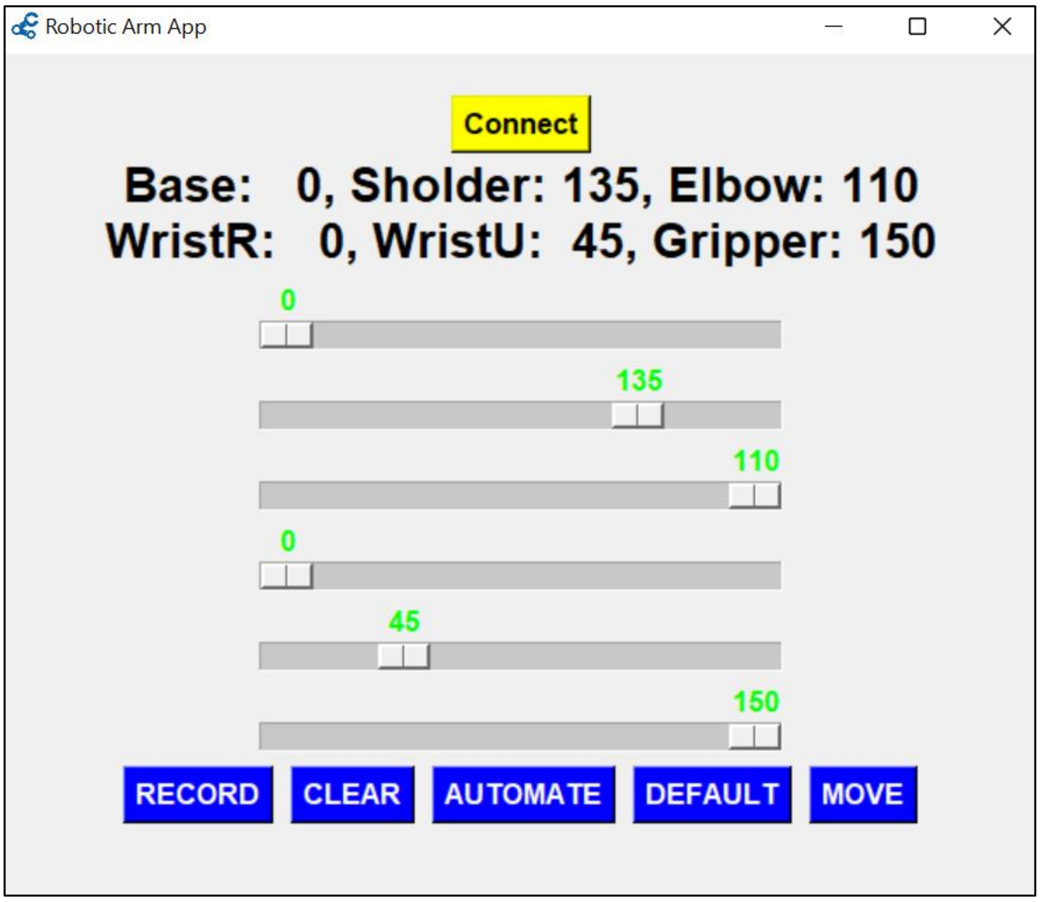


Fig. 7.3 App timeout error

- If 'TimeOutError' comes WiFi is not connected properly. So, reconnect the ESP32 Access Point WiFi.

- If nothing is happened, when clicking of the ‘Connect’ Button. Then there is no WiFi connected to your system. So, connect ESP32 WiFi to your system.
- After successful connection, Click ‘DEFAULT’ Button. Then powerup the Robotic arm and Click ‘MOVE’ Button to move our robotic arm in default position.
- After that adjust X, Y, Z, Wrist Rotate, Wrist Up Down, Gripper sliders and click ‘MOVE’ button, then robotic arm moves according to the adjustments.
- Here you can record the adjustments and automate the robotic arm.
- Adjust the sliders for required position and Click ‘RECORD’ button to record each position and then click ‘AUTOMATE’ button, then robotic arm moves these recorded positions automatically three times (three times is default, you can change it).
- Even the Automation completed the recorded positions are saved in your system. After closing the app, the recorded positions are still saved in the system.
- We can reuse that recorded positions again and again. We can add additional position after the previous positions by clicking ‘RECORD’ button.
- If you want to clear all saved positions then you can click ‘CLEAR’ button to clear all the recorded values. Then we should record from the beginning.
- The Wrist Up Down slider is automatically adjusted according to X, Y, Z.
- You can also adjust Wrist Up Down slider. It gives flexibility to operate robotic arm.
- You can close this app by clicking ‘Disconnect’ button also.

7.4 Test App



Base: 0, Sholder: 135, Elbow: 110
WristR: 0, WristU: 45, Gripper: 150

Fig. 7.4 Test app

Test app also works as similar as Robotic Arm App. But it controls the robotic arm's servo motors individually, you can automate, move robotic arm, move to default position, record, clear. But recorded data will be gone when app is closed. It is used for testing the robotic arm servo motors.

CHAPTER 8

SOURCE CODE

8.1 Files

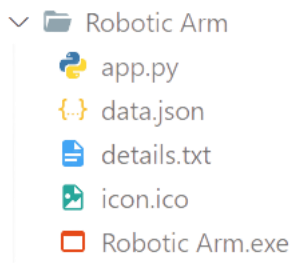

Robotic Arm
app.py
data.json
details.txt
icon.ico
Robotic Arm.exe

Fig. 8.1 Robotic arm coding files

8.2 app.py

```
from tkinter import *

import socket

import json

from math import *

from time import sleep

import sys

with open("details.txt", "r") as f:

    details = json.loads(f.read())

with open("data.json", "r") as f:

    data = json.loads(f.read())

x = details["x"]
```

```
y = details["y"]
z = details["z"]
l = details["length"]
base = details["base"]
sholder = details["sholder"]
elbow = details["elbow"]
wristR = details["wristR"]
wristU = details["wristU"]
gripper = details["gripper"]
baseList = data["baseList"]
sholderList = data["sholderList"]
elbowList = data["elbowList"]
wristRList = data["wristRList"]
wristUList = data["wristUList"]
gripperList = data["gripperList"]
sock = socket.socket()
host = details["IP"]  # ESP32 IP in local network
port = details["port"]  # ESP32 Server Port
def storeData():
    global data, baseList, sholderList, elbowList, wristRList, wristUList,
gripperList
    data = {
        "baseList": baseList,
        "sholderList": sholderList,
        "elbowList": elbowList,
        "wristRList": wristRList,
        "wristUList": wristUList,
        "gripperList": gripperList,
    }
    with open("data.json", "w") as f:
        f.write(json.dumps(data))
correct_range = True
def formula():
    global x, y, z, base, sholder, elbow, wristU
    try:
        if x == 0:
            x = 0.001
        if y == 0:
            y = 0.001
        if z == 0:
            z = 0.001
        base = acos(x / sqrt(pow(x, 2) + pow(y, 2)))
        sholder = acos(sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2)) / (2 * l)) +
acos(
            sqrt(pow(x, 2) + pow(y, 2)) / sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2))
        )
        elbow = pi - 2 * acos(sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2)) / (2 * l))
        wristU = 3 * pi / 2 - sholder - elbow
        base *= 180 / pi
        sholder *= 180 / pi
        elbow *= 180 / pi
        elbow = 180 - elbow
        wristU *= 180 / pi
        s5.set(wristU)
    except ValueError:
        l3.configure(text="ValueError", fg="#FF0000")
def display(a):
    l3.configure(text=a)
def check():
    global x, y, z, base, sholder, elbow, wristU, correct_range
    if (
        1 < sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2)) < 24
        and 0 <= elbow <= 120
        and 0 <= base <= 180
        and 0 <= sholder <= 180
        and 0 <= wristU <= 180
    ):
        l3.configure(text="CORRECT RANGE", fg="#00FF00")
        correct_range = True
    else:
        l3.configure(text="OUT OF RANGE", fg="#FF0000")
        correct_range = False
def update():
    l2.configure(
        text=f"X: {x:3.3f}, Y: {y:3.3f}, Z: {z:3.3f}\nWristR: {wristR:3d},
WristU: {wristU:3.3f}, Gripper: {gripper:3d}"
    )
def connect():
    try:
        sock.connect((host, port))
        l1.configure(text="---CONNECTED---", fg="#00FF00")
        formula()
        update()
    except TimeoutError:
        l1.configure(text="TimeOutError", fg="#FF0000")
def send():
    sock.send(
f"{int(base):3d}{int(sholder):3d}{int(elbow):3d}{wristR:3d}{int(wristU):3
d}{gripper:3d}".encode(
            "utf-8"
        )
    )
def xf(a):
    global x
    x = float(a)
    formula()
    update()
    check()
def yf(a):
    global y
    y = float(a)
    formula()
    update()
    check()
def zf(a):
    global z
    z = float(a)
    formula()
    update()
    check()
def wristRf(a):
    global wristR
    wristR = int(a)
    update()
def wristUf(a):
    global wristU
    wristU = float(a)
    update()
def gripperf(a):
    global gripper
    gripper = int(a)
    update()
win = Tk()
win.geometry("650x550")
win.iconbitmap("icon.ico")
win.title("Automated Robotic Arm using IoT")
l1 = Label(text="", font=("TimesNewRoman", 12, "italic", "bold"))
l1.pack()
b1 = Button(
    text="Connect", command=connect, font=("TimesNewRoman", 12,
"bold"), bg="#FFFF00"
)
b1.pack()
def disconnect():
    sock.close()
    # l1.configure(text="---DISCONNECTED---", fg="#FF0000")
    sys.exit()
Button(
    text="Disconnect",
    command=disconnect,
    font=("TimesNewRoman", 12, "bold"),
    bg="#FF0000",
).pack(padx=5, pady=5)
l2 = Label(text="", font=("TimesNewRoman", 20, "bold"))
l2.pack()
scale_length = 450
s1 = Scale(
    from_=details["s1From"],
    to=details["s1To"],
    orient=HORIZONTAL,
    length=scale_length,
    command=xf,
    resolution=0.001,
    font=("TimesNewRoman", 12, "bold"),
    fg="#00FF00",
)
s1.set(details["x"])
s1.pack()
s2 = Scale(
    from_=details["s2From"],
    to=details["s2To"],
    orient=HORIZONTAL,
    length=scale_length,
    resolution=0.001,
    command=yf,
    font=("TimesNewRoman", 12, "bold"),
    fg="#00FF00",
)
s2.set(details["y"])
s2.pack()
s3 = Scale(
    from_=details["s3From"],
    to=details["s3To"],
    orient=HORIZONTAL,
    length=scale_length,
    resolution=0.001,
    command=zf,
    font=("TimesNewRoman", 12, "bold"),
    fg="#00FF00",
)
s3.set(details["z"])
s3.pack()
s4 = Scale(
    from_=details["s4From"],
    to=details["s4To"],
    orient=HORIZONTAL,
    length=scale_length,
    command=wristRf,
    font=("TimesNewRoman", 12, "bold"),
    fg="#00FF00",
)
s4.set(details["wristR"])
s4.pack()
s5 = Scale(
    from_=details["s5From"],
    to=details["s5To"],
    orient=HORIZONTAL,
    resolution=0.001,
    length=scale_length,
    command=wristUf,
    font=("TimesNewRoman", 12, "bold"),
    fg="#00FF00",
)
s5.set(details["wristU"])
s5.pack()
s6 = Scale(
    from_=details["s6From"],
    to=details["s6To"],
    orient=HORIZONTAL,
    length=scale_length,
    command=gripperf,
    font=("TimesNewRoman", 12, "bold"),
    fg="#00FF00",
)
s6.set(details["gripper"])
s6.pack()
f2 = Frame(win)
```

```
f2.pack()
def move():
    if correct_range:
        send()
        display("MOVE")
        l3.configure(fg="#FF00FF")
def record():
    if correct_range:
        global baseList, sholderList, elbowList, wristRList, wristUList,
gripperList, base, sholder, elbow, wristR, wristU, gripper
        baseList.append(base)
        sholderList.append(sholder)
        elbowList.append(elbow)
        wristRList.append(wristR)
        wristUList.append(wristU)
        gripperList.append(gripper)
        display(f"RECORDED [ {len(baseList)} ]")
        l3.configure(fg="#FF00FF")
        storeData()
def clear():
    global baseList, sholderList, elbowList, wristRList, wristUList,
gripperList
```

```
    baseList = []
    sholderList = []
    elbowList = []
    wristRList = []
    wristUList = []
    gripperList = []
    display("CLEAR")
    l3.configure(fg="#FF00FF")
    storeData()
def default():
    global x, y, z, base, sholder, elbow, wristR, wristU, gripper
    x = details["x"]
    y = details["y"]
    z = details["z"]
    wristR = details["wristR"]
    gripper = details["gripper"]
    formula()
    s1.set(details["x"])
    s2.set(details["y"])
    s3.set(details["z"])
    s4.set(details["wristR"])
    s6.set(details["gripper"])
    display("DEFAULT")
    l3.configure(fg="#FF00FF")
def automate():
    global baseList, sholderList, elbowList, wristRList, wristUList,
gripperList
    for _ in range(3):
        for base, sholder, elbow, wristR, wristU, gripper in zip(
            baseList, sholderList, elbowList, wristRList, wristUList, gripperList
        ):
            sock.send(

f"{int(base):3d}{int(sholder):3d}{int(elbow):3d}{wristR:3d}{int(wristU):3
d}{gripper:3d}".encode(
                    "utf-8"
                )
            )
            sleep(2.5)
    display("Automation process completed")
    l3.configure(fg="#FF00FF")
Button(
    f2,
    text="RECORD",
    command=record,
    font=("TimesNewRoman", 12, "bold"),
    bg="#0000FF",
    fg="#FFFFFF",
).pack(side=LEFT, padx=5, pady=5)
Button(
    f2,
    text="CLEAR",
    command=clear,
    font=("TimesNewRoman", 12, "bold"),
    bg="#0000FF",
    fg="#FFFFFF",
).pack(side=LEFT, padx=5, pady=5)
Button(
    f2,
    text="AUTOMATE",
    command=automate,
    font=("TimesNewRoman", 12, "bold"),
    bg="#0000FF",
    fg="#FFFFFF",
).pack(side=LEFT, padx=5, pady=5)
Button(
    f2,
    text="DEFAULT",
    command=default,
    font=("TimesNewRoman", 12, "bold"),
    fg="#FFFFFF",
    bg="#0000FF",
).pack(side=LEFT, padx=5, pady=5)
Button(
    f2,
    text="MOVE",
    command=move,
    font=("TimesNewRoman", 12, "bold"),
    bg="#0000FF",
    fg="#FFFFFF",
).pack(side=LEFT, padx=5, pady=5)
f3 = Frame(win)
f3.pack()
l3 = Label(f3, text="", font=("TimesNewRoman", 12, "bold"))
l3.pack()
win.mainloop()
```

8.3 data.json

```
{"baseList": [], "sholderList": [], "elbowList": [], "wristRList": [],
"wristUList": [], "gripperList": []}
```

8.4 details.txt

```
{"IP": "192.168.4.1", "port": 80, "base": 0, "sholder": 135, "elbow": 135,
"wristR": 0, "wristU": 45, "gripper": 150, "s1From": -24, "s1To": 24,
"s2From": 0, "s2To": 24, "s3From": 0, "s3To": 24, "s4From": 0, "s4To":
180, "s5From": 0, "s5To": 180, "s6From": 90, "s6To": 150, "x": 0, "y": 0,
"z": 12, "length": 12}
```

8.5 Robotic Arm.exe – This is the Executable file. It is our app. When we click it, Robotic arm app will open.

CHAPTER 9

HARDWARE DETAILS

9.1 Hardware Requirement

Table 9.1 Hardware Requirement

S. NO	APPARATUS	QUANTITY	SPECIFICATION
1	Mg995 servo motor	3	180 degrees
2	Sg90 servo motor	3	180 degrees
3	Robotic arm plastic arm	1	-
4	ESP32 microcontroller	1	30 pin DEVKITV1
5	Adaptor Power Supply	1	5 Volt 6 AMP / 30WATT DC Switched
6	Jumper wires	30	-

9.2 MG995 Servo Motor

Fig. 9.1 MG995 servo motor

MG995 servo is a simple, commonly used standard servo for your mechanical needs such as robotic head, robotic arm. It comes with a standard 3-pin power and control cable for easy using and metal gears for high torque. A Me RJ25 Adapter also help you to connect the servo with Me Baseboard or Make block Orion easily.

Specification:

Table 9.2 MG995 Specification

Gear type	5 metal gear
Limit angle	180°±5°
Motor	DC motor
Operating voltage	4.8V
Idle current	5mA
Running current	350mA
Stall current	1500mA
Command signal	Pulse width modification
Pulse width range	500-2500usec

9.3 SG90 Servo Motor

Fig. 9.2 SG90 servo motor

Micro Servo Motor SG90 is a tiny and lightweight server motor with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos.

Specification:

Table 9.3 SG90 Specification

Gear type	Glass Fibre
Limit angle	180°
Motor	DC motor
Operating voltage	3.0 ~ 7.2 V
Idle current	10mA
Running current	220 ±50mA
Stall current	650 ±80mA
Command signal	Pulse width modification
Pulse width range	500-2500usec

9.4 Robotic Arm Plastic Arm

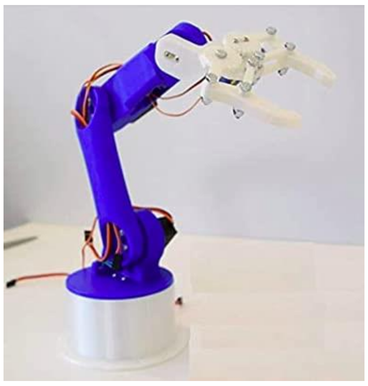


Fig. 9.3 Robotic arm plastic arm

This robotic arm plastic mode is made as 6 DOF. Base, Elbow, Shoulder of robotic arm are operated by MG955 servo motors and Wrist

Rotate, Wrist Up Down, Gripper are controlled by SG90 servo motors. The gaps and slots of servo motors are created in this plastic model. The servo motors are mounted with screws. The base of the robotic arm has 4 screws. This robotic arm can be mounted on any surface for the stable movement of the arm.

9.5 ESP32 Microcontroller

Fig. 9.4 ESP32 microcontroller

ESP32 is a series of low-cost, low-power system on a chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth. The ESP32 series employs either a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, Xtensa LX7 dual-core microprocessor or a single-core RISC-V microprocessor and includes built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power-management modules. ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process. It is a successor to the ESP8266 microcontroller.

9.6 Adaptor Power Supply

Fig. 9.5 5V/6A AC to DC adaptor

This is 5 Volt 6 AMP / 30-watt DC Switched Adaptor Power Supply. Each servo motor requires 5V and min 5mA current. Running currents are 280mA and 350mA for SG90 and MG995 servo motors. Stall currents are 1500mA and 730mA for SG90 and MG995 servo motors. There are three SG90 and three MG995 servos used in this robotic arm.

$$Total\ Current\ (mA) = (1500)(3) + (730)(3) = 6690mA$$

Therefore, robotic arm requires 6A of current to run well. So 5V/6A AC to DC Adaptor Power Supply is used.

CHAPTER 10

HARDWARE MODULE

10.1 Overview

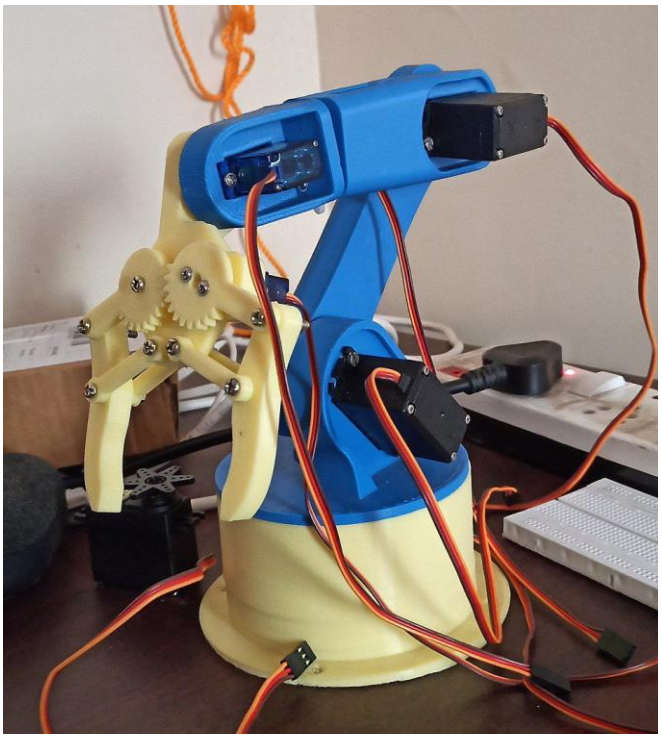


Fig. 10.1 Robotic arm plastic model

Hardware module is a template that hardware CIs are deployed from. Each Hardware Model template specifies the rack display size, depth, image, part slots and connectivity, and power calculations for a given device model. The above image shows that the Hardware module of the robotic arm. This is fill of 3D printed model. This model has 6 slots for 6 servo motors. Bottom three slots for MG995 servo motors.

The remaining three slots for SG90 servo motors. Within CAD, there are three main types of 3D modeling – solid, wireframe, and surface – and each has its own advantages and disadvantages. Of course, there are other types, but most exist either as a subset of these three or are highly specialized for their specific purposes. This robotic arm project is solid model. This module is Modular designed. Modular design, or modularity in design, is a design principle that subdivides a system into smaller parts called modules (such as modular process skids), which can be independently created, modified, replaced, or exchanged with other modules or between different systems.

10.2 Links and Joints

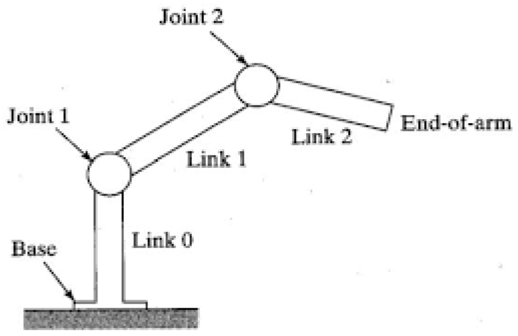


Fig. 10.2 Links and joints

The links are the rigid members connecting the joints. The joints (also called axes) are the movable components of the robot that cause relative motion between adjacent links.

Types of joints used in robots.

- Rotational Joint
- Linear Joint
- Twisting Joint
- Orthogonal Joint
- Revolving Joint

In our project, rotational joints are used.

The servo a_0 is placed at base. Servo a_1 is placed at Joint 1. Servo a_3 is placed at Joint 2. Then remaining three servos are placed at end effector.

10.3 Gripper

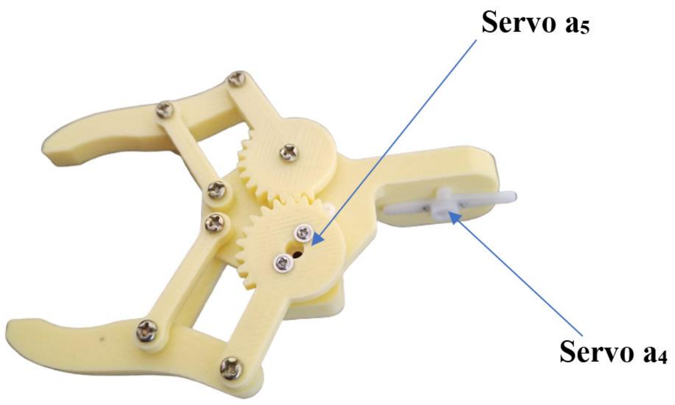


Fig. 10.3 Gripper

The servo a_3, a_4, a_5 are present in gripper. Servo a_3 is used for rotating the gripper clock and anticlockwise direction. This a_3 servo connects as a joint of robotic arm and the gripper. Servo a_4 is used to Up Down the gripper. When angle increases gripper goes up and when angle decreases gripper goes down. The servo a_4 has another feature that is this servo makes the gripper parallel to the ground, what ever the position of the robotic arm. Servo a_5 is used to close and open the gripper. When a_5 angle increases then gripper is opening, when a_5 angle decreases then gripper closing. The angles are assigned according to the object that going to pick.

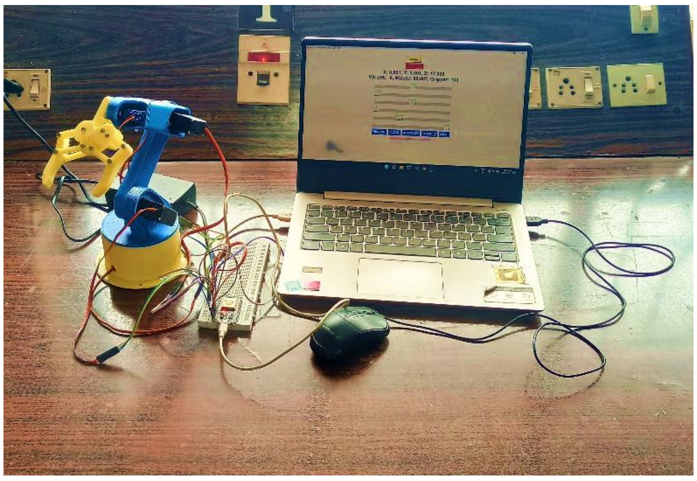


Fig. 10.4 Hardware Image 1

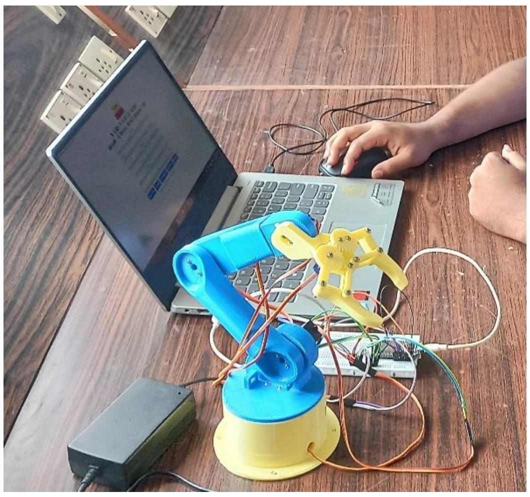


Fig. 10.5 Hardware Image 2

CHAPTER 11

CONCLUSION

The mathematical Inverse kinematics equation of our model gives the correct value of the angles for our robotic arm. The robotic arm is working as per the getting angles for the servo motors. End-effector of the robotic arm moves the desired location. When the record button is pressed the coordinate position is successfully recorded. When the automation button is pressed the robotic arm’s end-effecter is moving its correct coordinates as per the recorded coordinate positions. The robotic arm is successfully working in IoT. The robotic arm can be controlled by Mobile or Computer using IoT, the ESP32 successfully connected to the internet and getting data from the server, when the position is given through IoT the coordinates value is successfully changed in the cloud and ESP32 is getting the changed values of variables in IoT. Then those values are given to the robotic arm and robotic arm moves our desired position successfully. The recording of the robotic arm positions and the automation of the robotic arm through IoT works successfully.

REFERENCES

[1] Xi Jiang, Noah Apthorpe, “Automating Internet of Things Network Traffic Collection with Robotic Arm Interactions", Networking and Internet Architecture (cs.NI); Cryptography and Security (cs.CR); Robotics (cs.RO), 30 Sep 2021, doi:2110.00060.

[2] M. K. Ishak and N. M. Kit, "Design and Implementation of Robot Assisted Surgery Based on Internet of Things (IoT)," International Conference on Advanced Computing and Applications (ACOMP), 2017, pp. 65-70, doi: 10.1109/ACOMP.2017.20.

[3] S. Fu and P. C. Bhavsar, "Robotic Arm Control Based on Internet of Things," 2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT), 2019, pp. 1-6, doi: 10.1109/LISAT.2019.8817333.

[4] Victor H.Benitez, RodrigoSymonds, David E.Elguezabal, "Design of an affordable IoT open-source robot arm for online teaching of robotics courses during the pandemic contingency" , HardwareX, Volume 8, October 2020, e00158, doi: 10.1016/j.ohx.2020.e00158.

[5] N. K. Agrawal, V. K. Singh, V. S. Parmar, V. K. Sharma, D. Singh and M. Agrawal, "Design and Development of IoT based Robotic Arm by using Arduino," Fourth International Conference on Computing Methodologies and Communication (ICCMC), 2020, pp. 776-780, doi: 10.1109/ICCMC48092.2020.ICCMC-000144.

[6] K. Jahnavi and P. Sivraj, "Teaching and learning robotic arm model," 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), 2017, pp. 1570-1575, doi: 10.1109/ICICICT1.2017.8342804.

[7] Jegede Olawale, Awodele Oludele, Ajayi Ayodele, and Ndong Miko Alejandro Babcock University, Ilisan- Remo, Nigeria, "Development of a Microcontroller Based Robotic Arm", January 2007.

[8] Mohd Ashiq KamarilYusoff, Reza EzuanSamin, Babul Salam KaderIbrahim, "Wireless Mobile Robotic Arm", Volume 41, 2012, doi.org/10.1016/j.proeng.2012.07.285.

[9] Wan Muhamad Hanif WanKadir, Reza EzuanSamin, Babul Salam KaderIbrahim, "Internet Controlled Robotic Arm", 2012, doi.org/10.1016/j.proeng.2012.07.284.

[10] Jamshed Iqbal, Raza ul Islam, and Hamza Khan, "Modeling and Analysis of a 6 DOF Robotic Arm Manipulator", Canadian Journal on Electrical and Electronics Engineering Vol. 3, No. 6, July 2012.

[11] https://www.arduino.cc/reference/en/

[12] https://esp32io.com/tutorials/esp32-hello-world

[13] https://en.wikibooks.org/wiki/C%2B%2B_Language

[14] https://docs.platformio.org/en/stable/tutorials/espressif32/arduino_debugging_unit_testing.html