

ÜSKÜDAR UNIVERSITY
FACULTY OF ENGINEERING AND NATURAL SCIENCES



**INTELLIGENT ROBOTIC ARM CONTROLLED BY
MOBILE APP VIA BLUETOOTH**

LICENSE TERMINATION THESIS

MUHAMMED CAN

MACİT

160201032

Thesis Consultant: Dr. Öğr. Üyesi İhab ELAFF

07/2021

COMPUTER ENGINEERING DEPARTMENT

Thesis advisor: Dr. Öğr. Üyesi İhab ELAFF

Jury:

Dr. Öğr. Üyesi İhab ELAFF

Prof. Dr. Serhat Özekes

Doç.Dr. Kaan Yılancıoğlu

Submission Date: 29/06/2021

Thesis Defense: 04/07/2021

FOREWORD

The content of this thesis is a robotic arm that can be controlled with a mobile application and automatically catches an object. I would like to extend my endless thanks to my esteemed consultant Dr. Öğr. Üyesi İhab ELAFF , who shared his valuable knowledge with me in the realization of this study.

24.05.2021

MUHAMMED CAN MACİT

TABLE OF CONTENTS

FOREWORD	3
CONTENTS	4-5
SUMMARY	5
FIGURE LIST.....	6
EQUATION LIST	7
ABBREVIATIONS.....	7
1.INTRODUCTION.....	8
1.1. What is Robot	8
1.2 Early Conception of Robots.....	8-10
1.3. Application of Robotics	10-11-12
1.4 First Purpose of Thesis	13
1.5 Second Purpose of Thesis	13
2. MATERIALS AND METHODS.....	14
2.1 Explanation Robotic Arm Working Principles.....	14
2.2.Literature.....	15-24
3. SYTEM DESIGN AND IMPLEMENTATION	25
3.1 What we going to do?.....	25
3.2 General Properties of The Kinematics of The Robot Arm	25
3.3Coordianate Frames and Transformation Matrice Robot	25-28
3.4 Forward Kinematic	29
3.5Inverse Kinematic	29-30
3.6 Block Diagram.....	30
3.7 Our Kinematic Equation	31-32-33
3.1.1 Hardware Part.....	35-40
3.1.2 Software Part.....	40-54
4.CONCLUSION.....	55-57
REFERENCES	58-59
CV	60

SUMMARY

Today, the rapidly growing human needs in line with technology at the same rate is developing. The work carried out to meet these needs is increasing day by day and these studies are concentrated on robot arm studies. Robot arms can be used with an outside user or instead of predetermined commands. They work by bringing. Today, robot arms that are needed in almost every field are the most the field of development is industry and the medical sector. The robot arm designed and realized in the project has can move in 4 axis directions with 5 DC motors. The holder, it can take a desired material from one place and move it to another. While doing this process, stm32 and Bluetooth module connected to Arduino Robot control is provided by connecting to the application. In addition, with the catch button in the

application, it can catch an object that is not determined.

Abbreviations

ASCII: American Standard Code for Information Interchange

CS: C language part of code

APK: Android Packet

IPA: Instrument for Pre-accession Assistance

IIC: Inter-Integrated Circuit

SPI: Synchronous communication protocol

PC: Personal Computer

USB: Universal Serial Bus

VR: Virtual Realty

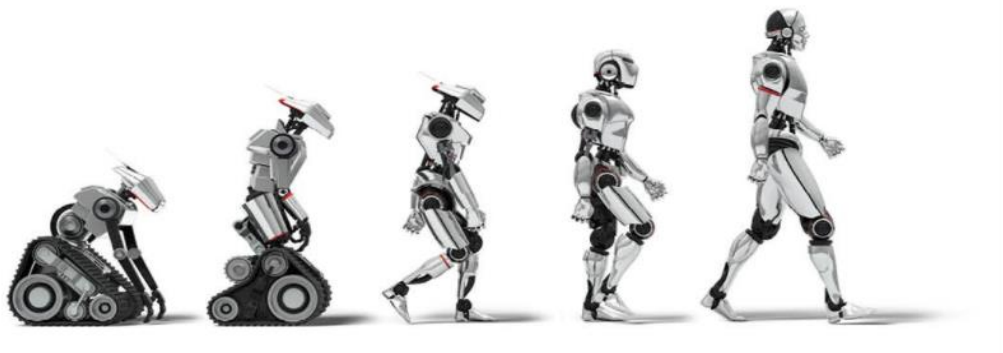
BC: Before Christ

CHAPTER 1

INTRODUCTION

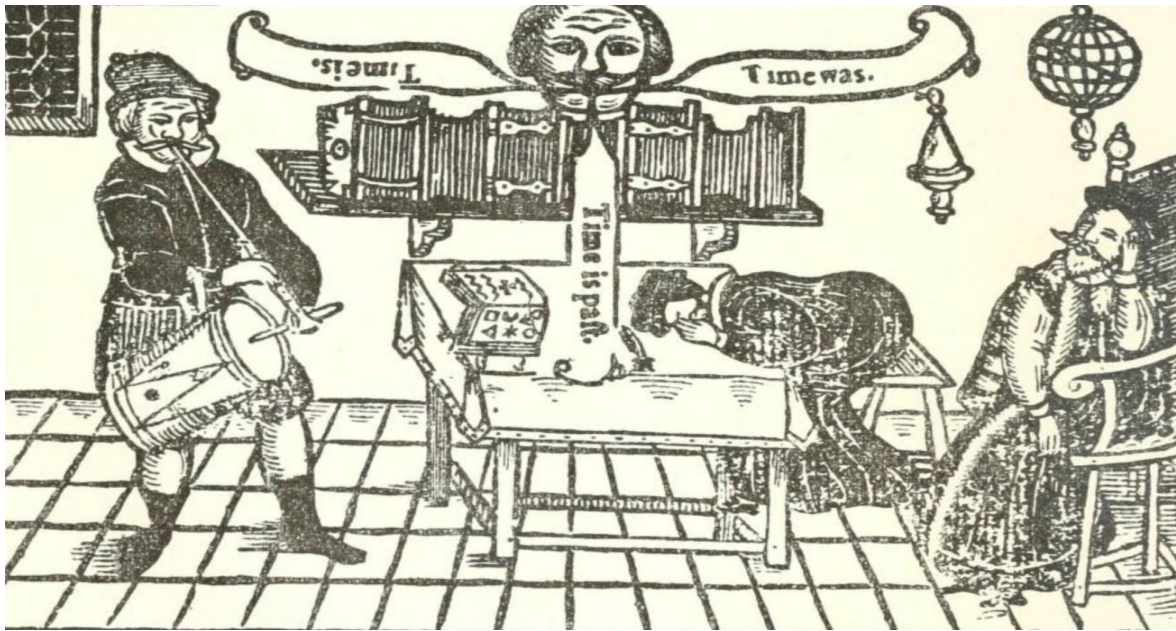
1.1 What is Robot?

A robot is a machine - especially one programmable by a computer - capable of carrying out a complex series of actions automatically. Robots can be guided by an external control device or the control may be embedded within. Robots may be constructed on the lines of human form, but most robots are machines designed to perform a task with no regard to their aesthetics.



1.2 Early Conception of Robots

One of the first instances of a mechanical device built to regularly carry out a particular physical task occurred around 3000 B.C.: Egyptian water clocks used human figurines to strike the hour bells. In 400 B.C., Archytus of Tarentum, inventor of the pulley and the screw, also invented a wooden pigeon that could fly. Hydraulically-operated statues that could speak, gesture, and prophecy were commonly constructed in Hellenic Egypt during the second century B.C.

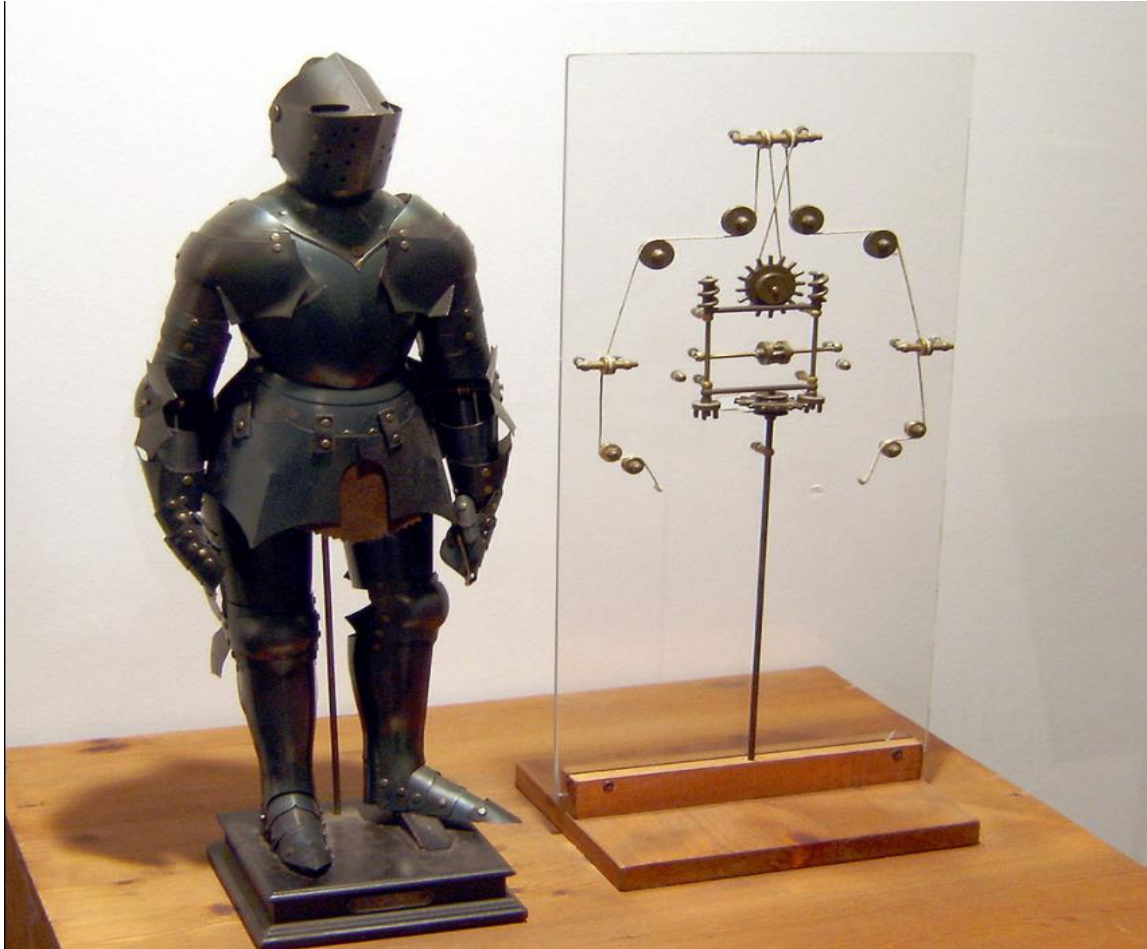


In the first century A.D., Petronius Arbiter made a doll that could move like a human being. Giovanni Torriani created a wooden robot that could fetch the Emperor's daily bread from the store in 1557. Robotic inventions reached a relative peak (before the 20th century) in the 1700s; countless ingenious, yet impractical, automata (i.e. robots) were created during this time period. The 19th century was also filled with new robotic creations, such as a talking doll by Edison and a steam-powered robot by Canadians. Although these inventions throughout history may have planted the first seeds of inspiration for the modern robot, the scientific progress made in the 20th century in the field of robotics surpass previous advancements a thousandfold.

The earliest robots as we know them were created in the early 1950s by George C. Devol, an inventor from Louisville, Kentucky. He invented and patented a reprogrammable manipulator called "Unimate," from "Universal Automation." For the next decade, he attempted to sell his product in the industry, but did not succeed. In the late 1960s, businessman/engineer Joseph Engleberger acquired Devol's robot patent and was able to modify it into an industrial robot and form a company called Unimation to produce and market the robots. For his efforts and successes, Engleberger is known in the industry as "the Father of Robotics."

Academia also made much progress in the creation new robots. In 1958 at the Stanford Research Institute, Charles Rosen led a research team in developing a robot called "Shakey." Shakey was far more advanced than the original Unimate, which was designed for specialized, industrial applications. Shakey could wheel around the room,

observe the scene with his television "eyes," move across unfamiliar surroundings, and to a certain degree, respond to his environment. He was given his name because of his wobbly and clattering movements.



The first industrial robot: UNIMATE



1954: The first programmable robot is designed by George Devol, who coins the term Universal Automation. He later shortens this to Unimation, which becomes the name of the first robot company (1962). UNIMATE originally automated the manufacture of TV picture tubes

*Also known the First Scientist who put the basics of Robotics Science is al-Jazari



1.3 Application of Robotics

-In the operating room

Surgeons have always needed steady hands, as even the slightest of movements could cause them to nick arteries, sever tissue, or even pierce organs. There are no fully autonomous robot surgeons, but several systems have been developed to augment human capabilities.

Some examples of surgical robotics applications include Preceyes' system for ocular procedures; Corindus Vascular Systems' CorPath system, which incorporates virtual reality (VR) for remote operations, and the Monarch robot from Auris Health.

-Law enforcement and emergency response

First responders, incident-response teams, and law enforcement officers increasingly rely on robots to go ahead of them in hazardous situations. The 2011 nuclear disaster in Fukushima, Japan, prompted advances in disaster-response robotics that continue to the present day.

Bomb-disposal robots, for example, are designed to assess threats and include a variety of sensors and remote monitoring tools. Similar robots can be used in emergencies navigating a burning building or a compromised structure after an earthquake.

-On the farm

Agriculture has always been subject to the vagaries of weather, soil conditions, pests, and the labor supply. Farming equipment has become much more sophisticated and can autonomously navigate, plant, weed, and harvest certain crops. Milking machines in dairies are just the beginning of robotics applications for livestock.

Additional robots handle duties like weed control, plant nursing and feeding, pollution monitoring, and even planting or seeding new crops. Demand has been high for robots to pick and move fruit, partly because of labor shortages.

-On the construction site

Robots applications in the construction and property development industries are being significantly enhanced thanks to technologies like AI, big data, and 3D printing.

Australia-based Fastbrick Robotics can even build brick houses four times faster than human workers. It combines a 3D printer and a robot that can lay bricks just as precisely as human laborers.

-On the battlefield

Military robotics is unusual in that all of the application described above also could help military personnel. In addition to ordnance-disposal systems and drones for forward observation, armed forces around the world are interested in portable and easy-to-use surgical systems and rapid construction.

1.4 Purpose of Thesis

Today people in the past have always need additional help system. Automation systems are needed to achieve this. People have felt the need to use auxiliary machines in places where they cannot afford due to their physical characteristics. Machines that previously needed manpower to work are now self-operating. One of the most used elements in automation systems applications robots. Robots help us everywhere in our lives. But we must not forget that we made them again.

1.5 Second Purpose of Thesis

In the project, the operations performed in the realization of the robot arm designed to perform the task determined in line with the predetermined commands and research to gain various knowledge about mechanics and software during these operations made and applied to the project. Robot arm consists of 5 DC motors. By programming the Stm32 microcontroller in the project, DC motor control is provided. For the robot to work, a 10 V – 12 V power supply has been preferred. It is aimed to move the robot in a controlled manner.

CHAPTER 2

Materials and Methods

2.1 Explanation Robotic Arm Working Principles

First, a historical research on robot arms was made and the basic information necessary for the establishment of the system was obtained. The robot arm used in the project is a joint type and 4 can move in the axis direction (left and right, up, and down) and the handle on it. it can hold and swing movements. Optimal control of the robot arm. Stm32 is the microcontroller used to provide it. The reason why this microcontroller is preferred is that it is open source, its use is 32bit and faster than another microcontroller. Detailed information about the DC motors to be used after these studies are carried out. information has been obtained. Correctly how the robot will do the project. the motor to be selected can work precisely and has high torque DC motor is preferred because it is required. Robot arm has 5 DC motors consists of. The robot arm made the appropriate Arduino microcontroller selected to be moved appropriately for its purpose software was developed with the software and then experiments with Bluetooth modules and DC motors information about the system operation was obtained. The software was implemented with the appropriate microcontroller selected to move the robot arm according to its purpose. Accordingly, commands are sent to the motors and they are directed to the right left or up and down. In addition, an object at a certain distance can be captured with the predetermined function.

2.2. Literature

2.2.1 Robotic Manipulator

In robotics, a manipulator is a device used to manipulate materials without direct physical contact by the operator. The applications were originally for dealing with radioactive or biohazardous materials, using robotic arms, or they were used in inaccessible places. In more recent developments they have been used in diverse range of applications including welding automation,[1] robotic surgery and in space. It is an arm-like mechanism that consists of a series of segments, usually sliding or jointed called cross-slides,[2] which grasp and move objects with a number of degrees of freedom.

In industrial ergonomics a manipulator is a lift-assist device used to help workers lift, maneuver and place articles in process that are too heavy, too hot, too large or otherwise too difficult for a single worker to manually handle. As opposed to simply vertical lift assists (cranes, hoists, etc.) manipulators have the ability to reach in to tight spaces and remove workpieces. A good example would be removing large stamped parts from a press and placing them in a rack or similar dunnage. In welding, a column boom manipulator is used to increase deposition rates, reduce human error and other costs in a manufacturing setting.

Additionally, manipulator tooling gives the lift assist the ability to pitch, roll, or spin the part for appropriate placement. An example would be removing a part from a press in the horizontal and then pitching it up for vertical placement in a rack or rolling a part over for exposing the back of the part.



2.2.2 Sliding Joint(Revolute)

A revolute joint (also called pin joint or hinge joint) is a one-degree-of-freedom kinematic pair used frequently in mechanisms and machines. The joint constrains the motion of two bodies to pure rotation along a common axis. The joint does not allow translation, or sliding linear motion, a constraint not shown in the diagram. Almost all assemblies of multiple moving bodies include revolute joints in their designs. Revolute joints are used in numerous applications such as door hinges, mechanisms, and other uni-axial rotation devices.

A revolute joint is usually made by a pin or knuckle joint, through a rotary bearing . It enforces a cylindrical contact area, which makes it a lower kinematic pair, also called a full joint. However, If there is any clearance between the pin and hole (as there must be for motion), so-called surface contact in the pin joint actually becomes line contact.

The contact between the inner and outer cylindrical surfaces is usually assumed to be frictionless. But some use simplified models assume linear viscous damping in the form $\{ \displaystyle T=B*\omega \}$, where T is the friction torque, ω is the relative angular velocity, and B is the friction constant. Some more complex models take stiction and stribeck effect into consideration.



2.2.3 Rotating Joint(Prismatic)

A prismatic joint provides a linear sliding movement between two bodies, and is often called a slider, as in the slider-crank linkage. A prismatic pair is also called a sliding pair. A prismatic joint can be formed with a polygonal cross-section to resist rotation. See for example the dovetail joint and linear bearings.


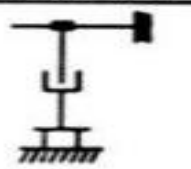
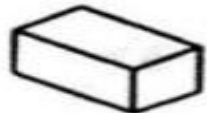
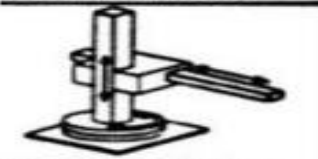
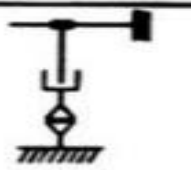


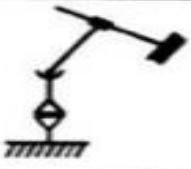


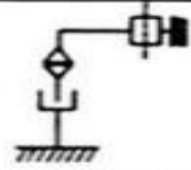


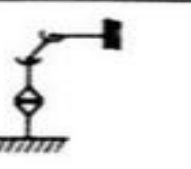

The relative position of two bodies connected by a prismatic joint is defined by the amount of linear slide of one relative to the other one. This one parameter movement identifies this joint as a one degree of freedom kinematic pair.

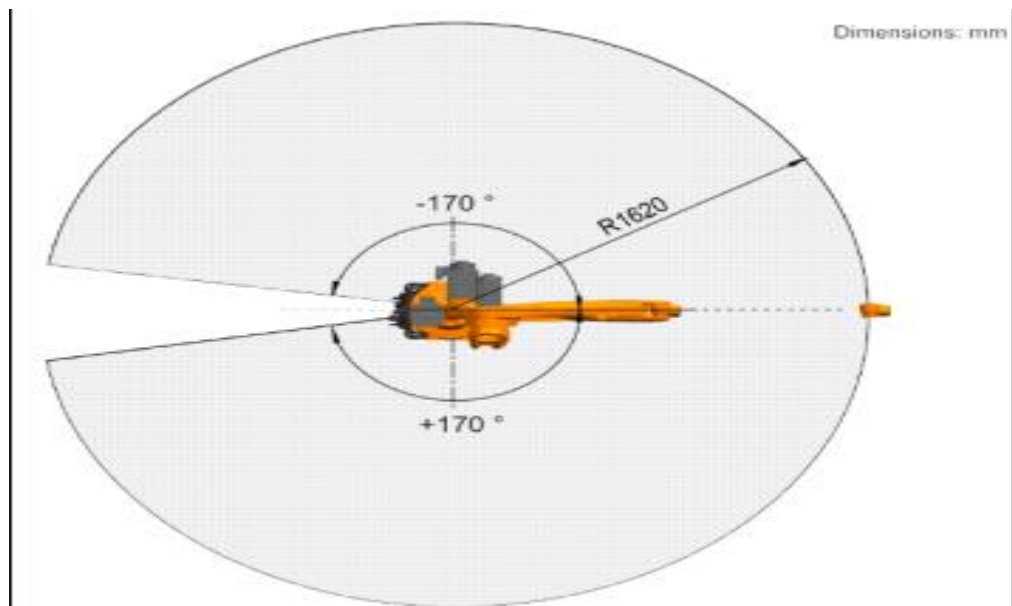
Prismatic joints provide single-axis sliding often found in hydraulic and pneumatic cylinders.



-Work Space

The workspace of a robot arm is the set of all positions that it can reach. This depends on a number of factors including the dimensions of the arm.

Principle	Kinematic Structure	Workspace
 Cartesian Robot		
 Cylindrical Robot		
 Spherical Robot		
 SCARA Robot		
 Articulated Robot		

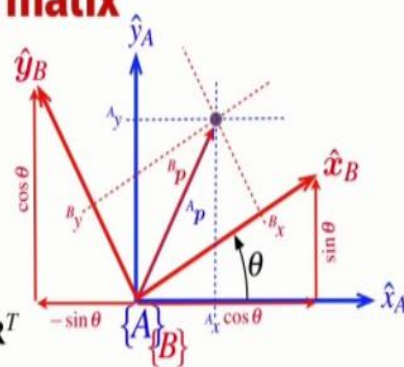


-General Rotation Matrix

Properties of the **rotation matrix**

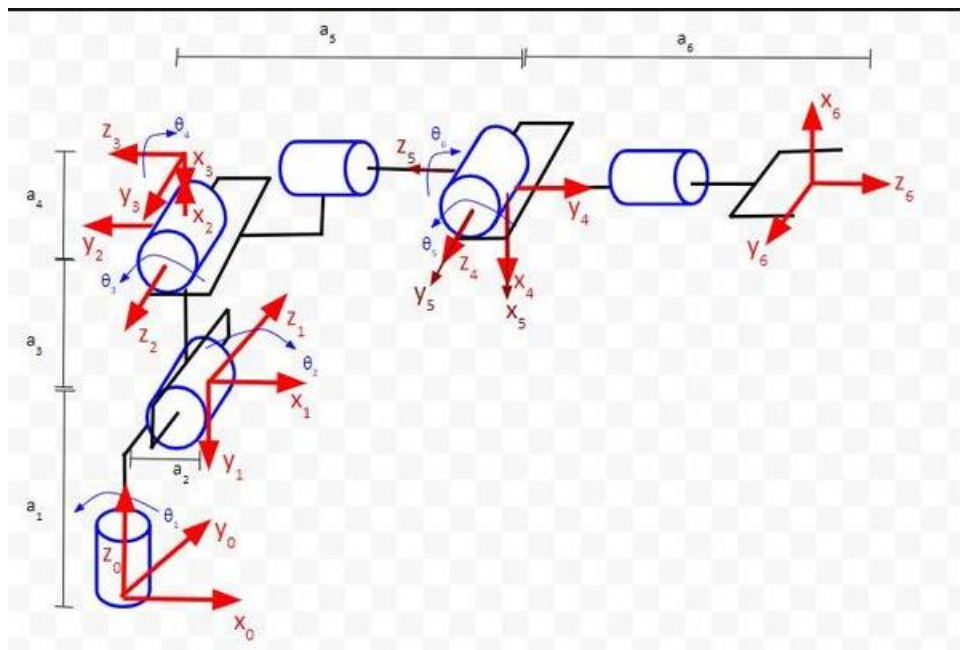
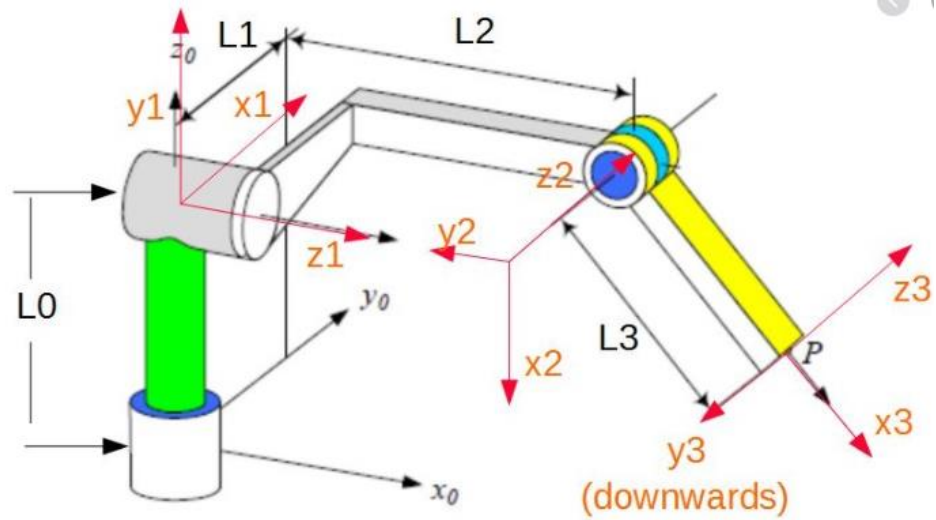
$${}^A\mathbf{R}_B = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

- An orthogonal (orthonormal) matrix
 - ➔ Each column is a unit length vector
 - ➔ Each column is orthogonal to all other columns
- The inverse is the same as the transpose $\mathbf{R}^{-1} = \mathbf{R}^T$
- The determinant is +1 $\det(\mathbf{R}) = 1$
 - ➔ the length of a vector is unchanged by rotation
- Rotation matrices belong to the Special Orthogonal group of dimension 2 $\mathbf{R} \in SO(2)$

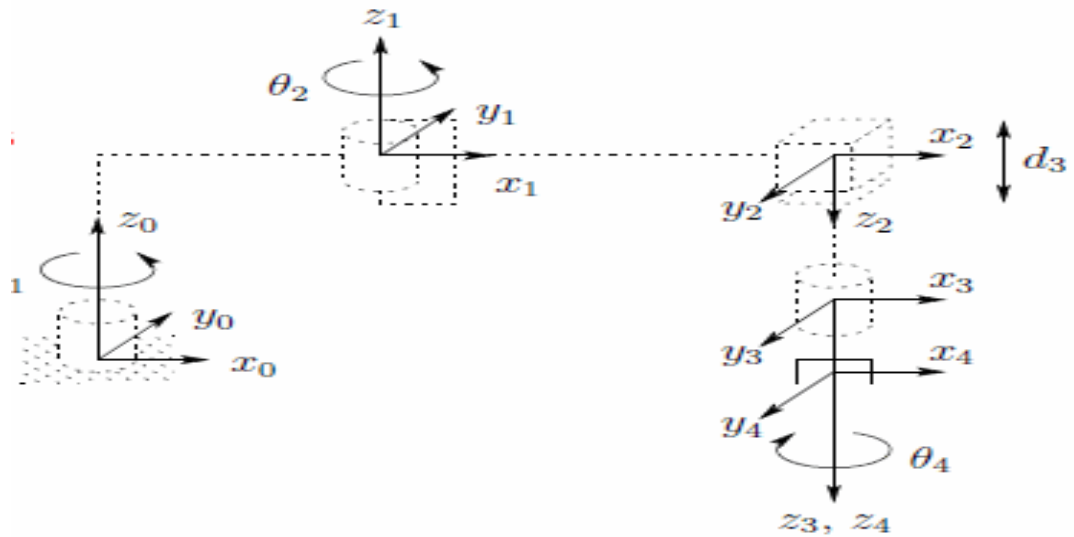


$$\mathbf{A} = \begin{pmatrix} \cos \theta_n & -\sin \theta_n \cos \alpha_n & \sin \theta_n \sin \alpha_n & l_n \cos \theta_n \\ \sin \theta_n & \cos \theta_n \cos \alpha_n & -\cos \theta_n \sin \alpha_n & l_n \sin \theta_n \\ 0 & \sin \alpha_n & \cos \alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

-Assigning coordinate frames for both sliding and rotating joints



2.2.4 Link Variable Table

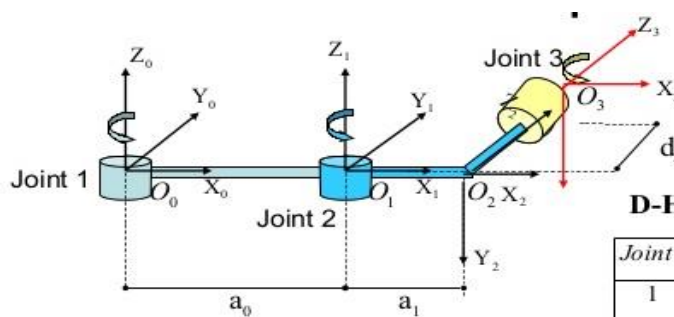


3.11 DH coordinate frame assignment for the SCARA manipulator

Table 3.5 Joint parameters for SCARA

Link	a_i	α_i	d_i	θ_i
1	a_1	0	0	θ^*
2	a_2	180	0	θ^*
3	0	0	d^*	0
4	0	0	d_4	θ^*

* joint variable



D-H Link Parameter Table

Joint i	α_i	a_i	d_i	θ_i
1	0	a_0	0	θ_0
2	-90	a_1	0	θ_1
3	0	0	d_2	θ_2

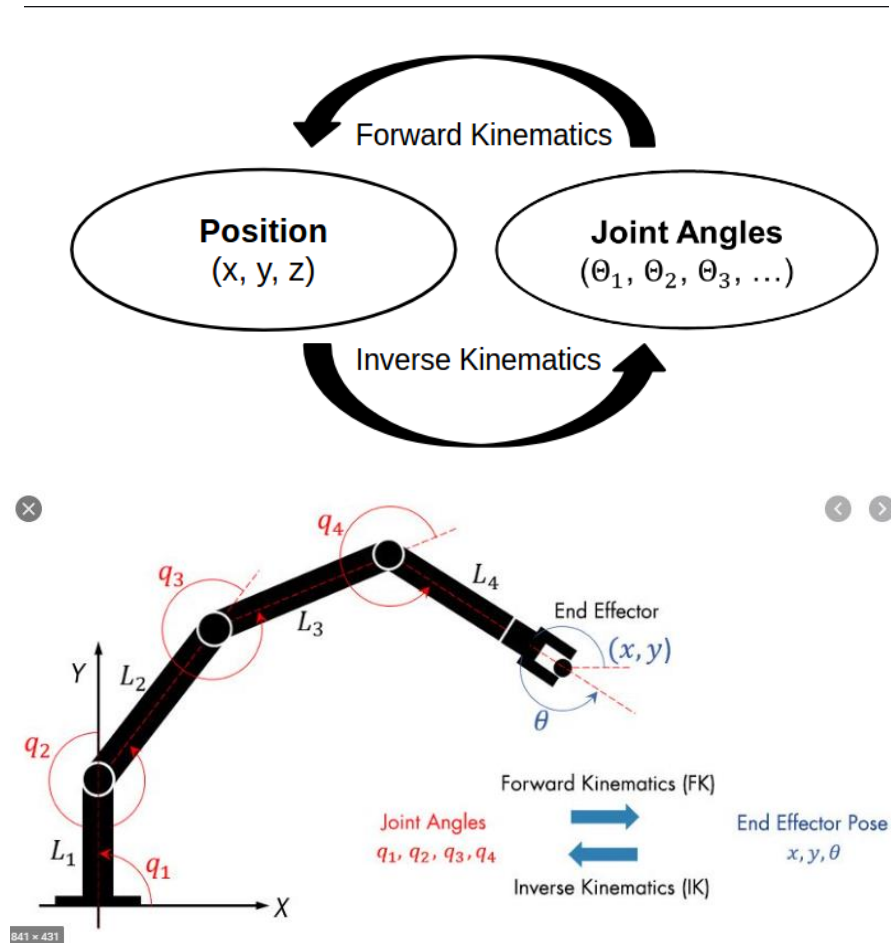
α_i : rotation angle from Z_{i-1} to Z_i about X_i

a_i : distance from intersection of Z_{i-1} & X_i to origin of i coordinate along X_i

d_i : distance from origin of $(i-1)$ coordinate to intersection of Z_{i-1} & X_i along Z_{i-1}

θ_i : rotation angle from X_{i-1} to X_i about Z_{i-1}

2.2.5 Forward and Inverse Kinematics



2.2.6 Ultrasound Sensor Operations

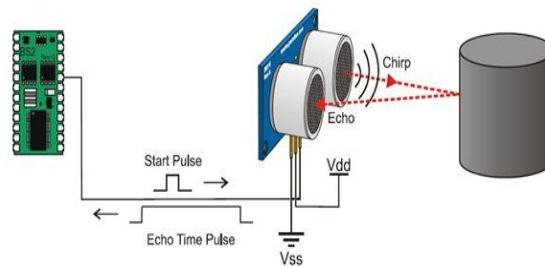
An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e. the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).

In order to calculate the distance between the sensor and the object, the sensor measures the time it takes between the emission of the sound by the transmitter to its contact with the receiver. The formula for this calculation is $D = \frac{1}{2} T \times C$ (where D is the distance, T is the time, and C is the speed of sound ~ 343 meters/second). For example, if a scientist set up an ultrasonic sensor aimed at a box and it took

0.025 seconds for the sound to bounce back, the distance between the ultrasonic sensor and the box would be:

$$D = 0.5 \times 0.025 \times 343$$

or about 4.2875 meters.



2.2.7 Circuit Diagram

Below is the theoretical connection of all the tools we use. Outputs from dc motors are connected to In298. Power input is given through the board. Power inputs of HCSR-04 and HC-06 are provided by stm32. The power supply of the Stm32 can be provided by the computer or by connecting it to the board.

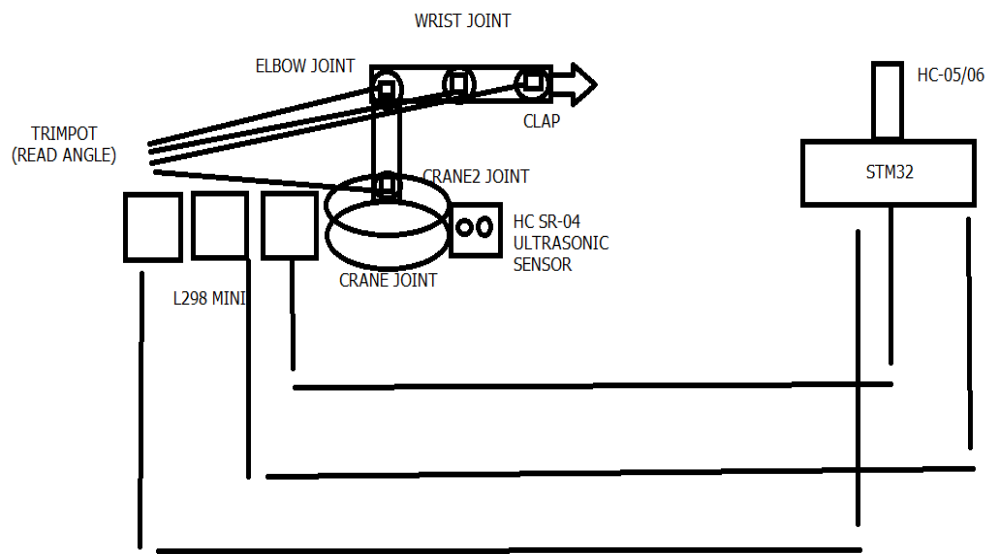


Figure 6 Circuit Diagram

CHAPTER 3

System Design and Implementation

3.1 What we going to do?

The aim of this project is to provide user control to the robot arm by using the necessary protocols and algorithms. For this, we first perform forward and backward kinematic operations. We reconcile the resulting matrices with the angles. After performing the necessary mathematical operations, we apply these operations to the hardware. With the mobile application we have created, we communicate with the bluetooth sensor that is active and send and receive data. Thanks to these data packets, we can move the joint of the robot arm that we want. In addition, we detect the distance of the object with the distance sensor and perform the necessary automatic operations. They are l298 mini robot drivers that move the joints of the robot arm. To summarize, to be able to control the arms by giving commands to the l298 mini robot drives according to the data received by the Bluetooth sensor from the mobile application, and also to give automatic instructions to the robot arm and hold the objects with the help of the distance sensor according to the type of data.

3.2 General Properties of the Kinematics of the Robot Arm

General Properties of the Kinematics of the Robot Arm. Robot arm links can be rotated or delayed relative to the reference coordinate frame. Robot kinematics is the subject of robotics that studies multi-degree of freedom kinematic chains that form the structure of robotic systems using geometry. The emphasis on geometry arises from the fact that the robot's parts are modeled as solid bodies and its joints have only rotation or translation.

Robot kinematics examines the relationships between the dimensions and connectivity of the kinematic chains and the position, velocity, and acceleration of each part in the robotic system to plan and control the movement to be made and to calculate the forces and torques of the actuators. Mass and inertia properties, the relationships between motion and

related forces and torques are studied under the topic of robot dynamics. General transformation matrix is quite complicated even for simple robots can be.

3.3 Coordinate Frames and Transformation Matrices for a General Robot Arm

Homogeneous representation of an n-dimensional position vector with an n + 1 dimensional vector called the coordinate representation. Below is a position vector between coordinate frames A 4 '4' matrix is seen showing in homogeneous coordinates (Fu, 1987).

$${}^R\mathbf{T}_H = \left[\begin{array}{c|c} \text{dönme} & \text{öteleme} \\ \hline (3*3) & (3*1) \\ \hline 0 & 1 \\ \hline (1*3) & (1*1) \end{array} \right] = \left[\begin{array}{cccc} x_x & y_x & z_x & p_x \\ x_y & y_y & z_y & p_y \\ x_z & y_z & z_z & p_z \\ 0 & 0 & 0 & 1 \end{array} \right]$$

Equation 1 Rotation and Translation

$p = p_i + p_j + p_k$ vector of the origin of the new frame,
 $x = x_i + x_j + x_k$ the direction vector of the x-axis of the new frame,
 $y = y_i + y_j + y_k$ the direction vector of the y-axis of the new frame,
 $z = z_i + z_j + z_k$ represents the direction vector of the z-axis of the new frame.

The 4th column of the transformation matrix has 3 elements corresponding to the translation in the x, y, and z directions.

$$\text{Trans}(p_x, p_y, p_z) = \left[\begin{array}{cccc} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{array} \right]$$

Equation 2 Trans Vector

Since rotation is possible in any of the 3 coordinate axes, there are 3 rotational transforms corresponding to rotations of angle θ in the x, y, and z axes. The following matrix can be written for the x-axis.

$$\mathbf{Rot}(x, \theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation 3 X Rotation

Matrices representing only rotations about the y and z axes can be written similarly.

$$\mathbf{Rot}(y, \theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{Rot}(z, \theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation 4 Rot y and Rot z

Elements of the transformation matrix can be found by the cascade product of pure rotation and translation matrices. When the orientation of the end point with respect to the reference frame in Cartesian space is desired, this can be achieved as a sequence of rotations about the axes of the fixed reference frame. While there are many ways to do this, one of the best known is the roll-pitch-yaw transformation. It is by determining 3 rotations. Rotation about the x-axis first, then the rotation about the y-axis and then the z-axis.

$$\text{RPY}(\phi, \theta, \Psi) = \text{Rot}(z, \phi) \text{Rot}(y, \theta) \text{Rot}(x, \Psi)$$

$$= \begin{bmatrix} C(\phi)C(\theta) & C(\phi)S(\theta)S(\Psi) - S(\phi)C(\Psi) & C(\phi)S(\theta)C(\Psi) + S(\phi)S(\Psi) & 0 \\ S(\phi)C(\theta) & S(\phi)S(\theta)S(\Psi) + C(\phi)C(\Psi) & S(\phi)S(\theta)C(\Psi) - C(\phi)S(\Psi) & 0 \\ -S(\theta) & C(\theta)S(\Psi) & C(\theta)C(\Psi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation 5 Calculation

In robotics, kinematics is the study of motion. It examines the relationships between the positions, velocities, and accelerations of the robot arm limbs, without considering forces and other factors that affect motion.

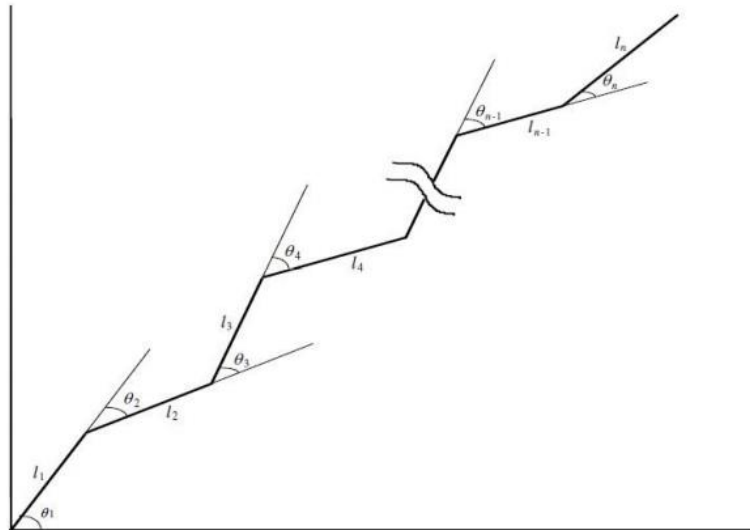


Figure 7 Robot arm showing relative angles

When examining the serial limb robot arm given in Figure7, when the position and

orientation of one of the limbs or both changes, the positions, and orientations of the limbs between this limb and its end point will also change. As the position and orientation of the limb's changes, it is often desirable to determine the position and orientation of the endpoint with respect to the basic reference frame.

3.4 Forward Kinematic

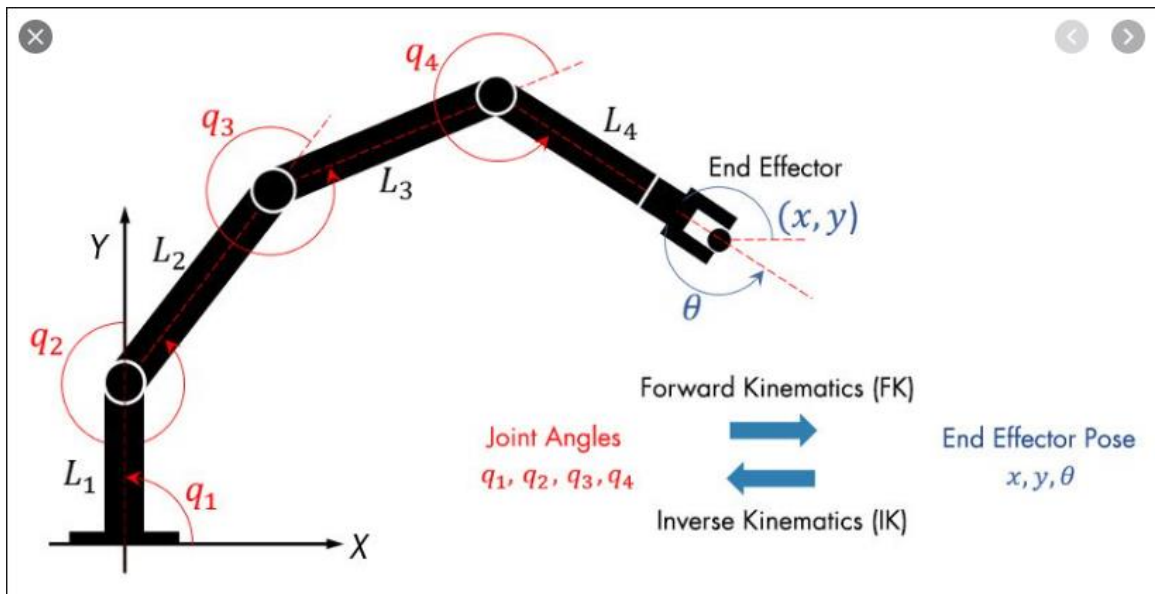
The articulation space definition of a robot arm can be related to the definition of Cartesian space. That is, the position and orientation of the endpoint for a given set of joint variables can be determined in Cartesian coordinates. This process is known as forward kinematics. Advanced kinematics is the event of finding the R T H transformation matrix that gives the position and orientation of the endpoint using the A matrices we mentioned earlier. The transformation matrix RTH can also be obtained by multiplying the general translation matrix by the orientation transform matrix of the RPY angles.

$${}^R\mathbf{T}_H = \begin{bmatrix} x_x & y_x & z_x & p_x \\ x_y & y_y & z_y & p_y \\ x_z & y_z & z_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

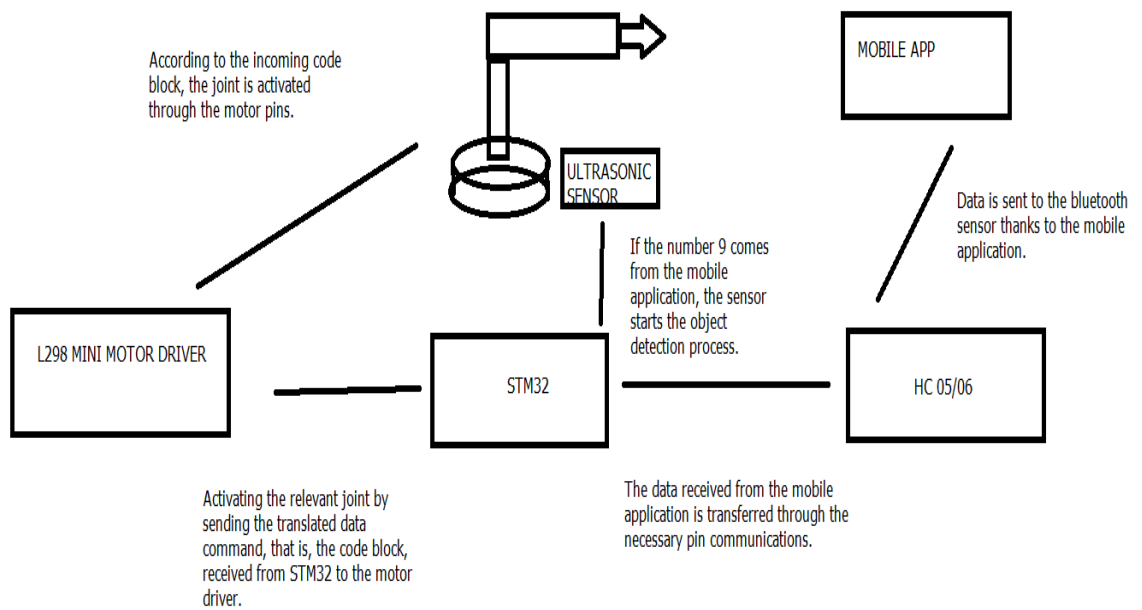
Figure 8 RTH

3.5 Inverse Kinematic

The process of finding the values that hinge variables should take for a given endpoint position and orientation in Cartesian coordinates is called inverse kinematics. Inverse kinematics can be quite difficult to analyze compared to forward kinematics. In many cases, it is necessary to use techniques that do not guarantee a solution and include trial and error.



3.6 BLOCK DIAGRAM



Kinematic Equations We Used in The Project

3.7. Forward and Inverse Kinematics Link Variable Table And Coordinate Frames

3.7.1 Link Variable Table

Firstly, we create the link variable table.

#	Variable	θ	l	d	α
1	θ_1	θ_1	0	d_1	90
1	θ_2	θ_2	l_2	0	0

3.7.2 A Matrices

Calculate the A matrices

$$\begin{array}{cc}
 A_1 & A_2 \\
 \begin{bmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} C_2 & -S_2 & 0 & l_2 c_2 \\ S_2 & C_2 & 0 & l_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
 \end{array}$$

3.7.3. Rth

$$\begin{bmatrix} c_1c_2 & -c_1s_2 & s_1 & l_2c_1c_2 \\ s_1c_2 & -s_1s_2 & -c_1 & l_2s_1c_2 \\ s_2 & c_2 & 0 & l_2s_2 + d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$p_x = l_2c_1c_2 \quad p_y = l_2s_1c_2 \quad p_z = l_2s_2 + d_1$$

$$\Theta_1 = \tan^{-1}(p_y/p_x) \quad \Theta_2 = \cos^{-1}(p_x/(l_2c_1))$$

$$p_x = (l_2 + d_3)c_1c_2 \quad p_y = (l_2 + d_3)s_1c_2 \quad p_z = (l_2 + d_3)s_2 + d_1$$

$$\Theta = \tan^{-1}(p_y/p_x) \quad \Theta = \tan^{-1}(s_1(p_z - d_1)/p_y) \quad d_3 = ((p_z - d_1) \div s_2) - l_2$$

3.1.1 Hardware Parts

3.1.2 Definition of DC Motor

A direct current (DC) motor is a type of electric machine that converts electrical energy into mechanical energy. DC motors take electrical power through direct current and convert this energy into mechanical rotation. DC motors use magnetic fields that occur from the electrical currents generated, which powers the movement of a rotor fixed within the output shaft. The output torque and speed depend upon both the electrical input and the design of the motor. A magnetic field arises in the air gap when the field coil of the DC motor is energized. The created magnetic field is in the direction of the radii of the armature. The magnetic field enters the armature from the North pole side of the field coil and “exits” the armature from the field coil’s South pole side. When kept in a magnetic field, a current-carrying conductor gains torque and develops a tendency to move. In short, when electric fields and magnetic fields interact, a mechanical force arises. This is the principle on which the DC motors work.

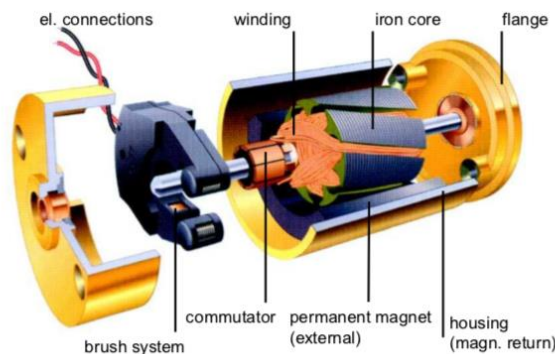


Figure 0 Dc Motor

3.1.3 Definition of Trimer Capacitor

It is an adjustable power button. Provides the movement of a motor of the robot by giving a specific value or angle or an instruction. It is attached to the joint of the second motor of our robot. Controlling of this, aimed that the robot will take an action after a certain second. Ease of use is aimed by adding the robot's motor that moves on the y axis. It has 3 legs. These go to power grounding and stm32. It gives instructions to the motor by encoding it via Stm32 and the movement is provided.



Figure 2: Trimer Capacitor

3.1.4 Definition of HC O6

HC-06 BT module (Figure3) is a module that works only as a slave and uses serial communication protocol. In Bluetooth communication, master and slave are determined according to the status of starting the connection. A master module can initiate the connection, but the slave module cannot initiate the connection. Since we will provide the connection of the module that works as a slave with a pc or android device in our project, an external device will initiate the connection. Bidirectional data in a healthy way can be sent and received. It provides the connection between our application and stm32 and exchanges data.



Figure 3: HC 06

3.1.5 Definition of Stm32

It basically consists of a microprocessor, ram memory, flash memory where the program to be run will be saved, and many interfaces depending on the model. Interfaces are programmable electronic structures. All STM32 series microcontrollers have IIC and SPI communication interfaces. While programming, the pins are set to 0/1 state. It is connected to the pc with the help of the USB, and it must return to the 1/0 state again to operate. We preferred this in our study because it is both fast and small.

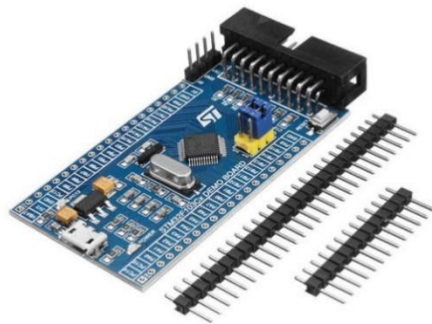


Figure 4: Stm32

3.1.6 Definition of L298N Motor Driver

The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A. Using the L298N is straightforward. If you want the left motor to rotate in one direction, apply a high pulse to IN1 and a low pulse to IN2. To reverse the direction, reverse the pulses to IN1 and IN2. In our study, we connected 2 motors to them. In Figure 5 We made the power connection using 10V with the help of the board. We pulled the parts from the motor a and b to the stm32 from the motor outputs. The reasons I mentioned above helped me choose this in our study.

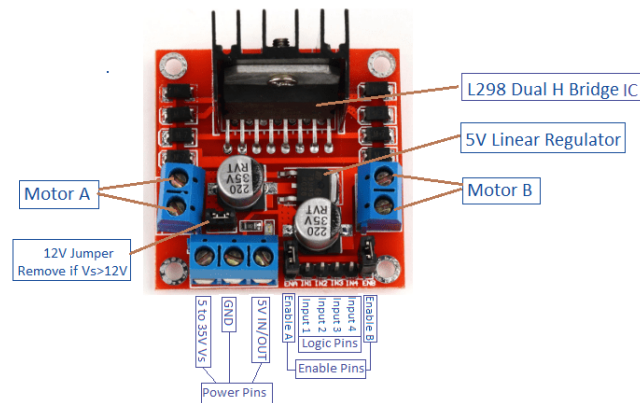


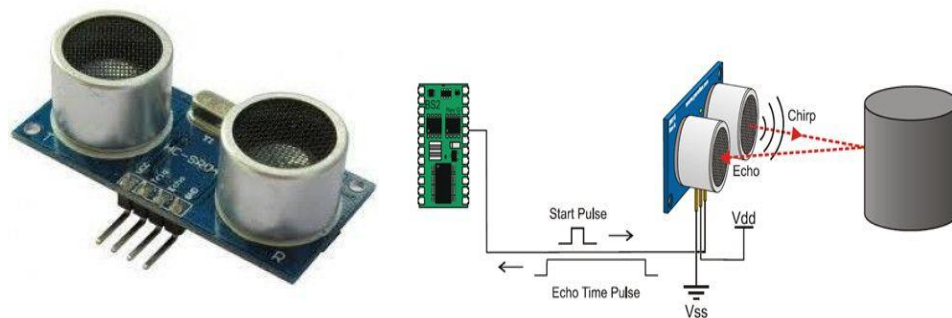
Figure: 5 L298N

3.1.7 Ultrasonic Sensor

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e. the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target). In order to calculate the distance between the sensor and the object, the sensor measures the time it takes between the emission of the sound by the transmitter to its contact with the receiver. The formula for this calculation is $D = \frac{1}{2} T \times C$ (where D is the distance, T is the time, and C is the speed of sound ~ 343 meters/second). For example, if a scientist set up an ultrasonic sensor aimed at a box and it took 0.025 seconds for the sound to bounce back, the distance between the ultrasonic sensor and the box would be:

$$D = 0.5 \times 0.025 \times 343$$

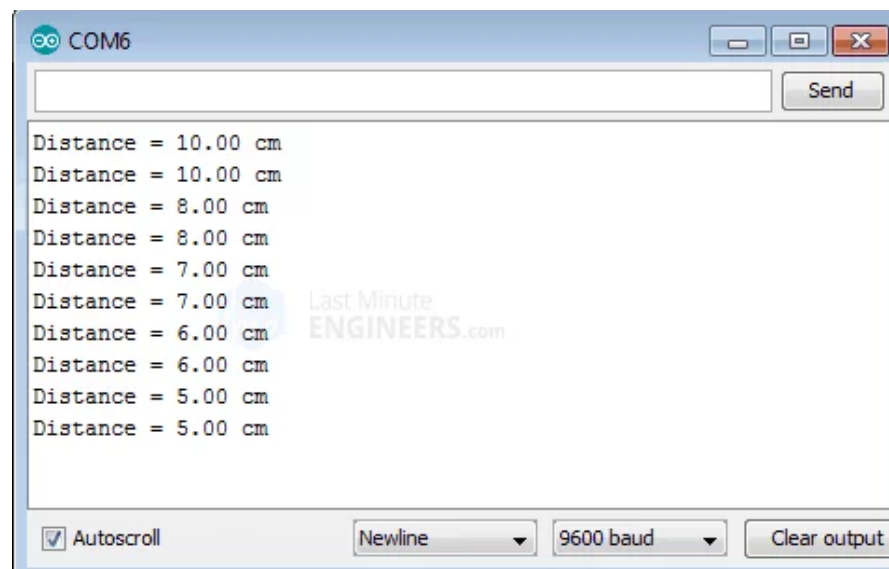
or about 4.2875 meters.



3.2 SOFTWARE PART

3.2.1 Ultrasonic Part of Robot

```
int calculateDistanceFront() {  
    digitalWrite(trigPinFront, LOW);  
    delayMicroseconds(2);  
  
    digitalWrite(trigPinFront, HIGH);  
    delayMicroseconds(1000);  
    digitalWrite(trigPinFront, LOW);  
    durationFront = pulseIn(echoPinFront, HIGH);  
    distanceFront = (durationFront / 2) / 28.5;  
    return distanceFront;  
}
```



3.2.2 Xamarin

The application development platform I have used in this project. Xamarin is an open-source platform for building modern and performance applications for .NET and iOS, Android, and Windows. Xamarin applications can be written on a PC or Mac, and a. APK file. It can be compiled into native application packages such as IPA file. It can share code and business logic between platforms. Cross-platform applications can be written in C # language.

3.2.3 Mobile App

First, we have a Bluetooth connected screen. Here we used the standard Bluetooth connection process. We performed the data sending and receiving via the Bluetooth module with this procedure (Figure 3).

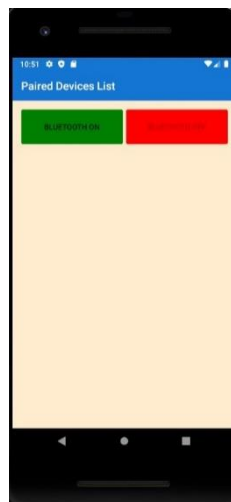


Figure 9: Paired List

We went for a simple design as a fronthand. The user can easily turn on the bluetooth and connect. When the connection is established, it automatically switches to the other page.

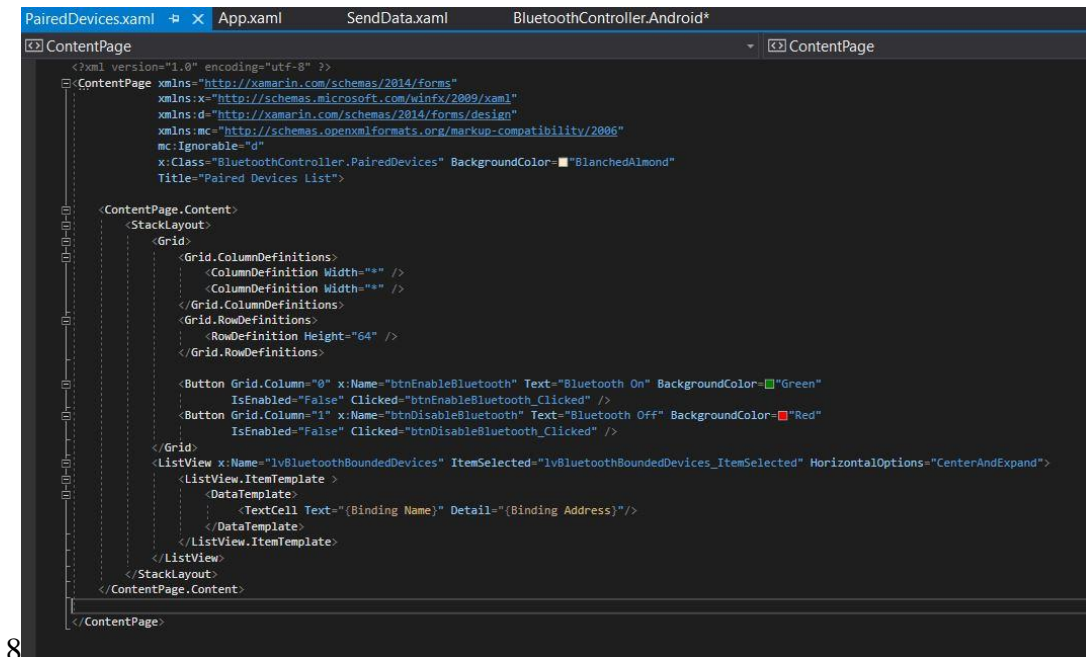


Figure 10: Paired Device Design

I used 2 clickable buttons as a design (Figure7). When we turn on Bluetooth, the devices are listed, and we call them. In the background of the other page, we made our communication by ensuring that the values we sent go to the numbers corresponding to the ASCII. That is why when we send 1 from our phone, we introduced our application as 49, as shown in the figures (Figure11).

```

btnClampClose.Clicked += delegate
{
    try
    {
        App.CurrentBluetoothConnection.Transmit(new Memory<byte>(new byte[1] { 50 }));
    }
    catch (Exception)
    {
    }
};

btnClampJointUp.Clicked += delegate
{
    try
    {
        App.CurrentBluetoothConnection.Transmit(new Memory<byte>(new byte[1] { 51 }));
    }
    catch (Exception)
    {
    }
};

btnClampJointDown.Clicked += delegate
{
    try
    {
        App.CurrentBluetoothConnection.Transmit(new Memory<byte>(new byte[1] { 52 }));
    }
    catch (Exception)
    {
    }
};

```

Figure 11: ASCII

3.2.4 Robotic Arm Controlling Page

We connected the connections of Dc Engines with the app via stm32. We filled in the functional articles through the application. Some motors move in right and left orbit, that is, 180 on the x-axis or 180 on the y-axis.

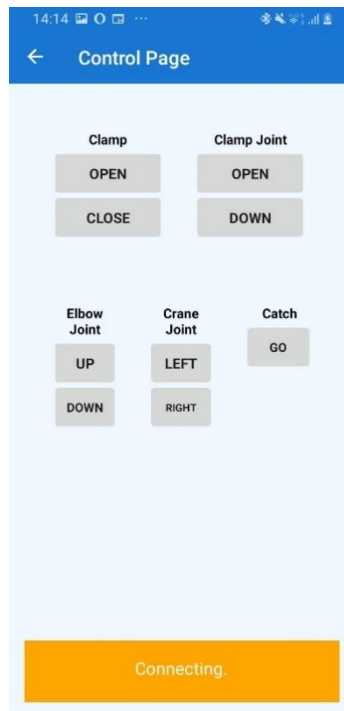


Figure 12: Control Page

There are 4 DC motor commands on this page. The Clamp has 2 functions, as can be seen from their names. These are for opening and closing the clamp. Clamp joint cannot open and close because it is responsible for the robot's maneuver. Again, we can send 2 commands, these are up and down. Our other engine is connected to the Elbow joint. Its maneuver direction is again up and down. Our 4th engine is Crane Joint. Its maneuver is left and right, not up and down. The user sends numbers by pressing these commands and the motors perform the command as we code in the Arduino. Our last button is the Catch button. If this

button finds an object using the distance sensor, it captures the object and releases it to the position we previously determined.

3.2.5 Code Part

```
<StackLayout>

  <StackLayout Orientation="Horizontal">

    <StackLayout Orientation="Vertical">
      <Label Text="Clamp" TextColor="Black" WidthRequest="110" FontAttributes="Bold" HorizontalTextAlignment="Center" VerticalTextAlignment="Center"></Label>
      <Button x:Name="btnClampOpen" Text="OPEN" FontSize="15" FontAttributes="Bold" WidthRequest="40" HeightRequest="40"></Button>
      <Button x:Name="btnClampClose" Text="CLOSE" FontSize="15" FontAttributes="Bold" WidthRequest="40" HeightRequest="40"></Button>
    </StackLayout>

    <StackLayout Orientation="Vertical">
      <Label Text="Clamp Joint" TextColor="Black" WidthRequest="110" FontAttributes="Bold" HorizontalTextAlignment="Center" VerticalTextAlignment="Center"></Label>
      <Button x:Name="btnClampJointUp" Text="OPEN" FontSize="15" FontAttributes="Bold" WidthRequest="40" HeightRequest="40"></Button>
      <Button x:Name="btnClampJointDown" Text="DOWN" FontSize="15" FontAttributes="Bold" WidthRequest="40" HeightRequest="40"></Button>
    </StackLayout>
  </StackLayout>
```

Figure 13: Sent Data

This is the front hand of the above user menu (Figure13). A separate label and button name have been created for each button. The motors and their processes are specially named. This figure contains only Clamp and Clamp Join. It consists of a label and 2 buttons as shown. Others are below (Figure14).

```

<StackLayout Orientation="Horizontal">

    <StackLayout Orientation="Vertical" >
        <Label Text="Elbow Joint" TextColor="Black" WidthRequest="160" FontAttributes="Bold" HorizontalTextAlignment="Center" VerticalTextAlignment="Center"></Label>
        <Button x:Name="btnElbowJointUp" Text="UP" FontSize="15" FontAttributes="Bold" WidthRequest="50" HeightRequest="40" ></Button>
        <Button x:Name="btnElbowJointDown" Text="DOWN" FontSize="13" FontAttributes="Bold" WidthRequest="50" HeightRequest="40" ></Button>
    </StackLayout>

    <StackLayout Orientation="Vertical" >
        <Label Text="Crane Joint" TextColor="Black" WidthRequest="160" FontAttributes="Bold" HorizontalTextAlignment="Center" VerticalTextAlignment="Center"></Label>
        <Button x:Name="btnCraneJointLeft" Text="LEFT" FontSize="15" FontAttributes="Bold" WidthRequest="50" HeightRequest="40" ></Button>
        <Button x:Name="btnCraneJointRight" Text="RIGHT" FontSize="11" FontAttributes="Bold" WidthRequest="50" HeightRequest="40" ></Button>
    </StackLayout>

    <StackLayout Orientation="Vertical" >
        <Label Text=" Catch" FontSize="14" TextColor="Black" WidthRequest="180" FontAttributes="Bold" HorizontalTextAlignment="Center" VerticalTextAlignment="Center" ></Label>
        <Button x:Name="ExtraJointUp" Text="Go" FontSize="13" FontAttributes="Bold" WidthRequest="50" HeightRequest="40" ></Button>
    </StackLayout>

</StackLayout>

```

Figure 14: Sent Data 2

```

private void CurrentBluetoothConnection_OnStateChanged(object sender, StateChangedEventArgs stateChangedEventArgs)
{
    var model = (DigitViewModel)BindingContext;
    if (model != null)
    {
        model.ConnectionState = stateChangedEventArgs.ConnectionState;
    }
}

2 bapvnu
private void CurrentBluetoothConnection_OnReceived(object sender, Plugin.BluetoothClassic.Abstractions.ReceivedEventArgs recievedEventArgs)
{
    DigitViewModel model = (DigitViewModel)BindingContext;
    if (model != null)
    {
        model.SetReceiving();
        for (int index = 0; index < recievedEventArgs.Buffer.Length; index++)
        {
            byte value = recievedEventArgs.Buffer.ToArray()[index];
            model.Digit = value;
        }
        model.SetReceived();
    }
}

2 bapvnu
private void CurrentBluetoothConnection_OnError(object sender, System.Threading.ThreadExceptionEventArgs errorEventArgs)
{
    if (errorEventArgs.Exception is BluetoothDataTransferUnitException)
    {
    }
}

```

Figure 15: Cs Part of Sent Data

We write and send our data into the index in the form of an array and turn off data reception. By constantly checking during the connection, if there is a change, the connection status changes (Figure15). Error and disconnection occur because it cannot be sent when there is no connection. If there is a connection, data is sent.

```

2 bapxuru
public ConnectionState ConnectionState
{
    get
    {
        return _connectionState;
    }
    set
    {
        if (_connectionState != value)
        {
            _connectionState = value;
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("ConnectionState"));
            UpdateConnectionStateBackgroundColor();
        }
    }
}

2 bapxuru
private void UpdateConnectionStateBackgroundColor()
{
    switch (ConnectionState)
    {
        case ConnectionState.Created:
        case ConnectionState.Initializing:
        case ConnectionState.Connecting:
            ConnectionStateBackgroundColor = Color.Orange;
            break;
        case ConnectionState.Connected:
            ConnectionStateBackgroundColor = Color.SeaGreen;
            break;
        case ConnectionState.ErrorOccured:
        case ConnectionState.Reconnecting:
            ConnectionStateBackgroundColor = Color.Red;
            break;
        default:
            ConnectionStateBackgroundColor = Color.Black;
            break;
    }
}

```

Figure 16: ModelViewClass

We can check connection state from figure 16 . Here we change the color of the button according to the connection status.

```

using Plugin.BluetoothClassic.Abstractions;
using System;
using System.ComponentModel;
using System.Drawing;

namespace BluetoothController.ViewModel
{
    6 bayyuru
    public class DigitViewModel : INotifyPropertyChanged
    {
        0 bayyuru
        public enum Properties
        {
            Digit,
            ConnectionState
        }

        public const byte DigitDefault = 0;
        private byte _digit = DigitDefault;
        private ConnectionState _connectionState;
        private Color _connectionStateBackgroundColor;

        0 bayyuru
        public DigitViewModel()
        {
            SetRecived();
            UpdateConnectionStateBackgroundColor();
        }

        public event PropertyChangedEventHandler PropertyChanged;

        2 bayyuru
        internal bool Reciving { get; set; }

        1 bayyuru
        public byte Digit
        {
            get
            {
                return _digit;
            }
            set
            {
                if (_digit != value)
                {
                    _digit = value;

                    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("Digit"));
                }
            }
        }
    }
}

```

Figure 17: ModelViewClass2

The necessary definitions for adjusting the part in Figure 13 are given here.(Figure17)

3.2.6 Arduino Programming Language

Arduino needs coding to function structurally. In the simplest way, it is necessary to write a code to an Arduino. However, since it is an easy and effective interface with Arduino, these operations are performed efficiently and conveniently. Arduino programming code base is C ++. This language is a source for many languages today. Languages such as C # are derived from these languages and their use is quite common today.

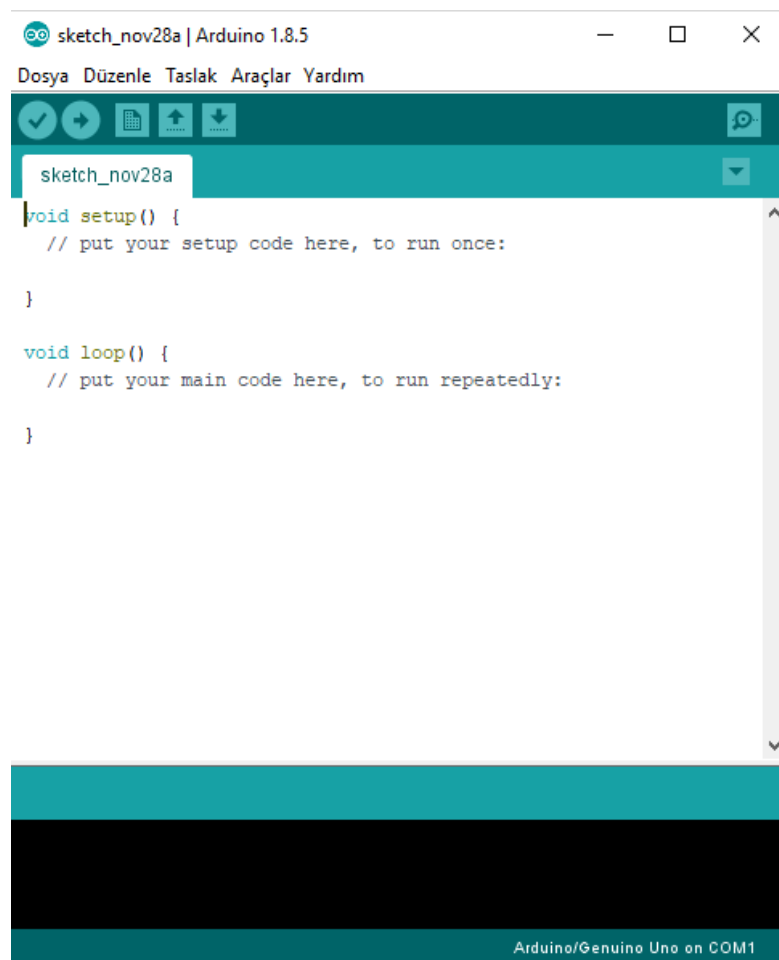


Figure 18: Arduino Code Screen

Click button allows the code to be compiled. And it shows mistakes. The right direction sign compiles the code and loads it into the card beside it. Before that, we must select the correct port and recognize the card. Document button allows us to open a new document. The last button saves our work. (Figure18) In addition, we can print the reading data of the distance sensor we use in our project on the screen via the Arduino application (Figure 19).

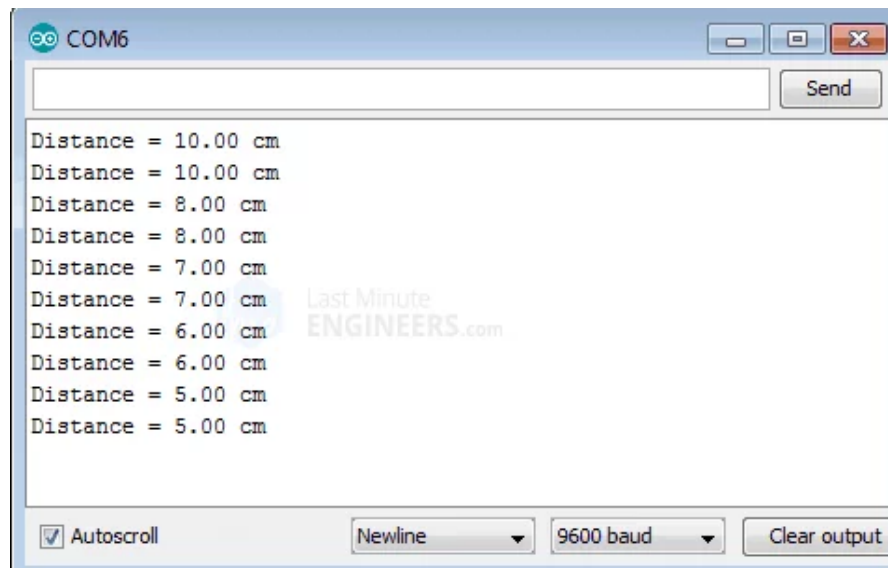


Figure 19: Distance Screen on Arduino

3.2.7 Coding Part of Arduino

```
1 #include <DC_Motor.h>
2 DC_Motor myDCMotor;
3
4 int data;|
5 const int in1=PA7;
6 const int in2=PA6;
7 const int in3=PA5;
8 const int in4=PA4;
9 const int in21=PA3;
10 const int in22=PA2;
11 const int in23=PA1;
12 const int in24=PA0;
13 int const trimPin = PB12;
14 int trimVal;
15 int angle;
16 int trigPinFront = 2; //HC0-6
17 int echoPinFront = 3;
18 long durationFront;
19 int distanceFront;
20 int df;
21 int data;
```

Figure 20: Arduino Code1

In the first line, we activate the libraries we use. Then we introduce the inputs and outputs of our engines. We are introducing the distance sensor we use. We introduce the trim. We are introducing the data we will use in the distance sensor.

```
void setup() {

    Serial.begin(9600);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);
    pinMode(in21, OUTPUT);
    pinMode(in22, OUTPUT);
    pinMode(in23, OUTPUT);
    pinMode(in24, OUTPUT);
    myDCMotor.attach(in1);
    myDCMotor.attach(in2);

}
```

Figure 21: Arduino Code2

Void setup is a function and in the new Arduino it will appear in the new counter without you typing. This function works only once when we turn on power to the card and leaves its turn to the loop function. If we press the reset button, that is, if we cut the power and turn it back on, it works again. we introduced the outputs in Figure21.

```
void loop() {

    if(Serial.available())
    {
        data = Serial.read();

        trimVal = analogRead(trimPin);
        Serial.print("trimVal : ");
        Serial.print(trimVal);
        angle = map(trimVal,0,1023,0,180); // 0 1023 value between 0-180 angle can move
        Serial.print(", angle: ");
        Serial.println(angle);

        delay(2500);
        myDCMotor.write(angle);

        df=calculateDistanceFront();
        Serial.print("front: ");
        Serial.println(df);

    }

    void olc() {
        delay(1000);
    }
}
```

Figure 22: Arduino Code 3

This is the part to get the data from the data. It is also the part that we add trim that allows the motor to move and take the gap. The four values next to the angle are what values can be taken and what can be used.

```
int calculateDistanceFront() {
    digitalWrite(trigPinFront, LOW);
    delayMicroseconds(2);

    digitalWrite(trigPinFront, HIGH);
    delayMicroseconds(1000);
    digitalWrite(trigPinFront, LOW);
    durationFront = pulseIn(echoPinFront, HIGH);
    distanceFront= (durationFront / 2) / 28.5;
    return distanceFront;
}
```

Figure 23: Arduino Code 4

At this stage, we use the distance sensor. The working principle of the distance sensor is to send and receive sound waves and to calculate the time in between. The reason in our region of 28.5 e is the friction coefficient of the air. With this function, the distance of the object is measured (Figure23).

```
void turnLeft(int timeout){
    analogWrite(PAin1,255);
    delay(timeout);
    analogWrite(PAin1,0);
    delay(50);
}
void turnRight(int timeout){
    analogWrite(PAin2,255);
    delay(timeout);
    analogWrite(PAin2,0);
    delay(50);
}
void bodyBackward(int timeout){
    analogWrite(PAin3,255);
    delay(timeout);
    analogWrite(PAin3,0);
    delay(50);
}
```

Figure 24: Arduino Code 5

We defined a function for our motors to move, we assigned 255 to the motor to turn left, and it progressed at a power of 255. Likewise, we did this to all engines (Figure19).

```
void grabAndDrag() {
    bodyUp(80);
    olc();
    bodyForward(500);
    olc();
    squizeItem(650);
    olc();
    bodyBackward(830);
    olc();
    bodyDown(80);
    olc();
    turnLeft(500);
    olc();

    bodyUp(80);
    olc();
    bodyForward(500);
    olc();
    releaseItem(350);
    olc();
}
```

Figure 25: Arduino Code 6

We have created one function by calling the engine functions we have created by putting certain values and delays for automatic capture here (Figure25).

```
if (data=='0')
{
  digitalWrite(Pin1,LOW);
  digitalWrite(Pin2,LOW);
  digitalWrite(Pin3,LOW);
  digitalWrite(Pin4,LOW);
  digitalWrite(Pin21,LOW);
  digitalWrite(Pin22,LOW);
  digitalWrite(Pin23,LOW);
  digitalWrite(Pin24,LOW);
}

if (data=='1')
{
  bodyUp();
  Serial.print(" angle: ");
  Serial.println(angle);
}

if (data=='2')
{
  bodyDown();
  Serial.print(" angle: ");
  Serial.println(angle);
}
```

Figure 26: Arduino Code 7

In this section, when the data is 0, it is set to disable the function. If we send commands from our phone, the functions we created will start working. Since we wrote the equivalent in ASCII language, we sent the correct data. (Figure26)

```

if (data=='7')
{

releaseItem();
}

if (data=='8')
{

squeezeItem();
}
if (data=='9')
{
    for(int i = 0;i<1000000000000000;i++){
        df=calculateDistanceFront();
        Serial.print("front: ");
        Serial.println(df);
        if(df>30 ){
            digitalWrite(Pin1,0);
            digitalWrite(Pin2,180);
            delay(500);
            digitalWrite(Pin1,0);
            digitalWrite(Pin2,0);
            delay(1000);
            digitalWrite(Pin1,180);
            digitalWrite(Pin2,0);
            delay(500);
            digitalWrite(Pin1,0);
            digitalWrite(Pin2,0);
            delay(1000);
        } else {
            grabAndDrag();
        }
    }
}
}

```

Figure 27: Arduino Code 8

In the last part of the code, there is an automatic capture section after all individual control is completed. With the distance sensor, we created a scan area until we find an object. When it sees the object, it stops and the grabanddrag function works (Figure27). Grabanddrag is name of function.

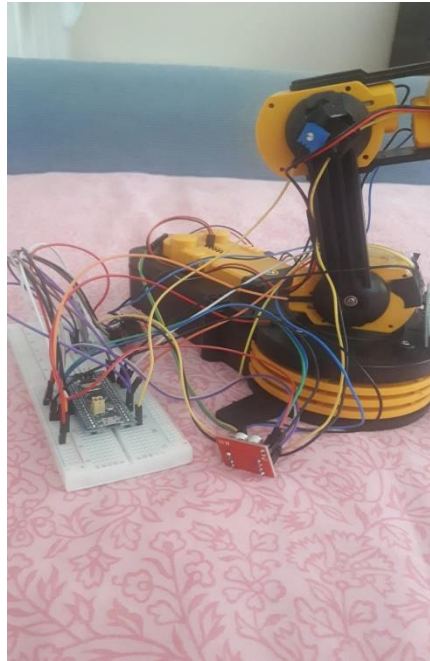
CHAPTER 4

CONCLUSION

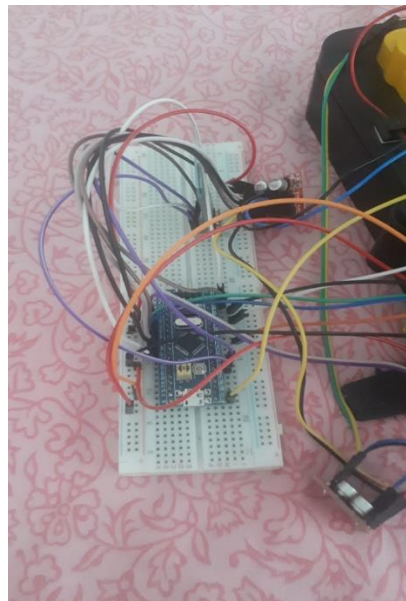
4.1 Conclusion

Robot arms can be developed in many areas. The robot arms, many works have been made easier and the error level that may arise has been minimized. For example, the robot we made is used with more sensitive motors and better equipped versions are used in the field of industry. Today there are even robots that assist in surgery. Cargoes distributed and classified without the need for manpower. Especially in the medical sector, there is no room for error. Humans make mistakes, but robots do not. With the robot arms developed in this way, the risk of infection of the patient in the medical sector is minimized, while human-induced errors during surgical intervention are minimized. Although the robot arm made with this project is a prototype, it can be developed for more comprehensive robotic systems. In addition, the robot arm sector, which is open to development, will maintain its importance in the future.

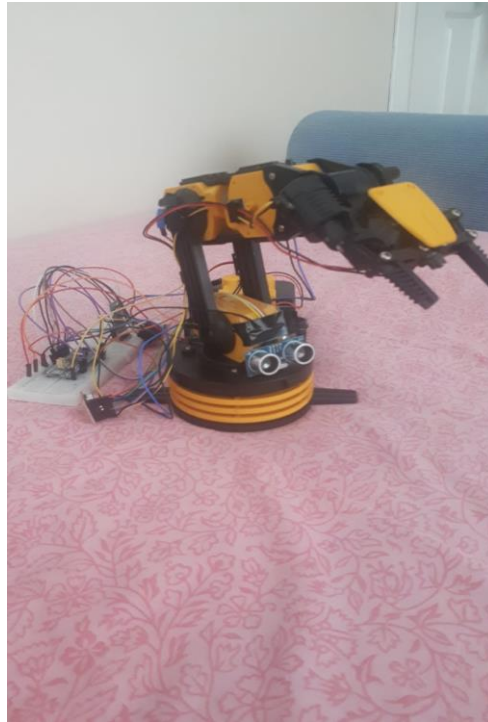
The aim of the project is to design a robot arm that moves in 4 axes and to control this robot arm with a suitable microcontroller and an application via a Bluetooth module. The necessary theoretical and practical knowledge for this purpose has been obtained and the necessary infrastructure has been established for the project. During the development and development of the project, by transferring the theoretical knowledge to practice, it is ensured that the project is realized in accordance with its purpose.



In Figure we can see the Trim sensor. With this, it can obtain the angle of the robot joint. In addition, we can make the robot move by giving value with our hands.



In Figure, the inputs of the outputs from the motors, bluetooth sensor, distance sensor to the stm32 are displayed. It works by connecting the power to the board.



In this picture, the final state is seen. The Bluetooth module and the distance sensor are in working condition. It is ready to be controlled by remote app. We can either run the motors sequentially and control them ourselves by sending data whenever we want. Or we can put the object in automatic mode and capture the object.

REFERENCES

- [1] Forward and Inverse Kinematic Analysis of Robots: Industrial Robotics
- [2] <https://users.cs.duke.edu/~brd/Teaching/Bio/asmb/current/Papers/chap3-forward-kinematics.pdf>
- [3] <https://www.uipath.com/rpa/robotic-process-automation>
- [4] <https://gelecegiyazanlar.turkcell.com.tr/konu/arduino/egitim/arduino-201/bluetooth-ile-iletisim>
- [5] https://en.wikipedia.org/wiki/DC_motor
- [6] https://www.tutorialspoint.com/arduino/arduino_dc_motor.htm
- [7] <https://github.com/stm32duino>
- [8] <https://www.stm32duino.com/>
- [9] <https://github.com/arduino/Arduino>
- [10] <https://github.com/Martinsos/arduino-lib-hc-sr04>
- [11] <https://www.instructables.com/Tutorial-for-Dual-Channel-DC-Motor-Driver-Board-PW/>
- [12] <https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-06-bluetooth-module-tutorial/>
- [13] <https://www.arduino.cc/en/Guide/ArduinoUno>
- [14] <https://eepower.com/resistor-guide/resistor-types/trimpot/#>
- [15] <https://dotnet.microsoft.com/apps/xamarin>
- [16] <https://www.tutorialspoint.com/xamarin/index.htm>
- [17] <https://dotnet.microsoft.com/learn/xamarin/hello-world-tutorial/intro>
- [18] <https://dotnet.microsoft.com/learn/xamarin>
- [19] <https://github.com/xamarin/XamarinForms>

- [20] <https://docs.microsoft.com/en-us/answers/questions/278526/usb-amp-bluetooth-connection-in-xamarinforms-code.html>
- [21] <https://docs.microsoft.com/en-us/answers/questions/278526/usb-amp-bluetooth-connection-in-xamarinforms-code.html>
- [22] <https://stackoverflow.com/questions/60984917/xamarin-android-bluetooth-connection>
- [23] <https://stackoverflow.com/questions/53353374/xamarin-pass-data-to-another-page>
- [24] Paden, Bradley Evan
<https://ui.adsabs.harvard.edu/abs/1985PhDT.....94P/abstract>
- [25] An Algorithm for Real-Time Forward Kinematics of Cable-Driven Parallel Robots
- [26] M.L. Husty, S. Mielczarek, and M. Hiller. A redundant spatial Stewart–Gough platform with maximal forward and inverse kinematics solution set. In *Advances in Robot Kinematics: Theory and Applications*, J. Lenarčič and F. Thomas (Eds.), Kluwer Academic Publishers, pp. 147–154 (2002).
- [27] Uskudar University Intelligent Robotic Lecture Notes
- [28] Uskudar University Mobile Programming Lecture Notes

