ROBOT ARM CONTROL WITH ARDUINO

Technical Report · June 2017

DOI: 10.13140/RG.2.2.10227.53286

Abdellatif Baba
Türk Hava Kurumu Üniversitesi

TÜRK HAVA KURUMU ÜNİVERSİTESİ

HAVACILIK ve UZAY BİLİMLERİ

T.C.

TÜRK HAVA KURUMU ÜNİVERSİTESİ REKTÖRLÜĞÜ TRANSKRİPT

ROBOT ARM CONTROL WITH ARDUINO

GRADUATION PROJECT REPORT SUBMITTED TO DEPARTMENT OF MECHANICAL AND AERONAUTICAL ENGINEERING

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS THE DEGREE OF MASTER OF SCIENCE WITHOUT THESIS MECHANICAL AND AERONAUTICAL ENGINEERING

BY

Aimn Mohamed Ahmed Ghiet

PROJECT ADVISOR: Dr. Abdellatif Baba

Spring 2017

ROBOT ARM CONTROL WITH ARDUINO

Dr. Abdellatif Baba

Aimn Mohamed Ahmed Ghiet

STUDENT ID NUMBER: 1403730048

Spring 2017

ABSTRACT

Today, technology is developing in the same direction in line with rapidly increasing human needs. The work done to meet these needs makes life easier every day, and these studies are concentrated in robotic arm studies. Robot arms work with an outside user or by performing predetermined commands. Nowadays, the most developed field of robot arms in every field is the industry and medicine sector.

Designed and realized in the project, the robot arm has the ability to move in 4 axis directions with 5 servo motors. Thanks to the holder, you can take the desired material from one place and carry it to another place, and also mix it with the material it receives. While doing this, robot control is provided by connecting to the android application via Bluetooth module connected to Arduino Nano microcontroller.

Table of Contents:

CHAPTERS

List of Figures

List of Tables

1. INTRODUCTION

These days' people always needed additional help systems. With the rapid increase in the flow of information, people are now guided to search for different markets and people have entered the competition to manufacture quality products cheaply. Automation systems are also needed to realize this. Because standardized automation systems are required to minimize errors as well as to have experienced and well-trained employees for quality products. Because of their physical characteristics, people needed to use auxiliary machines in places where their strength was not enough. These machines, which are operated with the need for human assistance in advance, have been made to operate spontaneously without the need of human power with the progress of technology. One of the most used components of automation systems is robots. Robotic systems; Mechatronics Engineering, Mechanical Engineering, Electrical Engineering and Computer Engineering have all come together to work together.

In the project, researchers have been done and implemented in order to have knowledge about mechanics and software during the operations carried out by the robot arm which is designed to fulfill the tasks determined in accordance with predetermined commands.

First, it was determined what function the robot arm would be and what movements it could make. Robotic arm made of Android phone or tablet control; it can carry the desired material, mix it up and perform the commands previously determined by a user. If this project is also a designated task; the robotic arm takes a piece of material and brings it to the desired position and then records its movements and lets it do the same action until we stop it. The servomotor is preferred in order to be able to perform these operations properly since the motor to be selected must operate precisely and must be at high torque. The robot arm is composed of 5 servo motors and can move in 4 axis directions with these motors.

In the project, Arduino Nano microcontroller written in Java language is programmed and servo motor control is provided. Thus, it is possible to perform the desired operations by means of the elements located on the Arduino without any circuit construction other than the circuit where the servo motor inputs are located. For the mechanical part, the robot arm is drawn with the SolidWorks program and the dimensions of the robot arm are specified. A 5V power supply is also preferred for the robot to work.

2. THEORETICAL INFRASTRUCTURE

The theoretical background of the project is examined below as main headings and subheadings.

2.1.Servo Motors

Servo; Detects the operation error of a mechanism, provides feedback and corrects faults. The servo motor can have alternating current (AC), direct current (DC) or stepper motors. In addition to these, there are drive and control circuits. Servo motors are the kinds of motors that can fulfill the commands we want. They can operate steadily even at very small or very large speeds. In these motors, the large moment can be obtained from the small size.

Servo motors are used in control systems such as fast operation, excessive axis movement, condition control and so on. Servo motors are the last control element of a mechanism. They are highly sensitive and servo motors are used in conjunction with electronic or programmable circuits. These engines are divided into AC and DC. When the AC servo motors are brushless type motors, the servo motors brush. Servo motors are mostly three cables. These are a red cable for power, black for grounding and yellow cables for control (data, data). One of the servomotors used in the production phase of the project is shown in Fig.1.

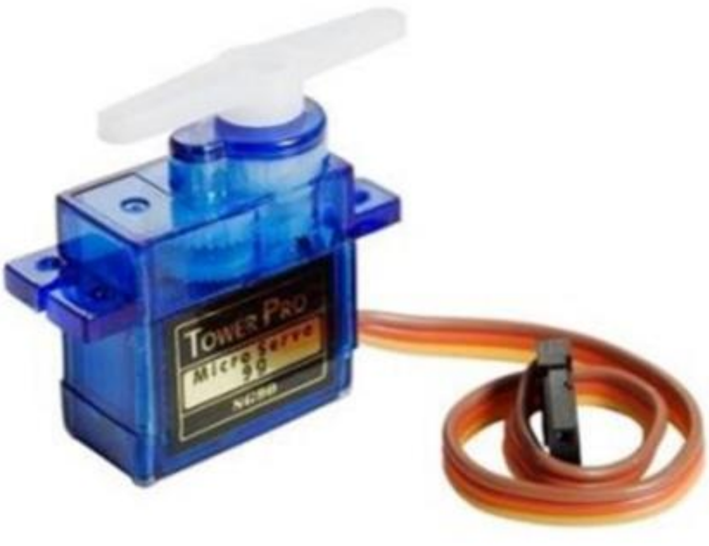


Figure 1DC servo motor

In the project Tower, Pro SG90 Mini servo motor is used. Some features of this servo motor; versatile operation, 10 µs pulse width control, V_{P-P}: 3-5 V

Square wave and working voltage of 4.8-6V. The used servomotor has a working voltage of 0.12 s / 60 ° and a torque of 1.2-1.6 kg/cm at low operating voltage.

Servo motors are controlled according to the signal condition. In doing so, the supplied pulse width modulated (PWM) signal is used with the data bus. Each servo motor is controlled by a PWM signal at 10-20 ms and at 0.5-1.5 ms.the position of the motor shaft is determined according to the duration (tk) of this signal at logic 1. These;

- When t_k = 0.5 ms, the motor shaft rotates to the end,
- When t_k= 0,5- 1 ms, the position of the motor shaft is in the middle,
- When t_k= 1 - 1.5 ms, the motor shaft turns to the right,
- When t_k= 10-20 ms (when the same signal is given again) it remains in its old position,

The position control of these motors is determined using the required pulses. The servo motors DC used in the project are kept at about 5V during operation.

2.2.Arduino Nano Microcontroller

Although microcontroller type PIC is usually used in programming and software field, Arduino has become very popular in the world in recent times. It is based on Arduino's past wiring and processing projects. Processing is written for non-programming users. Arduino wiring is produced on the basis of the programming language. The common feature of both is that it provides an environment where even the basic knowledge of electronics and programming can easily design. Arduino is now becoming more and more common nowadays. Even unmanned aerial vehicles made with Arduino, which is used almost every field, are visible.

The causes of the spread of Arduino at such a rapid rate are;

- It can be used on all platforms due to the simplicity of the development environment with driver usage.

- With the help of the advanced library, even complex operations can be easily solved.
- Programs written in Arduino can run fast because they are not run on any other platform.
- There is a lot of hardware support that is compatible with Arduino and can work together.
- Communication with the environment is easy because it is open source.
- If there are any problems due to a large number of Arduino users, the solution can be easily reached.

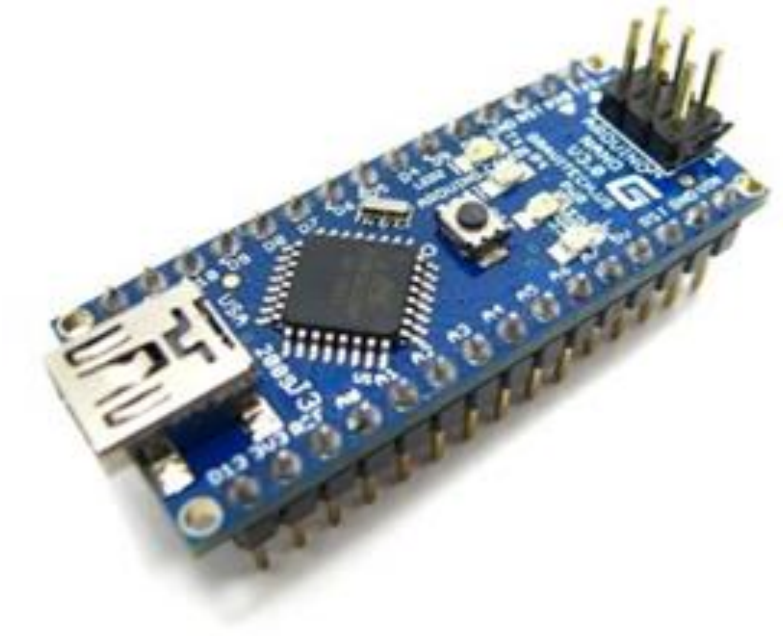


Figure 2Arduino Nano microcontroller used in the project

The Arduino Nano is a small, full and breadboard friendly Arduino card that houses a microcontroller (Arduino Nano 3.x) or Atmega168 (Arduino Nano 2.x) microcontroller. It has almost the same functions as the Arduino Duemilanove. Arduino is designed and used by Nano Gravitech.

The Arduino Nano offers a variety of possibilities for communication with a computer, another Arduino, or other microcontrollers. The ATmega328 and ATmega168 microcontrollers support UART TTL (5V) serial communication, accessible via the RX and TX pins. An FTDI on the card channels the FT232RL serial communications via USB and the FTDI drivers (available in the Arduino software) and the writings on the computer appear as a virtual com port. The RX and TX LEDs on the card flash on the FTDI chip while the USB den serial cable and the USB den is transmitting data.

The Software Serial library allows serial communication over any of the digital pins of the Arduino Nano. The ATmega328 and ATmega168 microcontrollers also support I2C (TWI) and SPI communications.

2.3.HC-06 Bluetooth Module

The HC-06 BT module is a slave operating only and uses a serial communication protocol. In Bluetooth communication, master and slave are determined according to the state of connection start. A master module can initiate the connection, but the slave module cannot initiate the connection. In our project, we will provide an external device to connect to a slave PC or an android device. Bi-directional data can be sent and received in a healthy way.

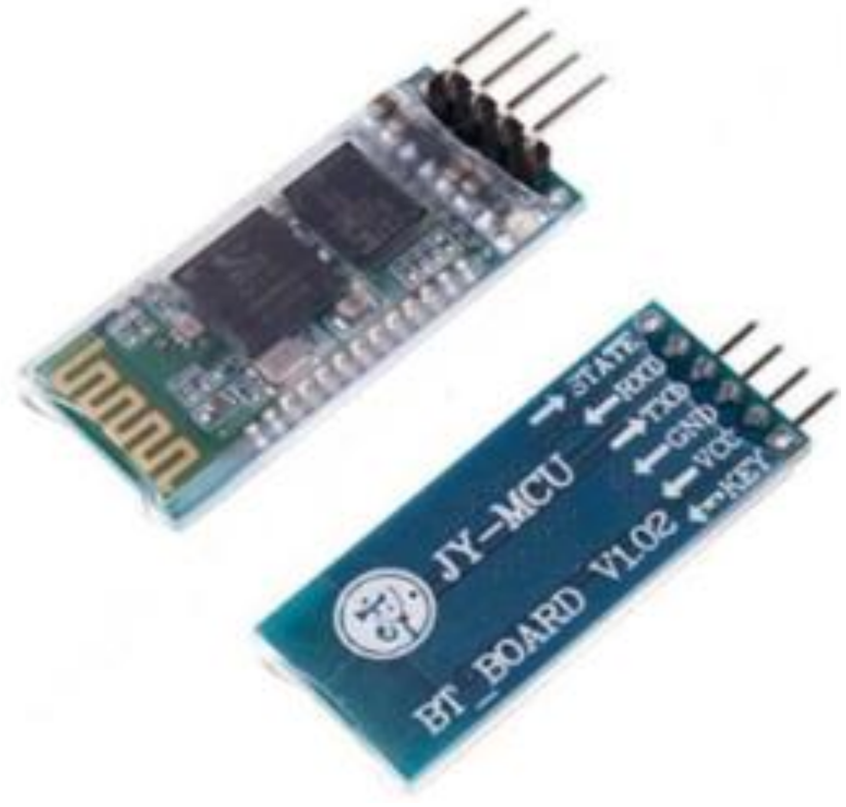


Figure 3HC-06 Bluetooth module

After adding the device to the Bluetooth devices, a virtual com port associated with the module is created on our computer. It is now possible to communicate via Bluetooth via the module. The codes that we send via UART with the microcontroller (PIC18F46K22) can be received via Bluetooth with PC, which is connected to the BT module. Data sent from the PC and COM overreaches the microcontroller.

2.4.Circuit Diagram (Above Board)

Circuit; Servo inputs, Arduino pin inputs, and Bluetooth module inputs. Servo motors are activated in this way, the Bluetooth module Arduino Nano connections and power supply connections are shown. Thanks to this circuit we use, it is possible to distribute the 5V from the power supply to the servo motors.

The circuit image is shown in Fig.4.

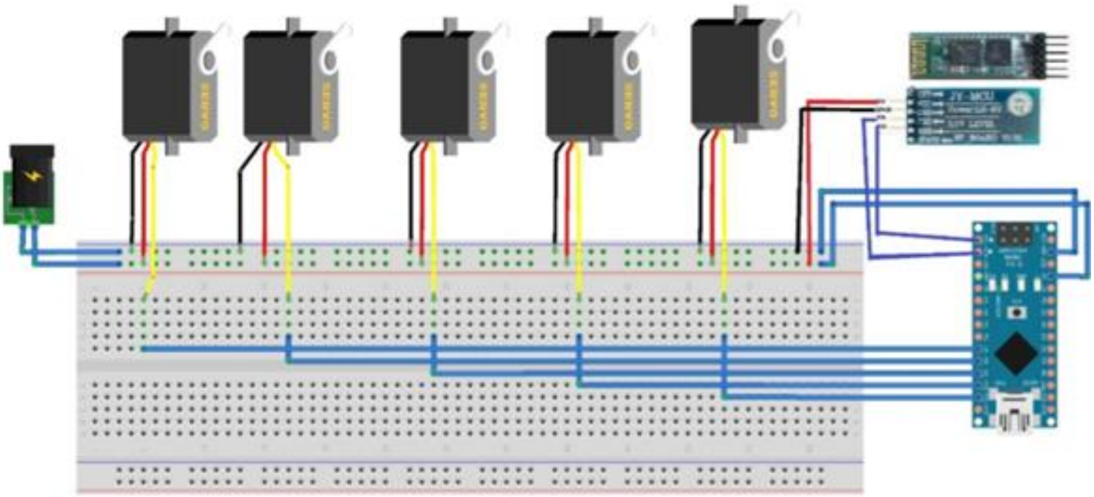


Figure 4Circuit

2.5.Power

The power supply selected for feeding the control circuit of the servomotors is capable of delivering the same current even if all the synchronous servomotors are operating. When all servo motors are operated at the same time, they draw 0.5A current. In addition, 5 V was needed for the Arduino used for robot movement in the project. This requirement is provided by a 5V power supply.

3. ROBOT ARM MECHANICS

3.1. General Characteristics of Robot Arm Mechanics

Kinematics in robotics is the science of motion investigation. Robot arm links can be rotated or offset according to the reference coordinate frame. A systematic and general approach developed by Denavit and Hartenberg establishes the relationship between the robot endpoint and the total displacement of robot arm links (1). Angular and linear displacements between limbs are called joint coordinates and are defined by limb variables. In order to determine the amount of rotation and displacement according to the reference coordinate system of the endpoint, the matrices A which represent the amounts of each limb rotation and displacement are multiplied in turn. If the coordinates of the end point are given, limb variables can be obtained by going backward. These operations are called forward and inverse kinematics. The next section will explain how to determine forward and reverse kinematics. The general transformation matrix can be quite complex even for simple robots. It can be found in standard textbooks such as the Jacobian matrix for standard robots (2) and (3).

Coordinate Frames and Transformation Matrices for a General Robot Arm

An n-dimensional position vector is denoted by an n + 1-dimensional vector and is called a homogeneous coordinate representation. In the following, a matrix of 4 X 4 is shown which shows a position vector in the homogeneous coordinates between the coordinate frames (4).

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

p = p i + p j + p k x y z The vector of the origin of the new frame,

x = x i + x j + x k x y z The contour vector of the x-axis of the new frame,

y = y i + y j + y k x y z The contour vector of the y-axis of the new frame,

z = z i + z j + z k x y z Represents the contouring vector of the z-axis of the new frame.

Column 4 of the transformation matrix has 3 elements corresponding to displacement in the x, y, and z directions.

$$\textbf{Trans}\,(p_x, p_y, p_z) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

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

The matrices representing only the revolutions around the y and z-axes can be written in a similar manner.

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

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

The elements of the transformation matrix can be found by the arithmetic multiplication of pure rotation and displacement matrices. This can be obtained by a sequence of rotations about the axes of the stationary reference frame if the orientation is desired according to the reference frame of the Cartesian space endpoint. Although there are many ways to do this, one of the best known is the "roll-pitch-yaw" transformation. 3 turns. First turn around the x-axis, then y and then around the z-axis.

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

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

Kinematics in robotics is the science of motion investigation. The robot examines the positions of the limb limbs, the relationships between velocities and accelerations, ignoring forces and other factors that affect movement.

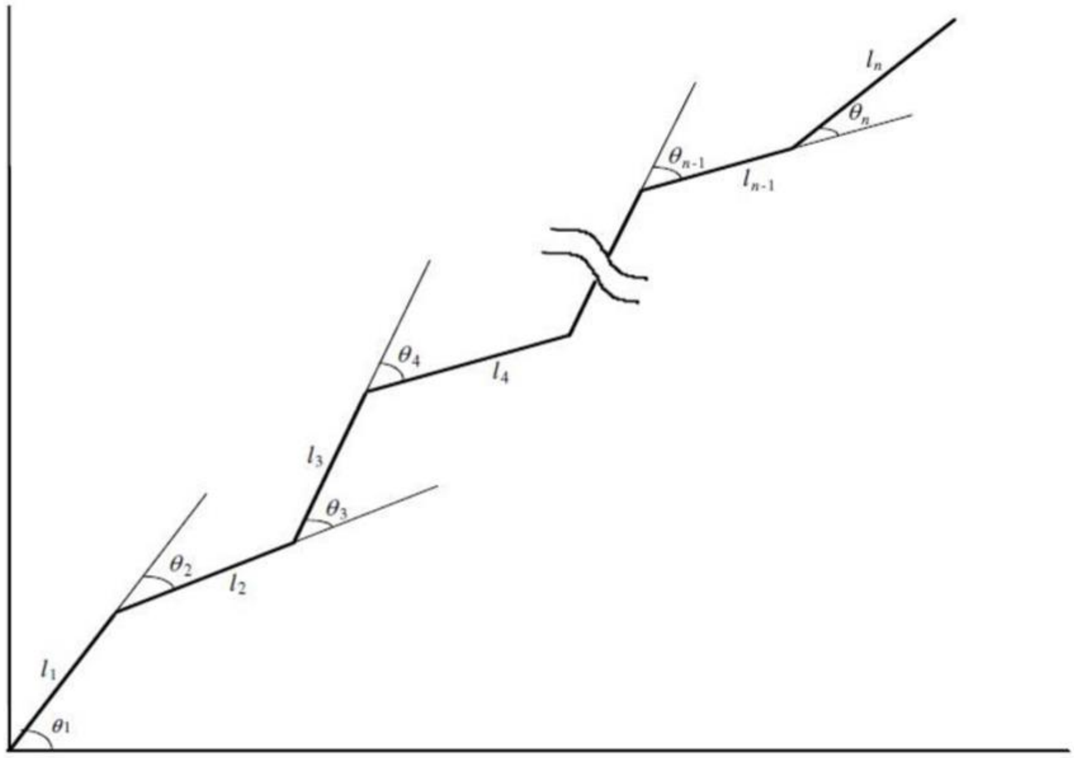


Figure 5Serial robot arm with n lines showing relative angles

The position and orientation of one of the limbs, or the orientation of the limbs between this limb and the end point, will change when the serial limb robot arm is shown in Fig. When the positions and orientations of the limbs change, it is generally desirable to determine the position and orientation of the end point according to the basic reference frame.

When each mobile station is connected to a coordinate frame, the transformation between both limbs can be defined by a homogeneous transformation matrix, called the matrix A. For example, the first limb is connected to the base frame by ${}^0A_1$, which is the matrix 1.A. The transformation matrix ${}^RT_H$ of the end point can be expressed according to the reference frame by the arithmetic product of matrices A from the matrices 1.A of the matrices A to the matrix A of the endpoints.

$${}^R\mathbf{T}_H = {}^R\mathbf{T}_1 . {}^1\mathbf{T}_2 \cdot\cdot {}^{n-2}\mathbf{T}_{n-1} . {}^{n-1}\mathbf{T}_H = {}^0\mathbf{A}_1 . {}^1\mathbf{A}_2 \cdot\cdot {}^{n-2}\mathbf{A}_{n-1} . {}^{n-1}\mathbf{A}_n$$

When this equation is in closed form, any term can be expressed by another and is very important in the kinematic analysis of the robot arm. Because the robot is used to analyze the forward and inverse kinematics of the arm.

3.2. D & H (Denavit and Hartenberg) Coordinate Frames

The distances between the limbs and the displacements are called the joint coordinates defined by limb variables. A series of n articulated limb robots with one DOF for each limb has n + 1 joints. Using the D & H parameters, any nonlinear limb, as shown in Figure 6, can be characterized by structural kinematic limb parameters (5);

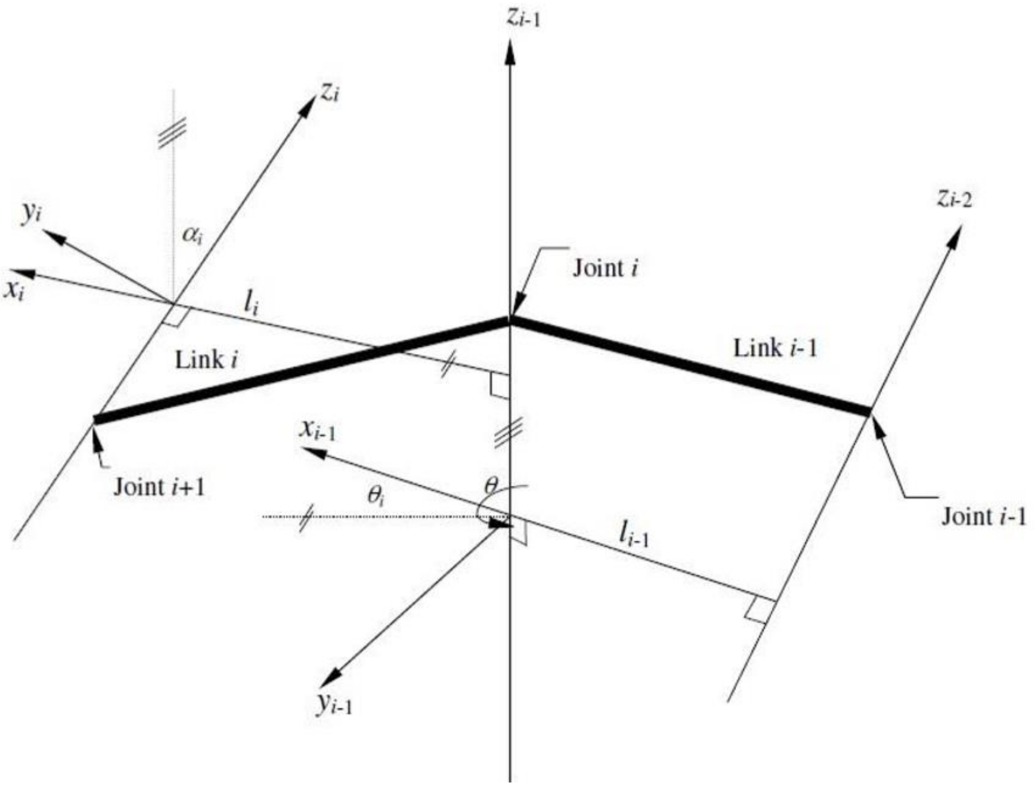


Figure 6D & H parameters

L_n: the length of the limb is the distance along the common normal between the axes of the joints. Z and Z_{i-1}.

i: is the positive angle of rotation about the positive X-axis from the positive Z_{i-1} axis to the positive Z_i axis.

*Ɵ*i: The positive angle of rotation about the positive Z_{i-1} axis from the positive X_{i-1} axis to the positive Zi axis.

Di is the distance to the intersection of the Zi-1 and xi axes along the Xi-1 axis from the origin of the first coordinate frame.

Some of the structural parameters may change over time during robot arm movements. These parameters are called changing parameters, e.g.Ɵi, I angle in the rotary joint, joint variables.

3.3. Advanced Mechanics

The definition of joint space 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 forwarding kinematics. Advanced kinematics is the discovery of the ${}^{R}T_{H}$ transformation matrix, which gives the position and orientation of the end point using the matrices we have already mentioned. The ${}^{R}T_{H}$ transformation matrix can also be obtained by multiplying by the general displacement matrix and the orientation transform matrix of the RPY axes.

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

3.4. Inverse Mechanics

For an endpoint position and orientation given in Cartesian coordinates, the process of finding the values that the joint variables must take is called inverse kinematics. Reverse kinematics can be difficult to solve according to the advanced kinematics. In many cases, it is necessary to use techniques that do not guarantee a solution and include trial and error.

4. DESIGN BRIEF

The design part is divided into two parts, the mechanical part design, and the mechanical part installation. In the design of the mechanical part, the millimetric drawings of the parts to be used in the robot arm construction were made through the help program. In the installation of the mechanical part, the naming of the servomotors used in the robot arm and the tasks during the operation of the robot are explained. The construction of the project consists of several steps. These steps are;

- Determination of the mechanical materials required for the production of the project,

- Determination of microcontroller and software to be used in the project,
- Search and selection of servo motors that will run the robot arm in a proper way,
- Proper selection of mechanical parts,
- Implementation of robot arm assembly,
- Testing the system to see if it works properly with the microcontroller we choose,
- Possible faults have been given in the form of restructuring the system by passing through the eye. These steps have been completed and the design of the robot has been completed.

Arduino and robot arm control are used in the project material are given in the following chart.

Table 1Material inventory chart

Number	Material	Quantity	Price	Amount
1	Tower Pro SG90 Mini Servo Motor	5	12.00	60.00
2	Arduino Nano (AT Mega328)	1	15.00	15.00
3	Robot limbs (made with 3D printer)	5	10.00	50.00
4	Bluetooth module (HC-06)	1	25.00	25.00

4.1. Mechanical Design

Materials needed for the mechanical part of the robot arm were supplied, and then the materials were drawn on SolidWorks in millimetric form. The mechanical part was assembled with these materials. Drawings of the mechanical part of the project are given in Figures 5, 6, 7, 8.

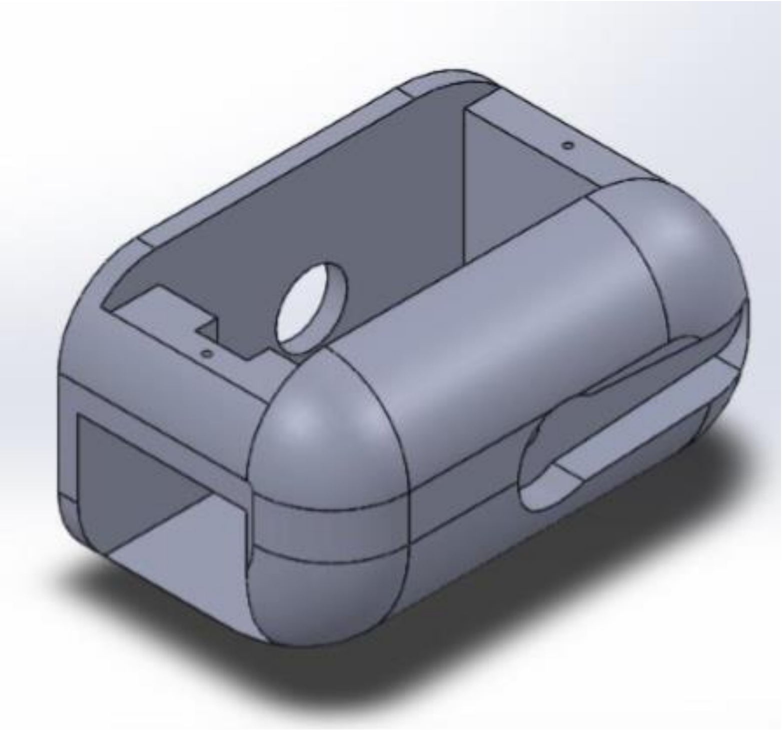


Figure 7Design of mechanical parts

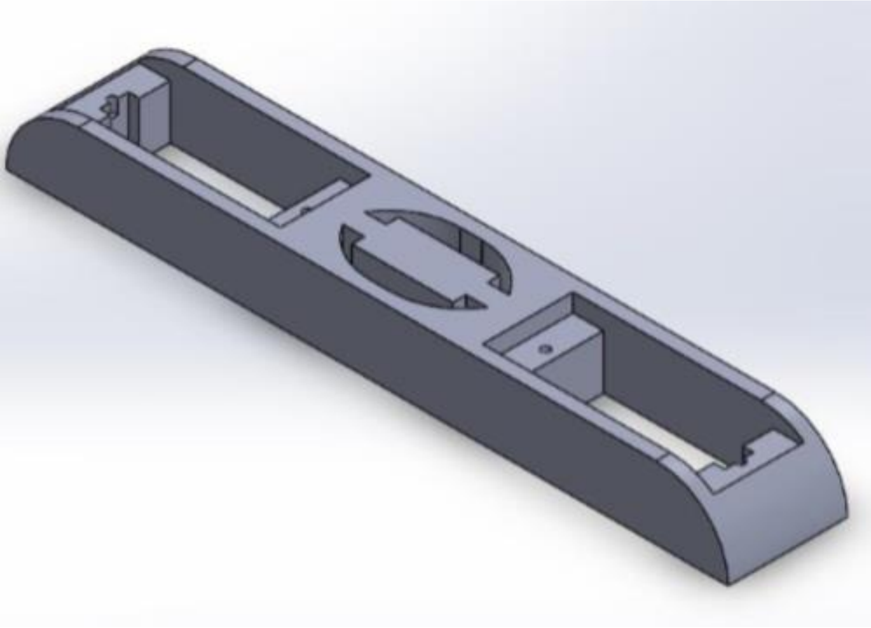


Figure 8Design of mechanical parts

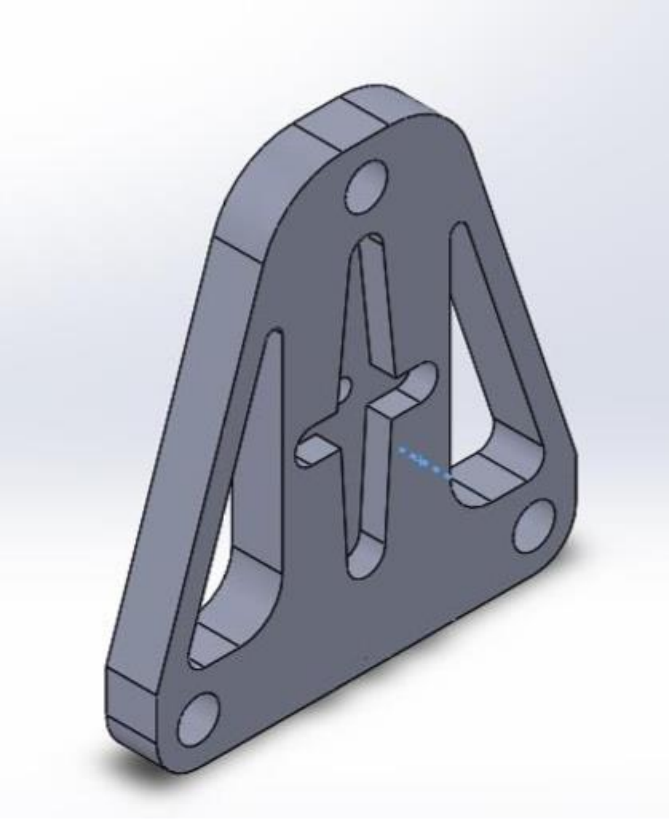


Figure 9Design of mechanical parts

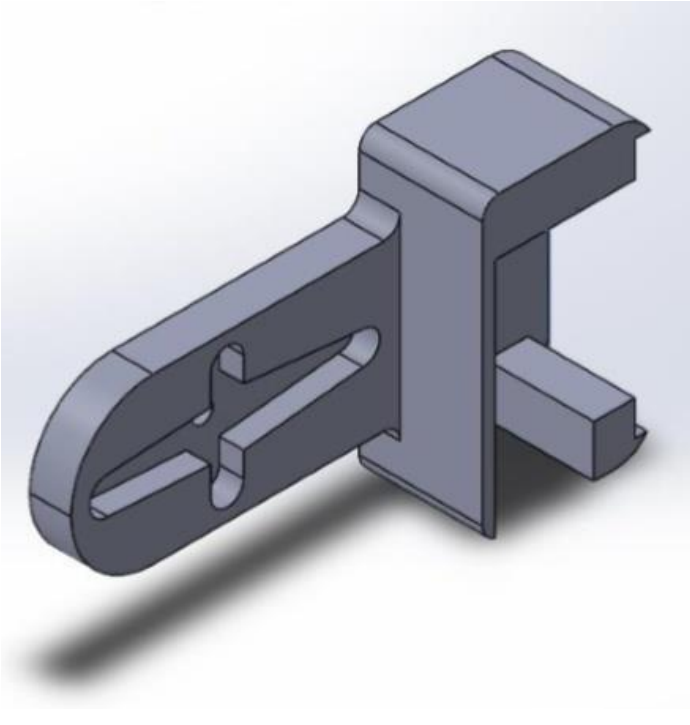


Figure 10Design of mechanical parts

4.2. Mechanical Part Mounting

The cut parts are assembled together with the servo motors and assembly of the robot arm is completed. The robot arm moves by 4 axes and performs this movement with 5 Mini Servo Motors.

After drawing the parts on SolidWorks and making the necessary parts, the design of the mechanical part is started and the robot arm is made of a stationary lower body and movable upper body. There is one servo motor in the stationary lower part of the robot arm and this robot is called axis 1. The 1st axis provides rotation of the robot arm to the right or left. There are 2 servo motors in the moving upper body part. Since these servo motors must operate parallel to each other, both start and end positions are set simultaneously. These two servo motors are named as 2nd axis and 3rd axis respectively.

The tasks of 2nd and 3rd axes are to move the robot arm up and down. Mounting of servo motors in 1st and 2nd and 3rd axes is given in Fig.11.

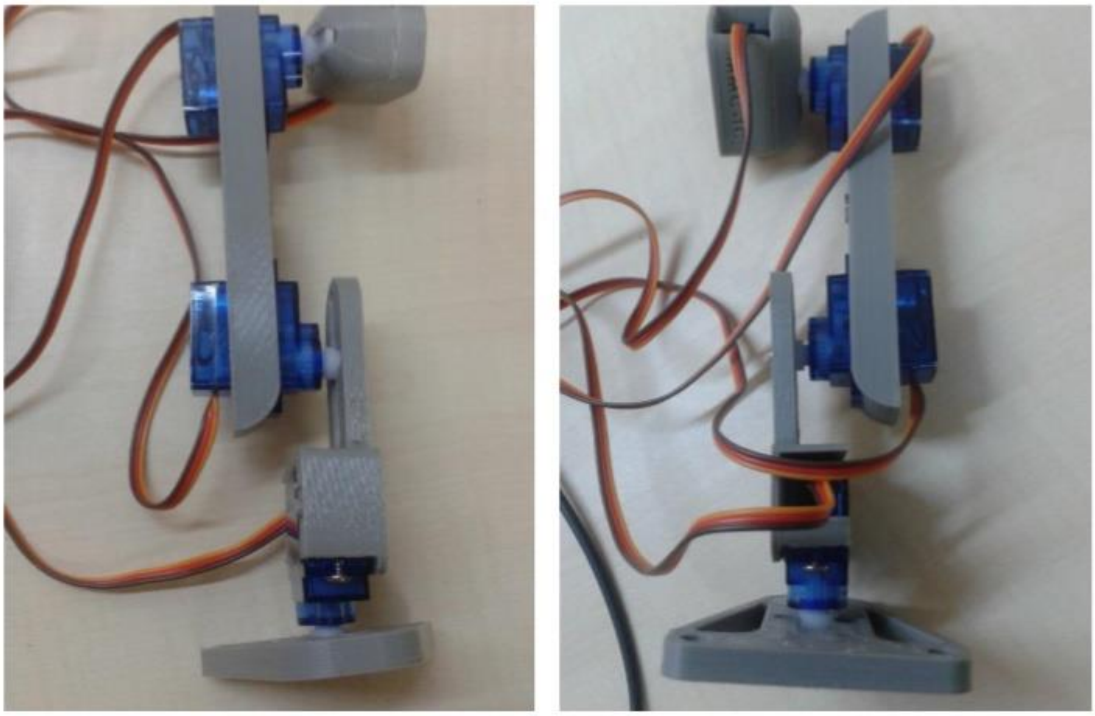


Figure 11Mounting of servo motors in 1st, 2nd, and 3rd axes

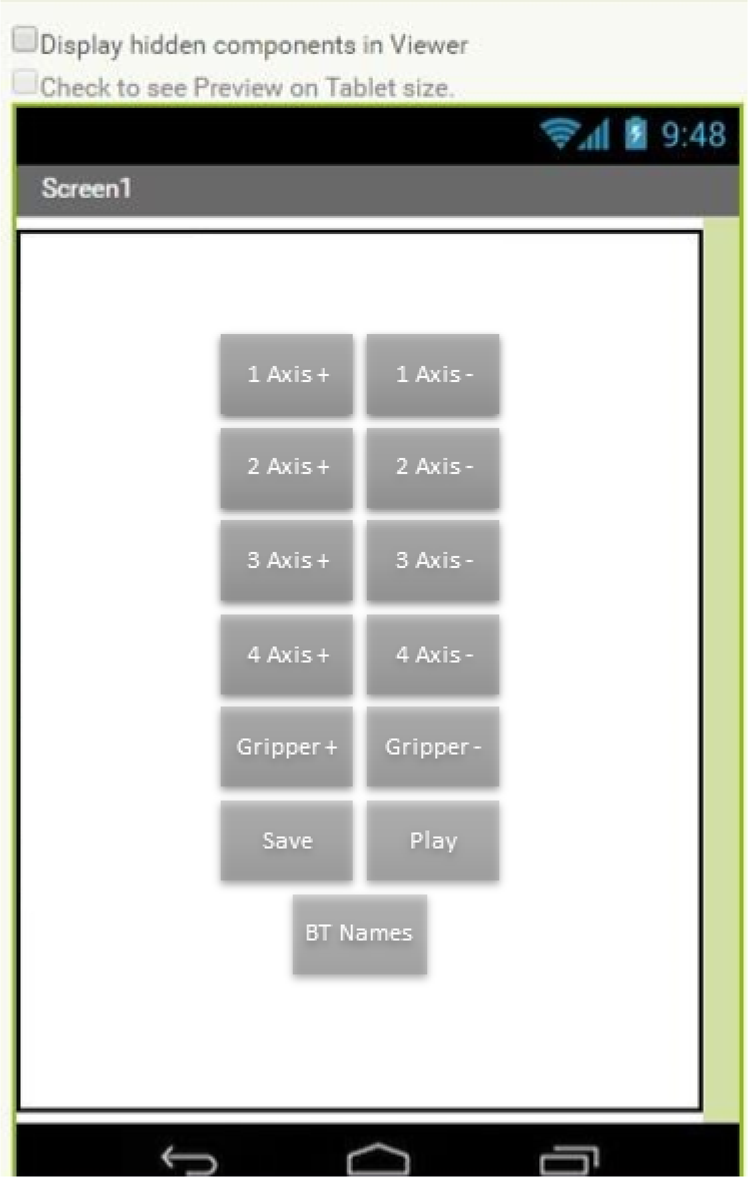
4.3. Android Programming

The Android application we used in this project was done with App Inventor.

What is App Inventor? App Inventor is a free web application developed by Google and later developed by the Massachusetts Institute of Technology (MIT). Other MIT projects use a graphical interface, such as Scratch and StarLogo TNG, to drag and drop visual objects instead of printing lines of code in mobile applications running on the Android operating system.

App Inventor Design is the part of the application that we will design for visual and active components of the application, from Buttons, Pictures, Text Fields; it can be added to the Media Components such as Audio, Video, etc., besides the GPS, Acceleration and Bluetooth sensor can be added to the device. In this section, you have a design criterion where many variables like size, color, the position can be controlled.

The design of the android application we use in the project is shown in Figure 12



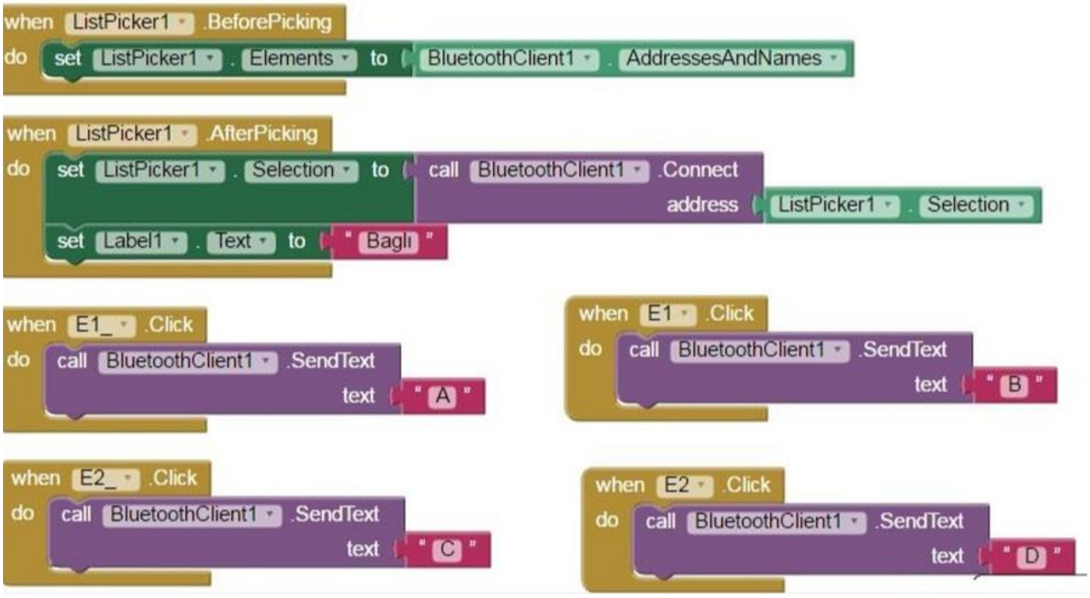
□Display hidden components in Viewer

□Check to see Preview on Tablet size.

Figure 12the design of the android application

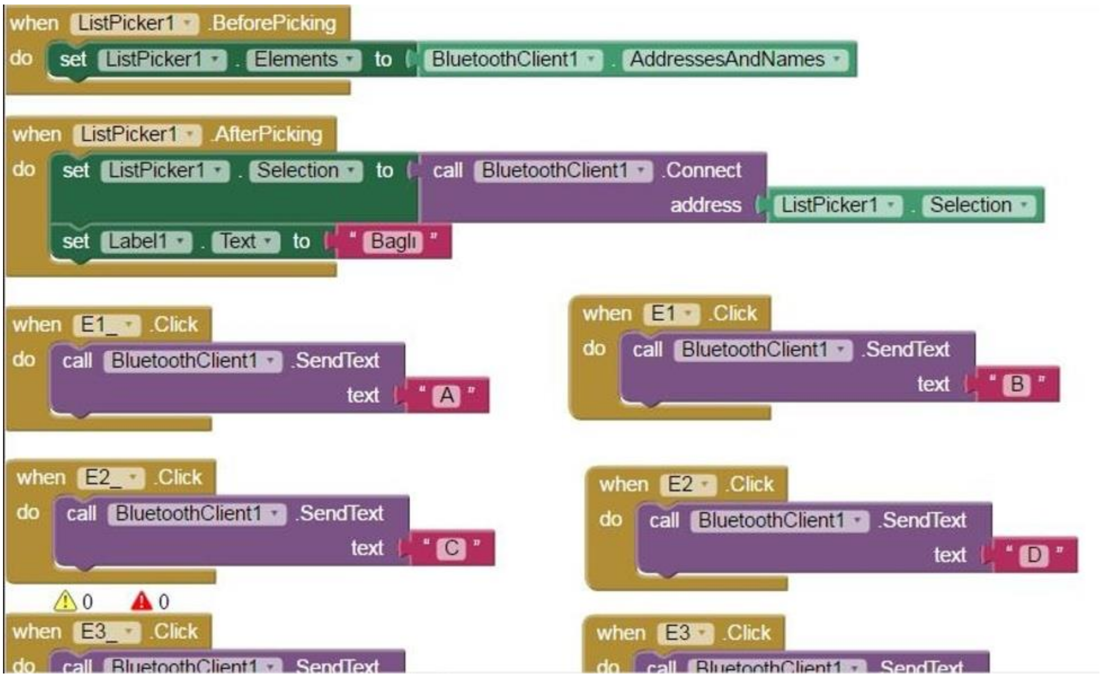
In the App Inventorun Block section, we create the code that will work in the background of the application we are doing with the interlocking blocks with code fragments, just like a Lego. We make sure that all the components that I add to my application have all the events/properties that can be used and combine them into the corresponding blokes.

In the project we have done, android blocks are shown in Figures 13, 14, 15, 16.



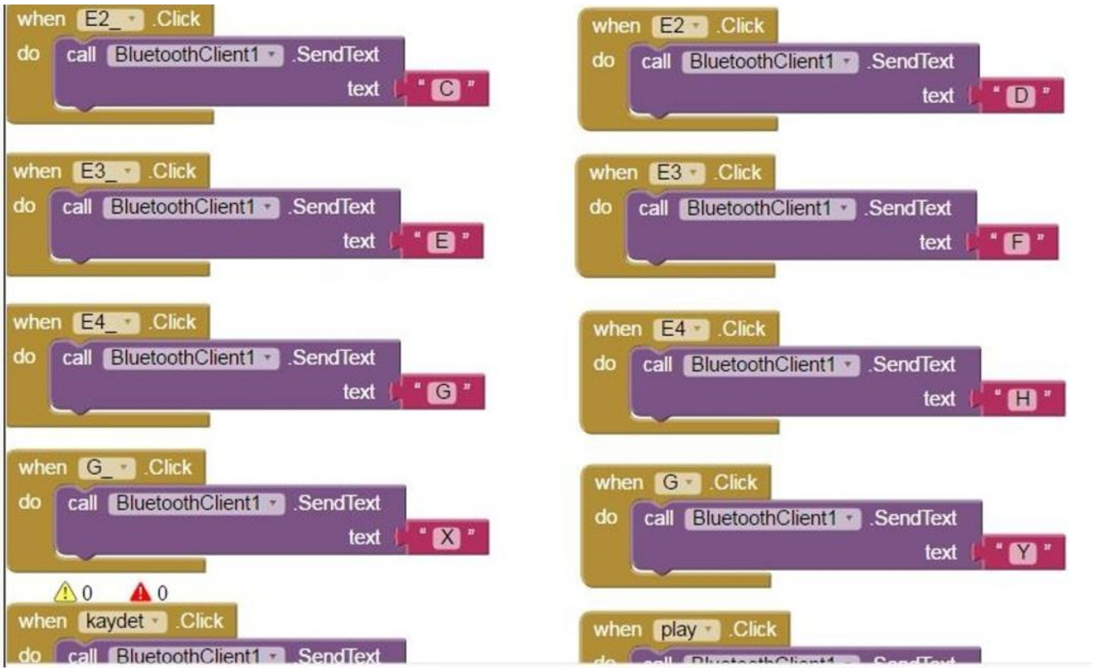
```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to  BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  set ListPicker1 . Selection to  call BluetoothClient1 .Connect
                                                     address  ListPicker1 . Selection
    set Label1 . Text to  " Bagli "

when E1_ .Click                                   when E1 .Click
do  call BluetoothClient1 .SendText               do  call BluetoothClient1 .SendText
                             text  " A "                                       text  " B "

when E2_ .Click                                   when E2 .Click
do  call BluetoothClient1 .SendText               do  call BluetoothClient1 .SendText
                             text  " C "                                       text  " D "
```

Figure 13android blocks



```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to  BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  set ListPicker1 . Selection to  call BluetoothClient1 .Connect
                                                     address  ListPicker1 . Selection
    set Label1 . Text to  " Bagli "

when E1_ .Click                                   when E1 .Click
do  call BluetoothClient1 .SendText               do  call BluetoothClient1 .SendText
                             text  " A "                                       text  " B "

when E2_ .Click                                   when E2 .Click
do  call BluetoothClient1 .SendText               do  call BluetoothClient1 .SendText
                             text  " C "                                       text  " D "
   ⚠0    ▲0
when E3_ .Click                                   when E3 .Click
do  call BluetoothClient1 .SendText               do  call BluetoothClient1 .SendText
```

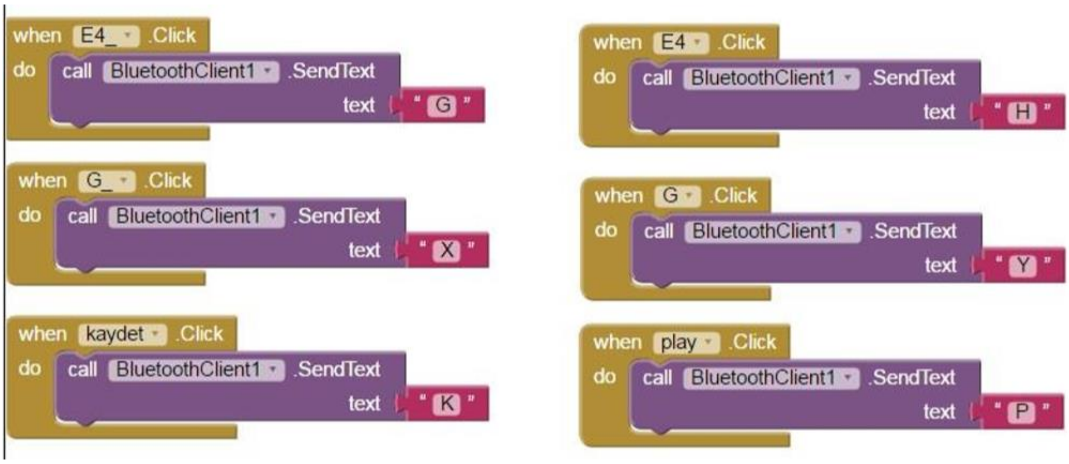
Figure 14android blocks



```
when E2_ .Click
do  call BluetoothClient1 .SendText
                            text " C "

when E3_ .Click
do  call BluetoothClient1 .SendText
                            text " E "

when E4_ .Click
do  call BluetoothClient1 .SendText
                            text " G "

when G_ .Click
do  call BluetoothClient1 .SendText
                            text " X "

⚠ 0   ▲ 0
when kaydet .Click
do  call BluetoothClient1 .SendText
```

```
when E2 .Click
do  call BluetoothClient1 .SendText
                            text " D "

when E3 .Click
do  call BluetoothClient1 .SendText
                            text " F "

when E4 .Click
do  call BluetoothClient1 .SendText
                            text " H "

when G .Click
do  call BluetoothClient1 .SendText
                            text " Y "

when play .Click
do  call BluetoothClient1 .SendText
```

Figure 15android blocks



```
when E4_ .Click
do  call BluetoothClient1 .SendText
                            text " G "

when G_ .Click
do  call BluetoothClient1 .SendText
                            text " X "

when kaydet .Click
do  call BluetoothClient1 .SendText
                            text " K "
```

```
when E4 .Click
do  call BluetoothClient1 .SendText
                            text " H "

when G .Click
do  call BluetoothClient1 .SendText
                            text " Y "

when play .Click
do  call BluetoothClient1 .SendText
                            text " P "
```

Figure 16android blocks

5. EXPERIMENTAL STUDIES

Experimental studies are within themselves; the method followed in the project, robot arm control system, and programming. In the method and robot arm control system part of the project, the basic commands are explained after the basic information is given and the robot can perform the required tasks. In the program section, there is the software information required for the movement of the robot. The programming part of the pro- gram is given in Appendix to the project to make the project more regular.

5.1. Projected Method

First, a historical research on robot arms was carried out and the basic information needed to establish the system was obtained. The robot used in the project types with arm joint and can move in 4 axis directions (left and right, up and down) and also can hold and swing motion thanks to the holder on it. The microcontroller Arduino Nano is used to provide optimal control of the robot arm. The reason for preferring this microcontroller is that it is more accessible to be able to get a solution to a possible error because the open source code is easier to use than the other microcontrollers and the number of users is higher.

After these studies, detailed information has been obtained about the servo motors to be used. The servomotor is preferred because it can be carried out smoothly in the robot project, the motor can be operated precisely and it must be at high torque. The robot arm, 5 servo motors are formed. Servo motors are numbered from top to bottom in order to explain their tasks because of the excess.

5.2. Robot Arm Control

The connection box is made to distribute the 5V voltage from the supply source to the servo motors. In doing so, servo motor inputs, Arduino pin inputs, and communication circuit elements are used. The mechanical part of the robot arm is designed by combining the pre-selected parts appropriately. In order to move the robot arm properly, software with the selected Arduino microcontroller has been implemented and then the experiment with the Bluetooth module and servo motors has been done to learn about the system operation. The software has been implemented with the appropriate Arduino microcontroller selected so that the robot arm can be moved appropriately for the intended purpose. Control of the robot arm is achieved by moving the axes of the android application in the '-' and '+' directions. The Android app is shown in Figure 17.

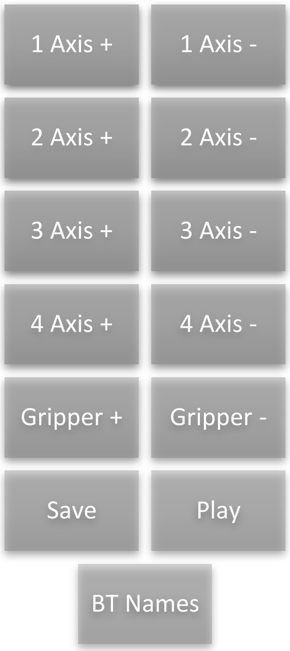


Figure 17Android app interface

6. CONCLUSION

Robotic arms, many areas are developable. Thanks to the robotic arms, many tasks are made easier and the resulting error level has been reduced to a minimum. For example; some pharmacy-based drug-giving robots and a projected robot arm have been developed. In addition to this, the ability to move the robot arm is further increased, and when the camera is placed in the finger area and the sensitivity is increased, it can be used in a wide range of applications from the medical sector to the automation systems. With the robotic arms developed in this way, the risk of infecting the patient in the medical sector is minimized, while the human errors are minimized during the surgical intervention. Despite the fact that the robotic arm made by this project is of prototype quality, it has a quality that can be improved for more robotic systems. Besides these, robotic arm sector, which is open to development, will keep its importance in the future.

The purpose of the project is to provide control of 4 axes moving robot arm design and this robot arm with a suitable microcontroller and Bluetooth module with android application. The necessary theoretical and practical information for this purpose has been obtained and the necessary infrastructure has been established for the project. During the process of making and developing the project, a lot of theoretical knowledge has been transferred to the practice and it has been ensured that it is suitable for the purpose of the project.

REFERENCES

1. WMHW Kadir, RE Samin, BSK Ibrahim. Internet controlled a robotic arm. Procedia Engineering. 2012.
2. MAK Yusoff, RE Samin, mobile robotic arm. Procedia Engineering. 2012.
3. AM Al-Busaidi, Development of an educational environment for online control of a biped robot using MATLAB and Arduino, (MECHATRONICS), 9th France-Japan.2012.
4. HS Juang, KY Lurrr. Design and control of a two-wheel self-balancing robot using the Arduino microcontroller board. Control and Automation (ICCA), 2013.
5. R Krishna, GS Bala, SS ASC, BBP Sarma. Design and implementation of a robotic arm based on haptic technology. Int. J. of Eng. Research. 2012.
6. Electric Electronic Technology-Step and Servo Motors, SVET, 2007.

APPENDIX 1: PROGRAMMING CODES

The program codes necessary to control the servo motors and perform robot arm movements are given below.

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

bytepackage_order=0; package
byteIncomingdata[200];
byteMMC_packageOrder=0;
byteData_size=0;
byteRead_Data_size=0;
intdata_size=0;

boolsec_btn_up = 0;
boolsec_btn_down = 0;
boolup_btn =0;
booldown_btn=0;
boolsave_btn =0;
boolplay_btn  =0;
String hmisend;

byteSecili_Servo=1;
bytehiz=75;

Servo Alt_servo;
Servo Alt_shoulder;
Servo Ust_elbow;
Servo G_wrist;
Servo Gripper;
```

```
intAlt_Servo_Degree=90;
intAlt_shoulder_Degree=90;
intUst_elbow_Degree=90;
intG_wrist_Degree=90;
intGripper_Degree=90;

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

pinMode(2,OUTPUT);
digitalWrite(2,1);
pinMode(13,OUTPUT);
digitalWrite(13,0);

Alt_servo.attach(5);//3
Alt_shoulder.attach(6);  //6
Ust_elbow.attach(9);//11
G_wrist.attach(10);//5
Gripper.attach(11);//9

Alt_servo.write(90);
Alt_shoulder.write(90);
Ust_elbow.write(90);
G_wrist.write(90);
Gripper.write(90);
}
voidDegree_updown()
{
delay(hiz);
if(bt_data=='B')
```

```
   {
//    Serial.print("Alt_servo: ");Serial.println(Alt_Servo_Degree);
Alt_Servo_Degree++;
if(Alt_Servo_Degree> 180) Alt_Servo_Degree=180;

Alt_servo.write(Alt_Servo_Degree);
   }
else if(bt_data=='D')
   {
//     Serial.print("Alt_shoulder: ");Serial.println(Alt_shoulder_Degree);
Alt_shoulder_Degree++;
if(Alt_shoulder_Degree> 180) Alt_shoulder_Degree=180;

Alt_shoulder.write(Alt_shoulder_Degree);
   }
else if(bt_data=='F')
   {
//     Serial.print("Ust_elbow: ");Serial.println(Ust_elbow_Degree);
Ust_elbow_Degree++;
if(Ust_elbow_Degree> 180) Ust_elbow_Degree=180;

Ust_elbow.write(Ust_elbow_Degree);
   }
else if(bt_data=='H')
{
//     Serial.print("G_wrist: ");Serial.println(G_wrist_Degree);
G_wrist_Degree++;
if(G_wrist_Degree> 180) G_wrist_Degree=180;
G_wrist.write(G_wrist_Degree);
   }
else if(bt_data=='Y')
```

```
   {
//      Serial.print("Gripper: ");Serial.println(Gripper_Degree);
Gripper_Degree++;
if(Gripper_Degree> 180) Gripper_Degree=180;
Gripper.write(Gripper_Degree);
   }

if(bt_data=='A')
   {
//     Serial.print("Alt_servo: ");Serial.println(Alt_Servo_Degree);
Alt_Servo_Degree--;
if(Alt_Servo_Degree< 0) Alt_Servo_Degree=0;
Alt_servo.write(Alt_Servo_Degree);
   }
else if(bt_data=='C')
   {
//    Serial.print("Alt_shoulder: ");Serial.println(Alt_shoulder_Degree);
Alt_shoulder_Degree--;
if(Alt_shoulder_Degree< 0) Alt_shoulder_Degree=0;
Alt_shoulder.write(Alt_shoulder_Degree);
   }
else if(bt_data=='E')
   {
//      Serial.print("Ust_elbow: ");Serial.println(Ust_elbow_Degree);
Ust_elbow_Degree--;
if(Ust_elbow_Degree< 0) Ust_elbow_Degree=00;
Ust_elbow.write(Ust_elbow_Degree);
   }
else if(bt_data=='G')
   {
//      Serial.print("G_wrist: ");Serial.println(G_wrist_Degree);
```

```
G_wrist_Degree--;
if(G_wrist_Degree< 0) G_wrist_Degree=00;
G_wrist.write(G_wrist_Degree);
   }
else if(bt_data=='X')
   {
//     Serial.print("Gripper: ");Serial.println(Gripper_Degree);
Gripper_Degree--;
if(Gripper_Degree< 0) Gripper_Degree=00;
Gripper.write(Gripper_Degree);
   }
}
intSaved_Location=0;
intstep=1;
voidSave()
{
if(bt_data=='K')
  {

EEPROM.write(step, Alt_Servo_Degree);
step++;
EEPROM.write(step, Alt_shoulder_Degree);
step++;
EEPROM.write(step, Ust_elbow_Degree);
step++;
EEPROM.write(step, G_wrist_Degree);
step++;
EEPROM.write(step, Gripper_Degree);
step++;
Saved_Location++;
```

```
  // Serial.print("LocationKaydedildi.!
ToplanLocationDizini:");Serial.println(Saved_Location);
}
EEPROM.write(0,Saved_Location);


}
bool play=0;
intplaystep=0;
voidplay()
{
if(bt_data=='P')
  {
    //Serial.println("Saved Is yapiliyor...");
play=1-play;
  }
if(play==1)
    {
playstep=1;
Saved_Location= EEPROM.read(0);
for(inti=0;i<Saved_Location;i++)
      {


intDegree = EEPROM.read(playstep);
while(Alt_Servo_Degree!=Degree)
        {
if(Alt_Servo_Degree>Degree) Alt_Servo_Degree--;
else if(Alt_Servo_Degree<Degree) Alt_Servo_Degree++;
delay(10);
Alt_servo.write(Alt_Servo_Degree);
        }
```

```
playstep++;
delay(100);
Degree = EEPROM.read(playstep);
while(Alt_shoulder_Degree!=Degree)
        {
if(Alt_shoulder_Degree>Degree) Alt_shoulder_Degree--;
else if(Alt_shoulder_Degree<Degree) Alt_shoulder_Degree++;
delay(10);
Alt_shoulder.write(Alt_shoulder_Degree);
        }

playstep++;
delay(100);
Degree = EEPROM.read(playstep);
while(Ust_elbow_Degree!=Degree)
        {
if(Ust_elbow_Degree>Degree) Ust_elbow_Degree--;
else if(Ust_elbow_Degree<Degree) Ust_elbow_Degree++;
delay(10);
Ust_elbow.write(Ust_elbow_Degree);
        }

playstep++;
delay(100);
Degree = EEPROM.read(playstep);
while(G_wrist_Degree!=Degree)
        {
if(G_wrist_Degree>Degree) G_wrist_Degree--;
else if(G_wrist_Degree<Degree) G_wrist_Degree++;
delay(10);
G_wrist.write(G_wrist_Degree);
```

```
        }


playstep++;
delay(100);
Degree = EEPROM.read(playstep);
while(Gripper_Degree!=Degree)
        {
if(Gripper_Degree>Degree) Gripper_Degree--;
else if(Gripper_Degree<Degree) Gripper_Degree++;
delay(10);
Gripper.write(Gripper_Degree);
        }
playstep++;

      }
    }
}

void loop()
{
if(Serial.available()>0)
  {
bt_data=Serial.read();

  }
Degree_updown();
Save();
play();
if(bt_data=='K' || bt_data=='P')
   {
digitalWrite(13,1);
```

```
bt_data=' ';
   }
else  digitalWrite(13,0);
```

Table of Contents