Robotic Arm

Contributors of robotic arm:

Mintu Kumar(2022296): (3D modeling, app development, circuit, and coding also)
Dhruv Sharma (22171) : (coding, debugging)
Shashank Mishra(22603): (Helped in application development and also helped in debugging)

Abstract:

In today's world, automation is used to help people do their work more efficiently. Robots are a type of automation that can assist humans in various tasks. They are popular because they can make work easier and get it done well. Robots are used in many ways, like welding, farming, and even in drones. A robot arm is a common type of robot that works like a human arm. To control a robot, you need a controller, which is like the robot's brain. **This project aims to create a controller using Arduino**, a popular microcontroller board, and develop a mobile app**(MIT Inventor)** to control a robot arm through Bluetooth. The project involves gathering information, designing the hardware, and programming the Arduino and the app. The testing showed that the robot could move accurately, with minor errors in positioning.

Introduction:

In today's highly technological age, automation plays a crucial role in simplifying tasks and improving efficiency. Robots, a common form of automation, are widely used in industries for their precision and continuous operation. The convenience of smartphones has made them essential in our daily lives, offering solutions to various needs at our fingertips.

This project aims to create a **6 Degree of Freedom** (DOF) pick and place robotic arm system using an Arduino controller. The system will be controllable via a smartphone using a Bluetooth module.

Robotic ARM Components:

1. **Mechanical Structure**:
    - The **body** or mechanical structure forms the physical framework of the robotic arm. It includes the **shoulder**, **elbow**, and **wrist** segments.
    - The **shoulder** is the base of the arm and can move forward, backward, or spin.

- The **elbow** is located in the center of the arm, allowing the upper part to move forward and backward.
- The **wrist** provides additional flexibility, allowing rotation and movement in various directions.

2. **Control System**:
    - The **controllers** act as the robotic arm's brains. They process instructions and can operate automatically or manually.
    - Controllers come in various forms, depending on the computing power required.
3. **Actuator**:
    - The **actuator(Which converts electrical energy into mechanical motion)** moves the joints. It converts electrical signals into mechanical motion.
    - The number of actuators determines the **degree of freedom**, allowing the robot to translate, rotate, or bend.
    - The servo comprises four components: the **DC motor**, a **gear train**, a **control circuit**, and a **potentiometer.** They also mentioned that the mechanism used in the servo motor is a closed-loop feedback control system that will determine and ensure the angle of the motor output shaft.

Method :

This research focuses on developing the robot arm controller using an **Arduino microcontroller board**. The flow of the study will start from the observation and data collection, the system schematic design, including the circuit and the mechanical structure, the Arduino program, **the Kadular Creator**, and the last is the system testing.

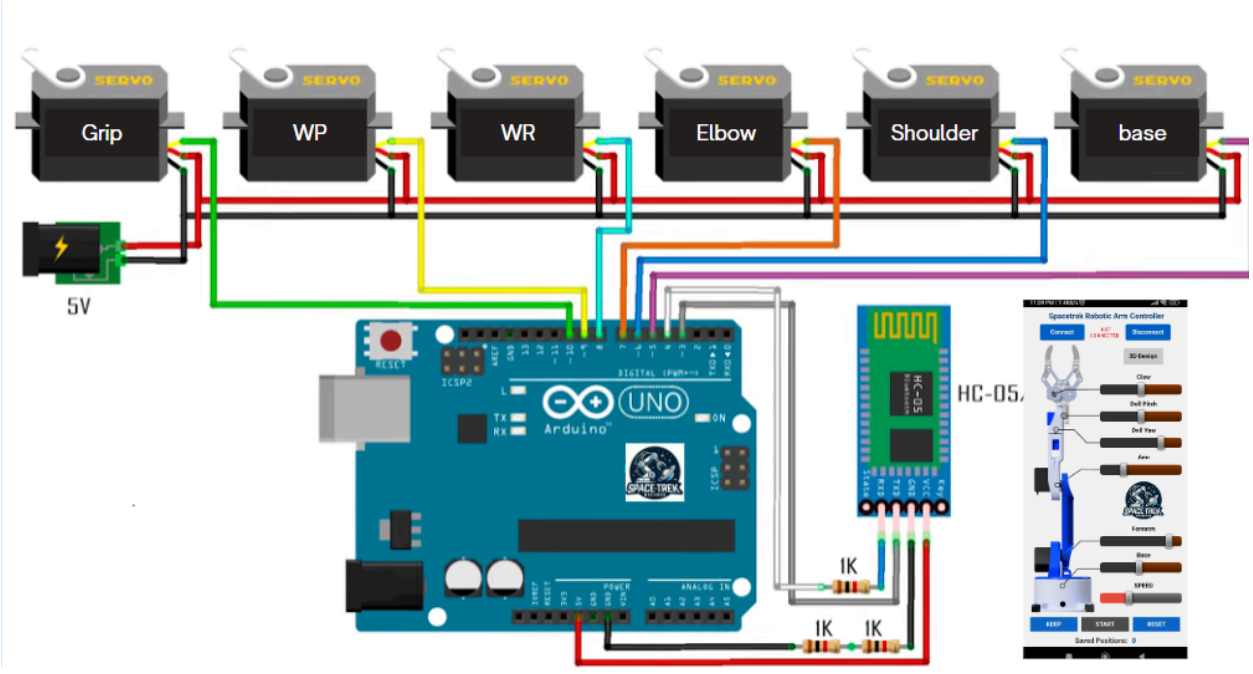
Observation and Information Gathering

This observation aims to learn more about the robot arm controller that uses an Arduino. This project aims to create a 6 DOF robot arm controller using an Arduino board and a smartphone. Regarding the project's primary goal, numerous Arduino-based robot arm projects are posted in various Arduino forums. Next, make changes and improvements to the system to accomplish the project's goal. Additionally, there are numerous details regarding the controller and servo specifications.

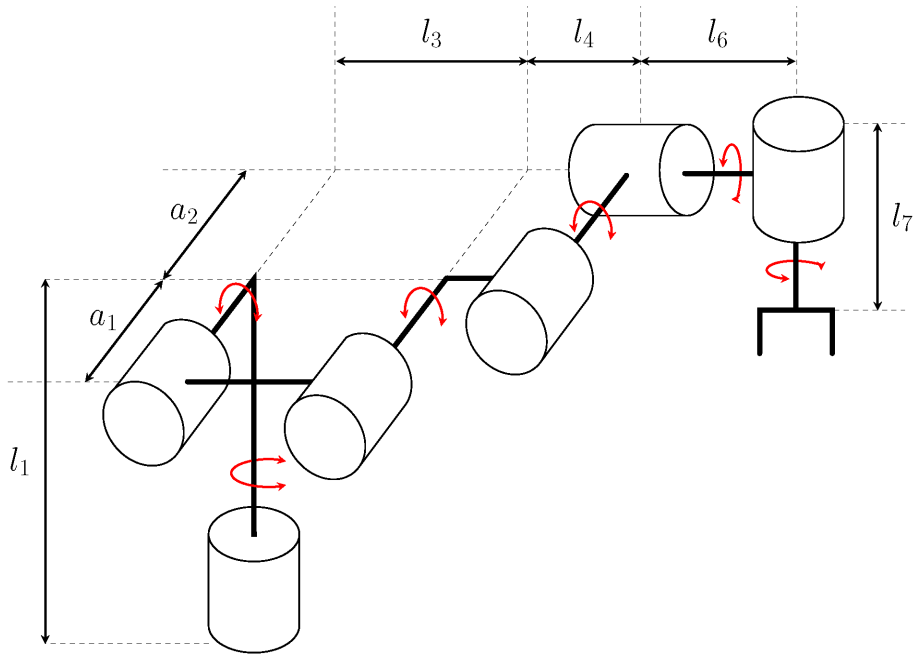
Table 1. Arduino program and application data

No.	Servo Name	Servo Placement	Initial Position (Degree)	Min Value (Degree)	Max Value (Degree)
1	Servo01	Base	90°	0°	180°
2	Servo02	Shoulder	90°	40°	100°
3	Servo03	Elbow	180°	90°	180°
4	Servo04	Wrist	0°	0°	180°
5	Servo05	Grip	180°	110°	180°
6	Servo06	Waist	0°	0°	180°

System Schematic Design:



Mechanical Schematic Design:



MATERIALS:

Component	Specification	Qty
Arduino Board	UNO R3	1
Servomotor	MG996r	3
Servomotor	SG90	3
Breadboard	Generic	1
Jumper wire	Generic	30
Bluetooth module	HC-05	1
External Power Supply	6V	1

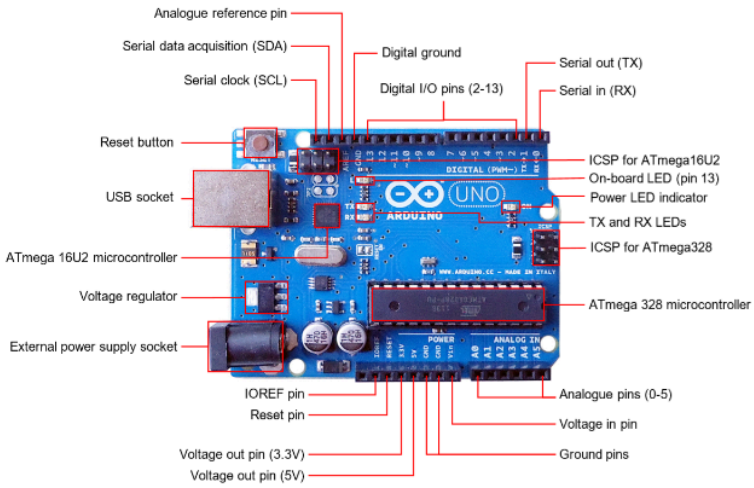
Future Enhancements:

There are several potential future enhancements you could consider for your robotic arm:

- Gesture Control: Implementing gesture recognition technology could allow the robotic arm to interpret hand movements or gestures made by the user, enabling more intuitive control
- Path Planning: Implementing path planning algorithms could optimize the movement of the robotic arm, making its motions more efficient and reducing the time required to complete tasks.
- Remote Operation: Enabling remote operation capabilities could allow users to control the robotic arm from anywhere with an internet connection, opening up possibilities for teleoperation and remote assistance applications.
- Increased Payload Capacity: Enhancements to increase the robotic arm's payload capacity would allow it to handle heavier objects and perform more demanding tasks, expanding its range of applications.

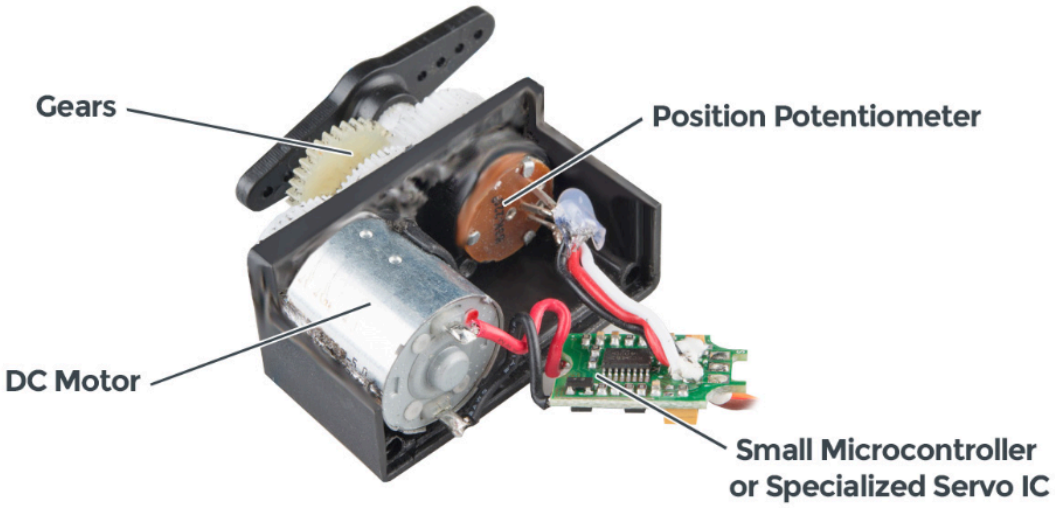
Arduino Board:

Arduino UNO is a single-board microcontroller that makes using electronics in multidisciplinary projects more accessible. The hardware comprises an open-source board designed around an 8-bit Atmel AVR microcontroller or a 32-bit Atmel ARM. The software comprises a standard programming language compiler and a boot loader that executes on the microcontroller.



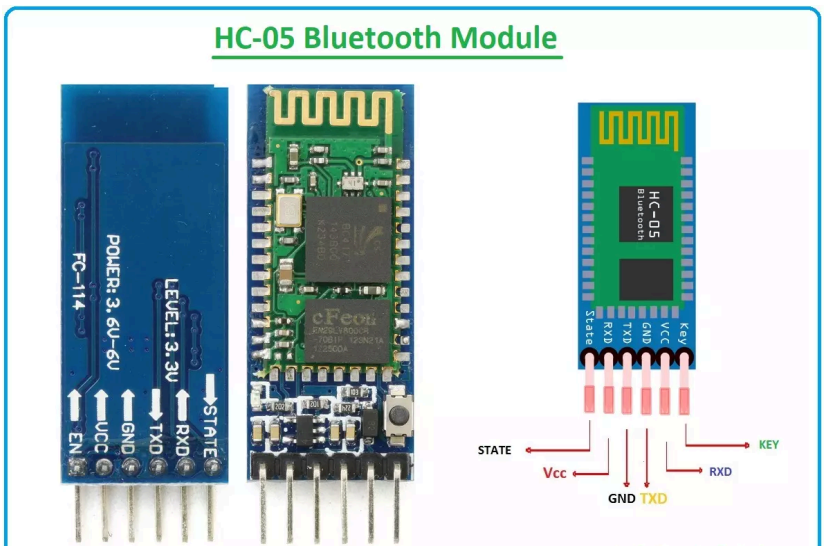
Servomotor:

The Servo Motors includes three wires or leads in them. The first two cables provide the positive supply and ground. The third wire is for the control signal. The wires of a servo motor are color-coded. The ground wire is colored by black color. The DC supply is colored red and has to be connected to a DC voltage supply in the range of 4.8 V to 6 V. For the third cable; all servo motors can have a different color. Generally, it is in yellow, the same as the motors that will be used in this project.



HC-05 Bluetooth module:

HC-05 module is an easy-to-use Bluetooth SPP (Serial Port Protocol) module for transparent wireless serial connection setup. The serial port Bluetooth module is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses CSR Blue Core 04.External single-chip Bluetooth system with CMOS technology and AFH (Adaptive Frequency Hopping Feature). It has a footprint as small as 12.7mmx27mm.



Code:

// contribution in code: Mintu kumar(2022296) and Dhruv sharma(20222171)

```
#include <SoftwareSerial.h>

#include <Servo.h>

Servo servo01;

Servo servo02;

Servo servo03;

Servo servo04;

Servo servo05;

Servo servo06;

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

// Define the stepper motors and the pins the will use

int servo1Pos, servo2Pos, servo3Pos, servo4Pos, servo5Pos, servo6Pos; // current position

int servo1PPos, servo2PPos, servo3PPos, servo4PPos, servo5PPos, servo6PPos; // previous position

int servo01SP[50], servo02SP[50], servo03SP[50], servo04SP[50], servo05SP[50], servo06SP[50]; // for storing
positions/steps

int speedDelay = 20;

int index = 0;

int dataIn;

int m = 0;

void setup() {

  servo01.attach(5);

  servo02.attach(6);

  servo03.attach(7);

  servo04.attach(8);
```

```
  servo05.attach(9);
  servo06.attach(10);
  Bluetooth.begin(9600); // Default baud rate of the Bluetooth module
  Bluetooth.setTimeout(5);
  delay(20);
  Serial.begin(9600);
  // Move robot arm to initial position
  servo1PPos = 90;
  servo01.write(servo1PPos);
  servo2PPos = 90;
  servo02.write(servo2PPos);
  servo3PPos = 90;
  servo03.write(servo3PPos);
  servo4PPos = 90;
  servo04.write(servo4PPos);
  servo5PPos = 90;
  servo05.write(servo5PPos);
  servo6PPos = 90;
  servo06.write(servo6PPos);
}
void loop() {
  // Check for incoming data
  if (Bluetooth.available() > 0) {
    dataIn = Bluetooth.read();  // Read the data
    if (dataIn == 0) {
      m = 0;
    }
```

```
if (dataIn == 1) {
  m = 1;
}
if (dataIn == 2) {
  m = 2;
}
if (dataIn == 3) {
  m = 3;
}
if (dataIn == 4) {
  m = 4;
}
if (dataIn == 5) {
  m = 5;
}
if (dataIn == 6) {
  m = 6;
}
if (dataIn == 7) {
  m = 7;
}
if (dataIn == 8) {
  m = 8;
}
if (dataIn == 9) {
  m = 9;
}
```

```
if (dataIn == 10) {
  m = 10;
}
if (dataIn == 11) {
  m = 11;
}
if (dataIn == 12) {
  m = 12;
}
if (dataIn == 14) {
  m = 14;
}
if (dataIn == 16) {
  m = 16;
}
if (dataIn == 17) {
  m = 17;
}
if (dataIn == 18) {
  m = 18;
}
if (dataIn == 19) {
  m = 19;
}
if (dataIn == 20) {
  m = 20;
}
```

```
if (dataIn == 21) {
  m = 21;
}
if (dataIn == 22) {
  m = 22;
}
if (dataIn == 23) {
  m = 23;
}
if (dataIn == 24) {
  m = 24;
}
if (dataIn == 25) {
  m = 25;
}
if (dataIn == 26) {
  m = 26;
}
if (dataIn == 27) {
  m = 27;
}



// Move robot arm
// Move servo 1 in positive direction
while (m == 16) {
  if (Bluetooth.available() > 0) {
```

```
    m = Bluetooth.read();
  }
  servo01.write(servo1PPos);
  servo1PPos++;
  delay(speedDelay);
}
// Move servo 1 in negative direction
while (m == 17) {
  if (Bluetooth.available() > 0) {
    m = Bluetooth.read();
  }
  servo01.write(servo1PPos);
  servo1PPos--;
  delay(speedDelay);
}
// Move servo 2
while (m == 19) {
  if (Bluetooth.available() > 0) {
    m = Bluetooth.read();
  }
  servo02.write(servo2PPos);
  servo2PPos++;
  delay(speedDelay);
}
while (m == 18) {
  if (Bluetooth.available() > 0) {
    m = Bluetooth.read();
```

```
  }
  servo02.write(servo2PPos);
  servo2PPos--;
  delay(speedDelay);
}
// Move servo 3
while (m == 20) {
  if (Bluetooth.available() > 0) {
    m = Bluetooth.read();
  }
  servo03.write(servo3PPos);
  servo3PPos++;
  delay(speedDelay);
}
while (m == 21) {
  if (Bluetooth.available() > 0) {
    m = Bluetooth.read();
  }
  servo03.write(servo3PPos);
  servo3PPos--;
  delay(speedDelay);
}
// Move servo 4
while (m == 23) {
  if (Bluetooth.available() > 0) {
    m = Bluetooth.read();
  }
```

```
  servo04.write(servo4PPos);
  servo4PPos++;
  delay(speedDelay);
}
while (m == 22) {
  if (Bluetooth.available() > 0) {
    m = Bluetooth.read();
  }
  servo04.write(servo4PPos);
  servo4PPos--;
  delay(speedDelay);
}
// Move servo 5
while (m == 25) {
  if (Bluetooth.available() > 0) {
    m = Bluetooth.read();
  }
  servo05.write(servo5PPos);
  servo5PPos++;
  delay(speedDelay);
}
while (m == 24) {
  if (Bluetooth.available() > 0) {
    m = Bluetooth.read();
  }
  servo05.write(servo5PPos);
  servo5PPos--;
```

```
  delay(speedDelay);
}
// Move servo 6
while (m == 26) {
  if (Bluetooth.available() > 0) {
    m = Bluetooth.read();
  }
  servo06.write(servo6PPos);
  servo6PPos++;
  delay(speedDelay);
}
while (m == 27) {
  if (Bluetooth.available() > 0) {
    m = Bluetooth.read();
  }
  servo06.write(servo6PPos);
  servo6PPos--;
  delay(speedDelay);
}

// If arm speed slider is changed
if (dataIn > 101 & dataIn < 250) {
  speedDelay = dataIn / 10; // Change servo speed (delay time)
}

// If button "SAVE" is pressed
if (m == 12) {
```

```
      servo01SP[index] = servo1PPos;  // save position into the array
      servo02SP[index] = servo2PPos;
      servo03SP[index] = servo3PPos;
      servo04SP[index] = servo4PPos;
      servo05SP[index] = servo5PPos;
      servo06SP[index] = servo6PPos;
      index++;                    // Increase the array index
      m = 0;
    }

    // If button "RUN" is pressed
    if (m == 14) {
      runSteps();
      // If button "RESET" is pressed
      if (dataIn != 14) {
        memset(servo01SP, 0, sizeof(servo01SP)); // Clear the array data to 0
        memset(servo02SP, 0, sizeof(servo02SP));
        memset(servo03SP, 0, sizeof(servo03SP));
        memset(servo04SP, 0, sizeof(servo04SP));
        memset(servo05SP, 0, sizeof(servo05SP));
        memset(servo06SP, 0, sizeof(servo06SP));
        index = 0;  // Index to 0
      }
    }
  }
}
// Automatic mode custom function - run the saved steps
```

```
void runSteps() {
  while (dataIn != 13) {   // Run the steps over and over again until "RESET" button is pressed
    for (int i = 0; i <= index - 2; i++) {  // Run through all steps(index)
      if (Bluetooth.available() > 0) {      // Check for incomding data
        dataIn = Bluetooth.read();
        if ( dataIn == 15) {          // If button "PAUSE" is pressed
          while (dataIn != 14) {        // Wait until "RUN" is pressed again
            if (Bluetooth.available() > 0) {
              dataIn = Bluetooth.read();
              if ( dataIn == 13) {
                break;
              }
            }
          }
        }
        // If speed slider is changed
        if (dataIn > 100 & dataIn < 150) {
          speedDelay = dataIn / 10; // Change servo speed (delay time)
        }
      }
      // Servo 1
      if (servo01SP[i] == servo01SP[i + 1]) {
      }
      if (servo01SP[i] > servo01SP[i + 1]) {
        for ( int j = servo01SP[i]; j >= servo01SP[i + 1]; j--) {
          servo01.write(j);
          delay(speedDelay);
```

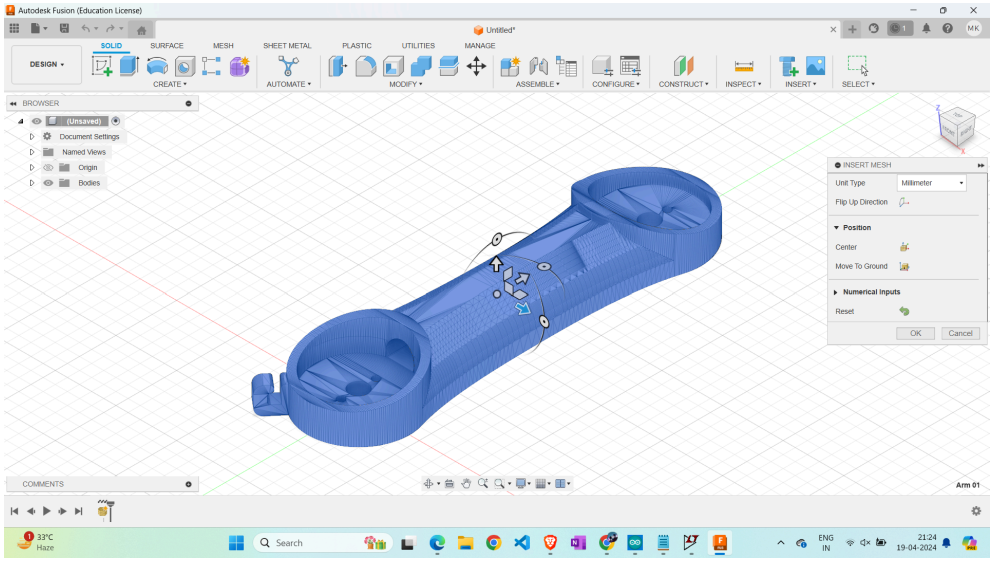
```
  }
}
if (servo01SP[i] < servo01SP[i + 1]) {
  for ( int j = servo01SP[i]; j <= servo01SP[i + 1]; j++) {
    servo01.write(j);
    delay(speedDelay);
  }
}

// Servo 2
if (servo02SP[i] == servo02SP[i + 1]) {
}
if (servo02SP[i] > servo02SP[i + 1]) {
  for ( int j = servo02SP[i]; j >= servo02SP[i + 1]; j--) {
    servo02.write(j);
    delay(speedDelay);
  }
}
if (servo02SP[i] < servo02SP[i + 1]) {
  for ( int j = servo02SP[i]; j <= servo02SP[i + 1]; j++) {
    servo02.write(j);
    delay(speedDelay);
  }
}

// Servo 3
if (servo03SP[i] == servo03SP[i + 1]) {
```

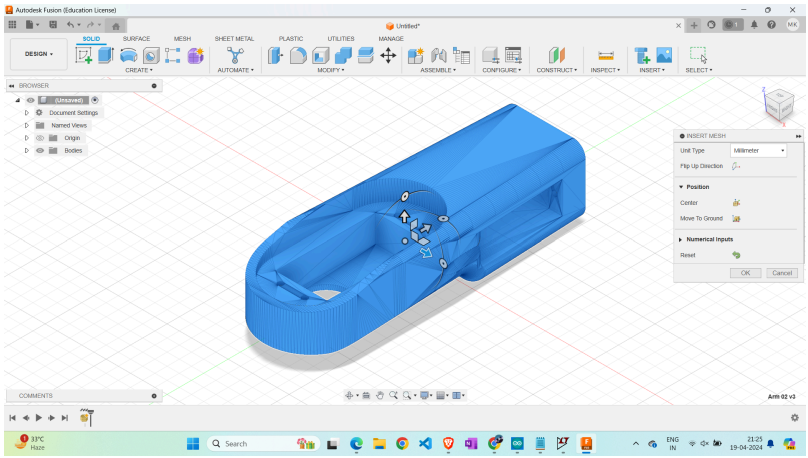
```
}
if (servo03SP[i] > servo03SP[i + 1]) {
  for ( int j = servo03SP[i]; j >= servo03SP[i + 1]; j--) {
    servo03.write(j);
    delay(speedDelay);
  }
}
if (servo03SP[i] < servo03SP[i + 1]) {
  for ( int j = servo03SP[i]; j <= servo03SP[i + 1]; j++) {
    servo03.write(j);
    delay(speedDelay);
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

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

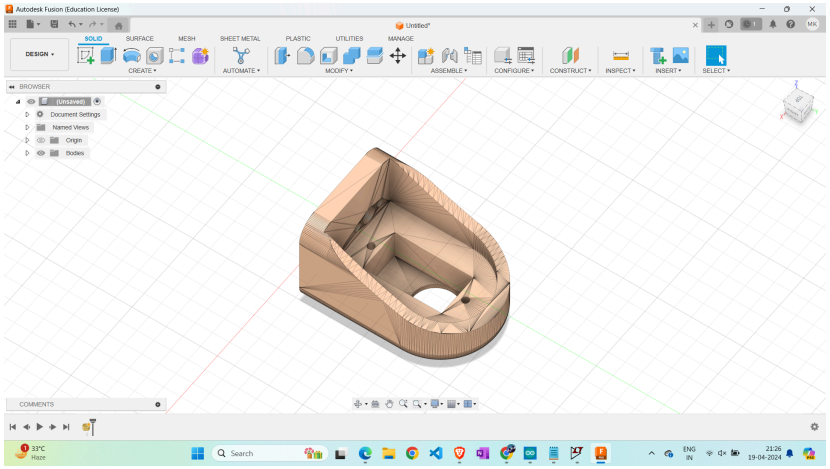
```
  }
}

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

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

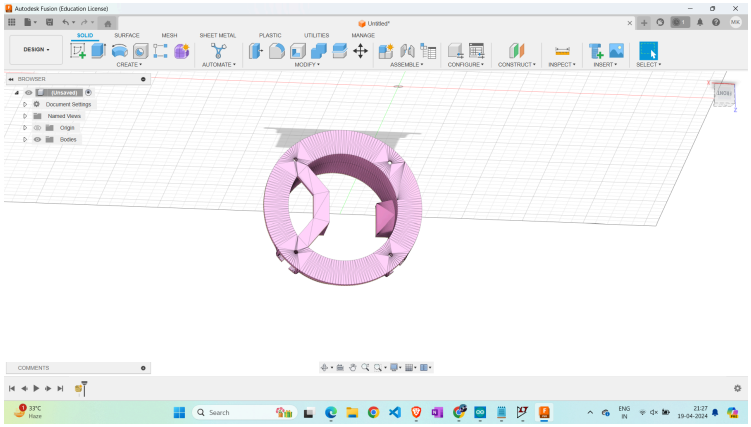
```
    }
    if (servo06SP[i] < servo06SP[i + 1]) {
      for ( int j = servo06SP[i]; j <= servo06SP[i + 1]; j++) {
        servo06.write(j);
        delay(speedDelay);
      }
    }
  }
 }
}
```

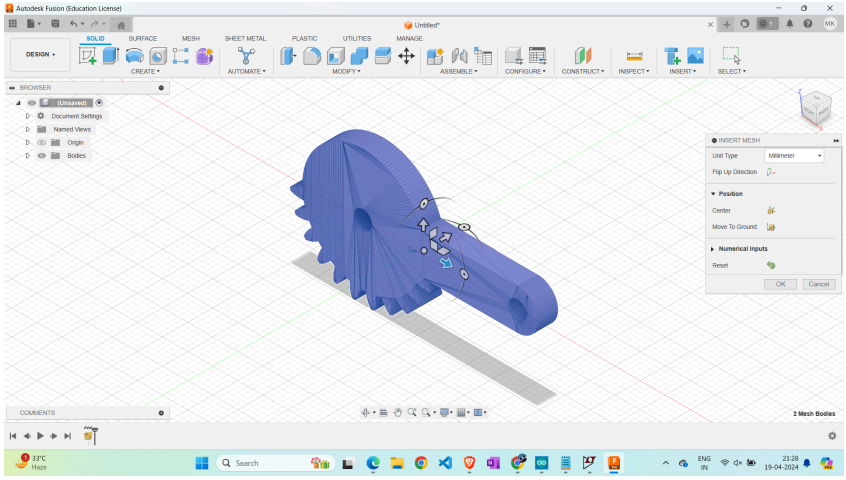
3D models:

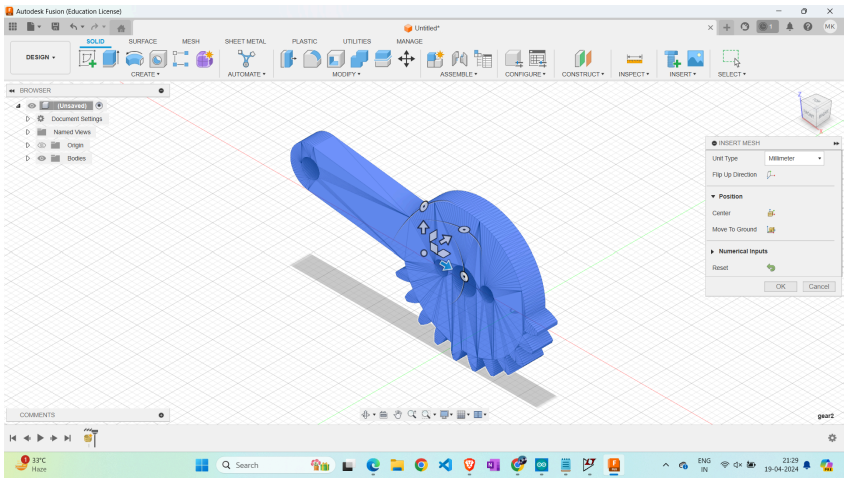


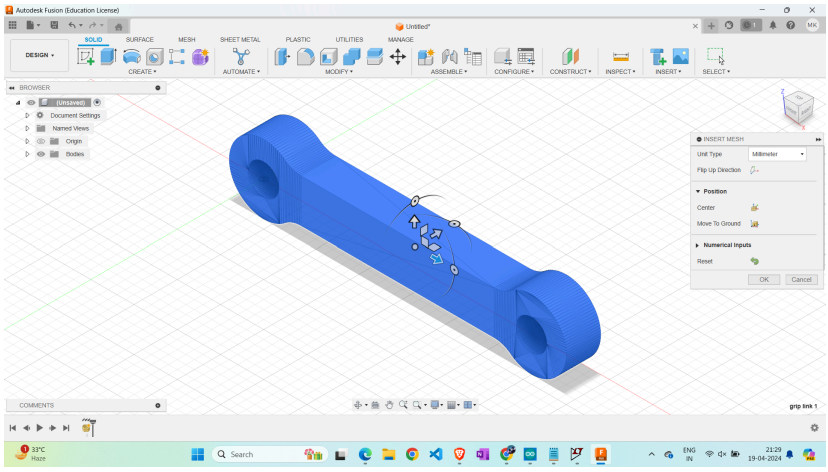

Autodesk Fusion (Education License)
Untitled*
DESIGN
SOLID
SURFACE
MESH
SHEET METAL
PLASTIC
UTILITIES
MANAGE
CREATE
AUTOMATE
MODIFY
ASSEMBLE
CONFIGURE
CONSTRUCT
INSPECT
INSERT
SELECT
BROWSER
(Unsaved)
Document Settings
Named Views
Origin
Bodies
INSERT MESH
Unit Type
Millimeter
Flip Up Direction
Position
Center
Move To Ground
Numerical Inputs
Reset
OK
Cancel
COMMENTS
Arm 02 v3
33°C
Haze
Search
ENG
IN
21:25
19-04-2024

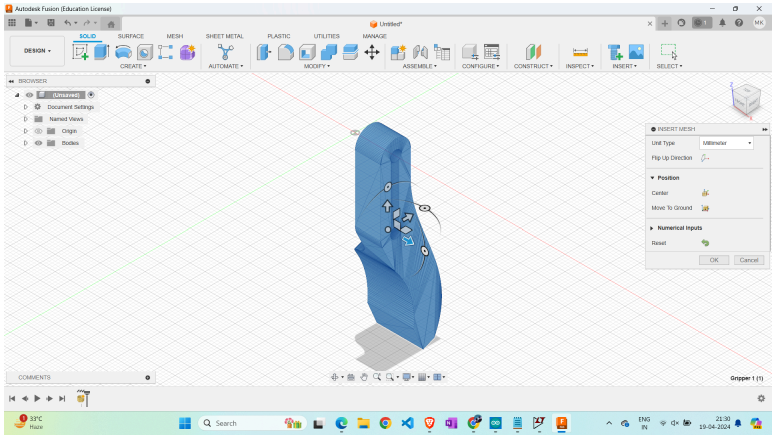

Autodesk Fusion (Education License)
Untitled*
DESIGN
SOLID
SURFACE
MESH
SHEET METAL
PLASTIC
UTILITIES
MANAGE
CREATE
AUTOMATE
MODIFY
ASSEMBLE
CONFIGURE
CONSTRUCT
INSPECT
INSERT
SELECT
BROWSER
(Unsaved)
Document Settings
Named Views
Origin
Bodies
COMMENTS
33°C
Haze
Search
ENG
IN
21:26
19-04-2024

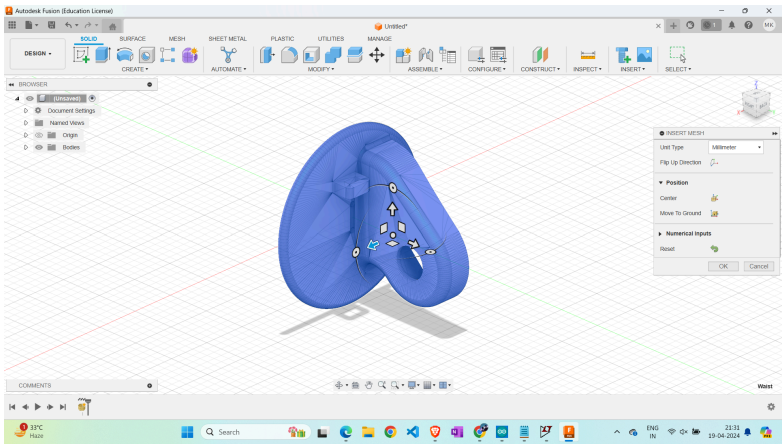

Autodesk Fusion (Education License)
Untitled*
DESIGN
SOLID
SURFACE
MESH
SHEET METAL
PLASTIC
UTILITIES
MANAGE
CREATE
AUTOMATE
MODIFY
ASSEMBLE
CONFIGURE
CONSTRUCT
INSPECT
INSERT
SELECT
BROWSER
(Unsaved)
Document Settings
Named Views
Origin
Bodies
COMMENTS
33°C
Haze
Search
ENG
IN
21:27
19-04-2024


Autodesk Fusion (Education License)
Untitled*
DESIGN
SOLID
SURFACE
MESH
SHEET METAL
PLASTIC
UTILITIES
MANAGE
CREATE
AUTOMATE
MODIFY
ASSEMBLE
CONFIGURE
CONSTRUCT
INSPECT
INSERT
SELECT
BROWSER
(Unsaved)
Document Settings
Named Views
Origin
Bodies
INSERT MESH
Unit Type
Millimeter
Flip Up Direction
Position
Center
Move To Ground
Numerical Inputs
Reset
OK
Cancel
COMMENTS
2 Mesh Bodies
33°C
Haze
Search
ENG
IN
21:28
19-04-2024

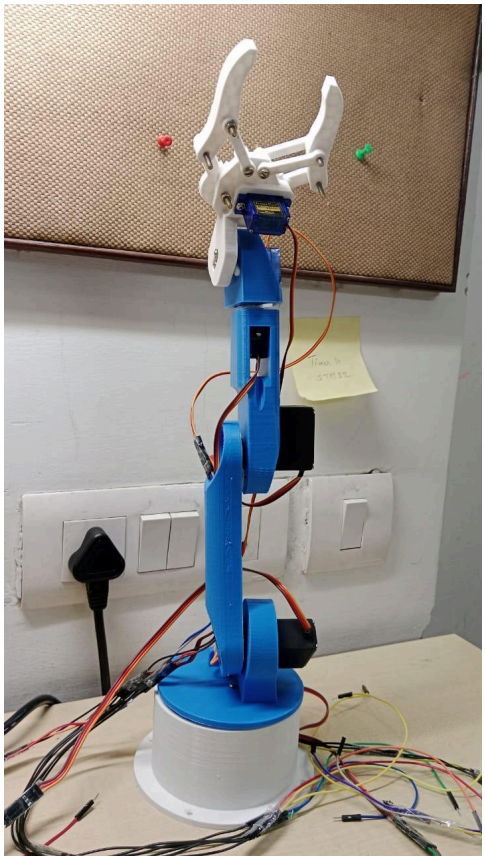

Autodesk Fusion (Education License)
Untitled*
DESIGN
SOLID
SURFACE
MESH
SHEET METAL
PLASTIC
UTILITIES
MANAGE
CREATE
AUTOMATE
MODIFY
ASSEMBLE
CONFIGURE
CONSTRUCT
INSPECT
INSERT
SELECT
BROWSER
(Unsaved)
Document Settings
Named Views
Origin
Bodies
INSERT MESH
Unit Type
Millimeter
Flip Up Direction
Position
Center
Move To Ground
Numerical Inputs
Reset
OK
Cancel
COMMENTS
gear2
33°C
Haze
Search
ENG
IN
21:29
19-04-2024


Autodesk Fusion (Education License)
Untitled*
DESIGN
SOLID
SURFACE
MESH
SHEET METAL
PLASTIC
UTILITIES
MANAGE
CREATE
AUTOMATE
MODIFY
ASSEMBLE
CONFIGURE
CONSTRUCT
INSPECT
INSERT
SELECT
BROWSER
(Unsaved)
Document Settings
Named Views
Origin
Bodies
INSERT MESH
Unit Type
Millimeter
Flip Up Direction
Position
Center
Move To Ground
Numerical Inputs
Reset
OK
Cancel
COMMENTS
grip link 1
33°C
Haze
Search
ENG
IN
21:29
19-04-2024

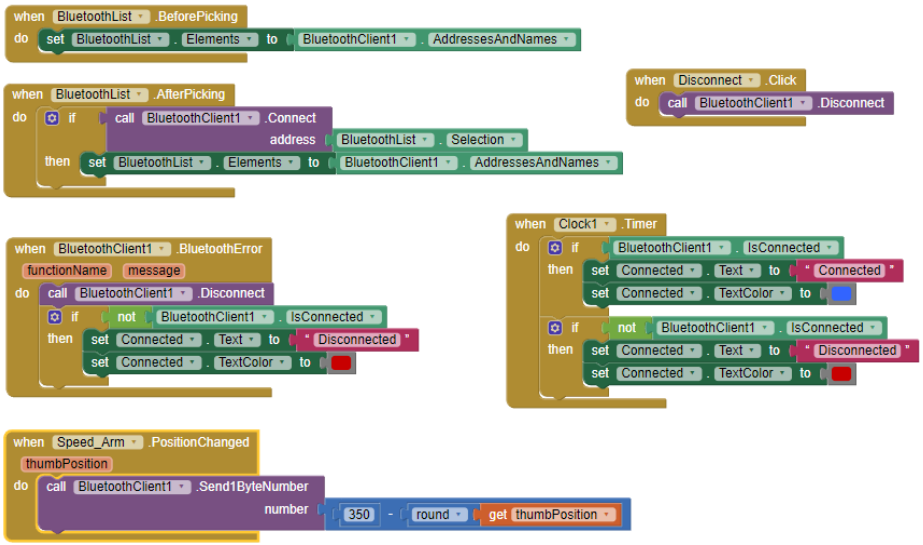




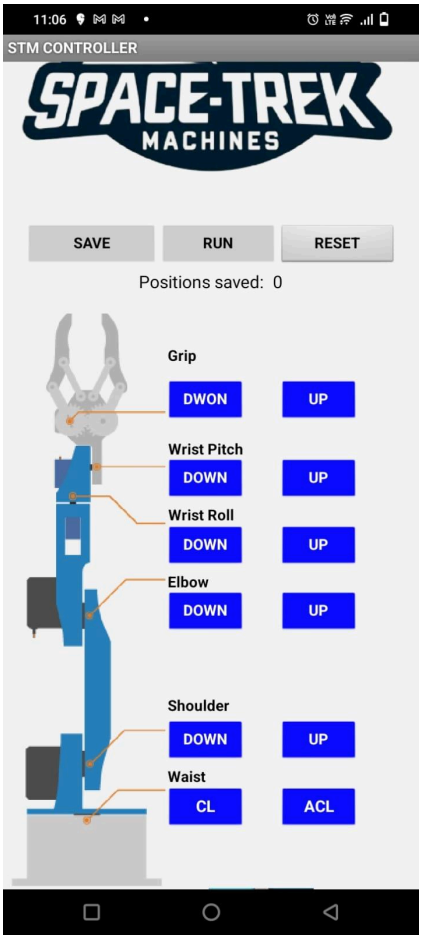
Model:



Block diagram of application :



Application Interface:



THE END