INTERNSHIP REPORT

Dr. B. R. Ambedkar National Institute of Technology, Jalandhar (Punjab), India – 144011

Project Report: Voice Controlled Robotic Arm

Supervised By: - Prabha Singh

Prepared By: - Anurag kumar Maurya (22106020)

Voice Controlled Robotic Arm

Components: -

Jumper wire:

- **Length**: 20 cm (200 mm)
- **Wire Type**: Flexible, typically made of stranded copper for durability and conductivity.
- **Connectors**:
  - **Male**: Fits into female headers or breadboard holes.
  - **Female**: Fits onto male pins, such as those on Arduino boards or sensors.
- **Application**: Connecting Arduino boards to sensors, modules, or other devices.

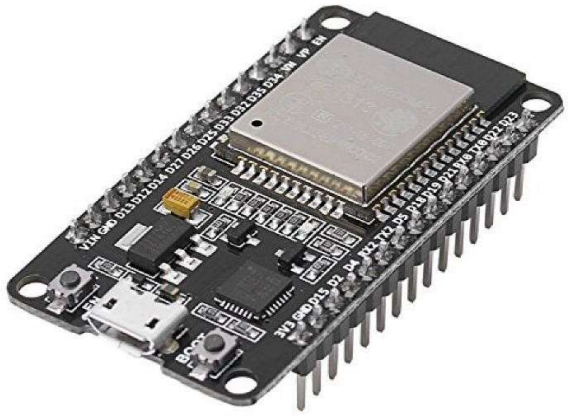
Servo Motor:

- **Dimensions**: 22.2 mm × 11.8 mm × 31 mm.
- **Weight**: 9 g. • **Operating Voltage**: 4.8V to 6V.

Micro USB Cable:

- **USB-A Male:** The standard rectangular USB plug that connects to laptops, desktops, power banks, or wall adapters.
- **Micro-B Male**: The smaller connector designed for portable devices, such as smartphones, tablets, or microcontroller boards like Arduino or Raspberry Pi.
- **Cable Type**: USB 2.0 supports data transfer rates of up to 480 Mbps.

ESP32 38Pin Development Board:



- **CPU**: Dual-core 32-bit Xtensa LX6 microprocessor.
- **RAM**: 520 KB SRAM + 8 KB RTC SRAM.
- **Wi-Fi**: 802.11 b/g/n, 2.4 GHz.
- **Analog Inputs**: 16 ADC pins (12-bit resolution).
- **Analog Outputs**: 2 DAC pins.

○ 3D Printed Robotic Arm Frame:

- **Base:** Provides stability and support for the arm.
- **Joints:** Allow rotational or linear movement.
- **Links:** Connect the joints and transmit motion.
- **Gripper:** Perform tasks like picking, placing, or holding objects.
- **Applications:** Pick and Place, Prototyping etc.

○ 5V Battery:

- **Output Voltage:** 5V

- **Application:** Robotics and Automation, IoT Devices, Microcontroller Projects

Photos of Project:

Code:

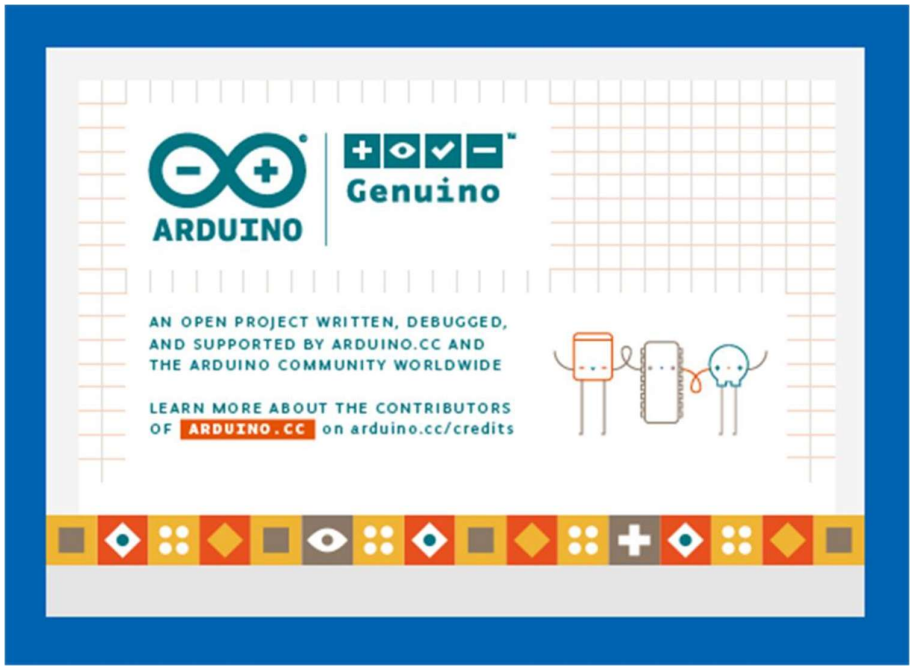
```
#include <WiFi.h>

#include <ESP32Servo.h>

#include <WebServer.h>


// Wi-Fi credentials const char* ssid

= "Himesh's Wifi"; const char*

password = "12345678";


// Servo pins const int basePin =

18; const int shoulderPin = 19;

const int elbowPin = 21; const

int wristRotatePin = 22; const

int wristUpDownPin = 23; const

int gripperPin = 25;


// Servo objects

Servo baseServo, shoulderServo, elbowServo, wristRotateServo,
wristUpDownServo, gripperServo;


// Create a web server on port 80

WebServer server(80);


// Function prototypes void handleServoControl(String servoName,

Servo& servo, int pin); void setup() {
```

```
  // Initialize Serial Monitor
  Serial.begin(115200);

  // Connect to Wi-Fi
  Serial.print("Connecting to Wi-Fi...");
WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWi-Fi connected!");
  Serial.print("ESP32 IP Address: ");
  Serial.println(WiFi.localIP());

  // Attach servos to pins and initialize
Serial.println("Attaching servos...");
baseServo.attach(basePin);
shoulderServo.attach(shoulderPin);
elbowServo.attach(elbowPin);
wristRotateServo.attach(wristRotatePin);
wristUpDownServo.attach(wristUpDownPin);
  gripperServo.attach(gripperPin);

  baseServo.write(90);
shoulderServo.write(90);
```

```
elbowServo.write(90);
wristRotateServo.write(90);
wristUpDownServo.write(90);
gripperServo.write(90);

  Serial.println("All servos initialized to 90 degrees.");

  // Set up HTTP endpoints for each servo   server.on("/base", []() {
handleServoControl("base", baseServo, basePin); });
  server.on("/shoulder", []() { handleServoControl("shoulder", shoulderServo,
shoulderPin); });
  server.on("/elbow", []() { handleServoControl("elbow", elbowServo,
elbowPin); });
  server.on("/wrist_rotate", []() { handleServoControl("wrist_rotate",
wristRotateServo, wristRotatePin); });
  server.on("/wrist_updown", []() { handleServoControl("wrist_updown",
wristUpDownServo, wristUpDownPin); });
  server.on("/gripper", []() { handleServoControl("gripper", gripperServo,
gripperPin); });

  server.begin();
  Serial.println("HTTP server started. Use '/[servo_name]?angle=...' to control
servos.");
}

void loop() {
  // Handle client requests
server.handleClient();
```

```
}
// Function to handle servo control via HTTP requests
void handleServoControl(String servoName, Servo& servo, int pin) {
  Serial.printf("Handling request for /%s...\n", servoName.c_str());

  // Check for 'angle' parameter in URL
if (server.hasArg("angle")) {     int angle
= server.arg("angle").toInt();
    Serial.printf("Received angle for %s: %d\n", servoName.c_str(), angle);
if (angle >= 0 && angle <= 180) {       servo.write(angle);
     Serial.printf("%s servo rotated to %d degrees\n", servoName.c_str(), angle);
server.send(200, "text/plain", "Success");
    } else {
     Serial.printf("Invalid angle for %s! Must be between 0 and 180.\n",
servoName.c_str());       server.send(400, "text/plain", "Invalid angle
(0-180 degrees only)");
    }
  } else {
   Serial.printf("Missing 'angle' parameter in the request for %s.\n",
servoName.c_str());     server.send(400, "text/plain", "Missing
'angle' parameter");
  }
}
```

○ Software used: Arduino IDE



Construction:

The construction of the voice-controlled robotic arm involves assembling both hardware and software components into a functional system. The base of the arm is built using a 3D-printed frame, designed for stability and modularity, providing mounting points for servo motors and other components. The servo motors, such as SG90 or MG996R, are strategically placed at each joint of the arm to control its movement, including rotation, lifting, and gripping actions. These motors are connected to the ESP32 38-pin development board using jumper wires, which serve as the primary microcontroller for the project. The ESP32 processes voice commands received via Wi-Fi or Bluetooth and generates precise pulse-width modulation (PWM) signals to drive the servos. A 5V battery powers the entire system, ensuring portability and uninterrupted operation. The micro USB cable is used for programming the ESP32 and acts as an additional power source during development. The components are carefully integrated into the frame, ensuring proper alignment of the joints and smooth motion. This setup is complemented by voice recognition software, either locally processed or cloud-based, which interprets spoken commands and converts them into actionable instructions for the robotic arm. This modular and scalable design enables flexibility in functionality and ease of future upgrades.

Working:

- **Voice Command Input**:
  - The user gives a voice command, which is captured via a microphone or a smartphone app connected to the ESP32 via Wi-Fi or Bluetooth.
- **Voice Recognition**:
  - The voice recognition software processes the captured voice command, converting it into a digital signal (e.g., “move up,” “rotate left,” or “close gripper”).
- **Signal Processing by ESP32**:
  - The ESP32 microcontroller receives the processed voice command and interprets it based on pre-programmed instructions.

- It generates the corresponding pulse-width modulation (PWM) signals for servo control.

Servo Motor Control:

- The PWM signals are sent from the ESP32 to the servo motors that control the joints of the robotic arm.
- Each servo motor controls a specific joint (e.g., base, shoulder, elbow, wrist, or gripper) to achieve the desired movement.

Arm Movement:

- For example:
  - "Move up" sends a signal to the shoulder servo motor to lift the arm.
  - "Rotate left" sends a signal to the base servo motor to rotate the arm in the specified direction.
  - "Close gripper" triggers the gripper servo to close and grasp an object.

Power Supply:

- The system is powered by a 5V battery, ensuring the robotic arm is portable and operates independently without the need for a fixed power source.

Development & Debugging:

- The micro USB cable is used during the programming phase to upload the control code to the ESP32. It can also serve as a power source during development and testing.

Real-Time Execution:

- The robotic arm performs the movements in real-time, responding to voice commands and carrying out tasks such as object manipulation or movement based on the user's instructions.

Scope of Future Improvement:

- **Enhanced Voice Recognition Accuracy**:
  - Implement AI-based voice recognition models or integrate services like Google Assistant or Amazon Alexa for more accurate command recognition, enabling multi-language support and handling complex commands.
- **Integration of Vision and Sensors**:
  - Add cameras or ultrasonic sensors to the robotic arm for object detection and spatial awareness. This could enable the arm to autonomously pick and place objects, avoiding obstacles and improving precision.
- **Improved Battery Life and Power Management**:
  - Upgrade to a higher-capacity, rechargeable battery (e.g., Li-ion or LiPo) with better energy efficiency to extend the arm's runtime. Implement power-saving techniques like sleep modes for the ESP32 when idle.
- **Increased Degrees of Freedom (DoF)**:
  - Add more joints or actuators to increase the flexibility and precision of the robotic arm, allowing it to perform more complex tasks or handle a wider variety of objects.
- **Wireless Control and Remote Monitoring**:
  - Incorporate Bluetooth or Wi-Fi-based mobile applications for wireless control and remote operation. This can provide real-time feedback and the ability to control the arm from anywhere, with added features like motion tracking or a user-friendly interface.