ADVANCED HARDWARE DESIGN COURSE PROJECT

Android Remote Controlled Vehicular Robotic Arm

Team Name: ISAA

Team Members
Alan Devkota
Adarsh Jain
Swetha Velpula
Isham Sarjil Ahsan

1. Table of Contents

2. Abstract

As the demand for robots is increasing and are being integrated into working tasks to replace humans in both industrial and service areas. Therefore, we propose a model of vehicular robotic arm. In this project, a vehicular robotic arm is made and controlled by an Android mobile app. The development of this model is through ESP32 along with a mobile phone for controlling the robot. This prototype may be expected to overcome the problems of picking hazardous objects or non-hazardous objects that are far away from the user and where displacement of very heavy objects is needed from one place to another as automation is required in many industries.

3. Introduction

3.1 Problem Statement:

In manufacturing industries, similar products (belonging to a particular category) often need to be continuously transferred from one location to another. We want to accomplish this task using an automatic robot.

3.2 Objective:

The basic idea of our project is to develop a system (robot) that constantly performs the task of picking an object and moving it to the desired location. For automatic operation, our robot would be able to store the movements and then automatically repeat them using a save button on our android application. Moreover, our robot would have the functionality to operate both in manual and automatic modes.

4. Components

The components used in this project are listed below.

4.1 ESP32

ESP32 is a series of low-cost, low-power systems on a chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth. It consists of processors, memory, wireless connectivity, peripheral interfaces (Ethernet, touch sensors, Motor PWM etc.)

Figure 1: ESP32

4.2 Mecanum wheel chassis car

The chassis car kit uses the latest mecanum wheels for movement. It moves in any direction and run perfectly in narrow spaces and on rough ground.

Figure 2: Mecanum wheel chassis car

4.3 TT Motors (Geared DC)

It is a type of DC motor which converts the Direct Current (DC) electrical energy to mechanical energy. The reduction ratio of the

motor is 1:120, output voltage 3-6V. It has higher starting torque, quick starting and stopping, variable speeds with voltage input.

Figure 3: Geared DC Motors

4.4 Robotic arm

A robotic arm is a type of mechanical arm, usually programmable, with similar functions to a human arm; the arm may be the total of the mechanism. The links of such a manipulator are connected by joints allowing either rotational motion (such as an articulated robot) or translational (linear) displacement. The links of the manipulator can be considered to form a kinematic chain.

Figure 4: Robotic Arm

4.5 L298N Motor Driver

Motor drivers acts as an interface between the motors and the control circuits. Motor requires high amount of current whereas the controller circuit works on low current signals. So, the function of motor drivers is to take a low-current control signal and then turn it into a higher-current signal that can drive a motor.

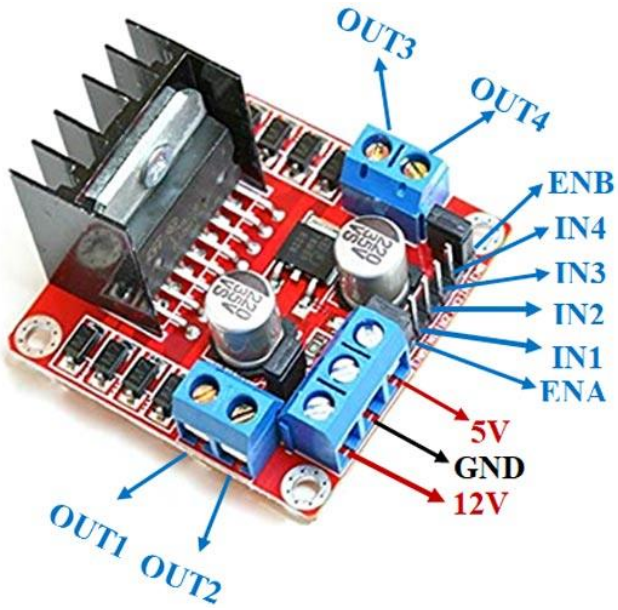

Figure 5: L298N motor driver

5. Methodology

The system consists of a robotic arm with four degrees of freedom to pick up the objects. We mounted an OWI robotic arm to a chassis containing two wheels. We control the motors on the wheels and the robotic arm via an android app developed on our phone that has buttons to rotate the wheels and control the robotic arm. The phone uses Bluetooth to send signals to control the movement of the robotic arm and the robotic vehicle. The Wi-Fi module embedded inside the ESP32-Wrover microcontroller receives this signal and sends it to the microcontroller. The microcontroller decodes the transmitted signal and produces its control signal to drive the motors in the robotic arm and the vehicle via a motor driver (or a couple of motor drivers). A battery/power bank supplies power to the robot, robotic arm, and microcontroller.

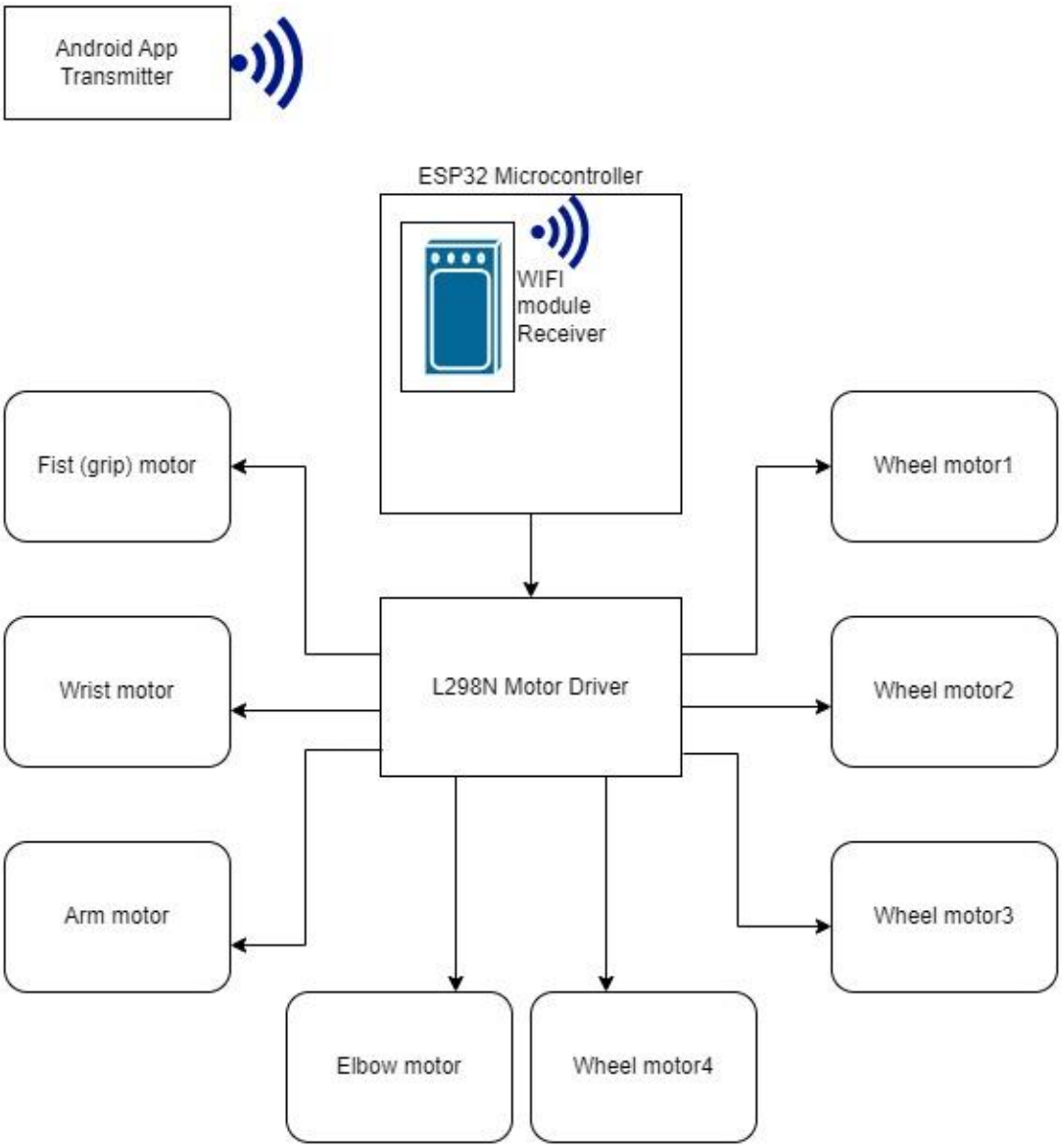

Figure 6: Block Diagram of the system

6. Design Circuits and Snaps

6.1 Overall circuit and the model

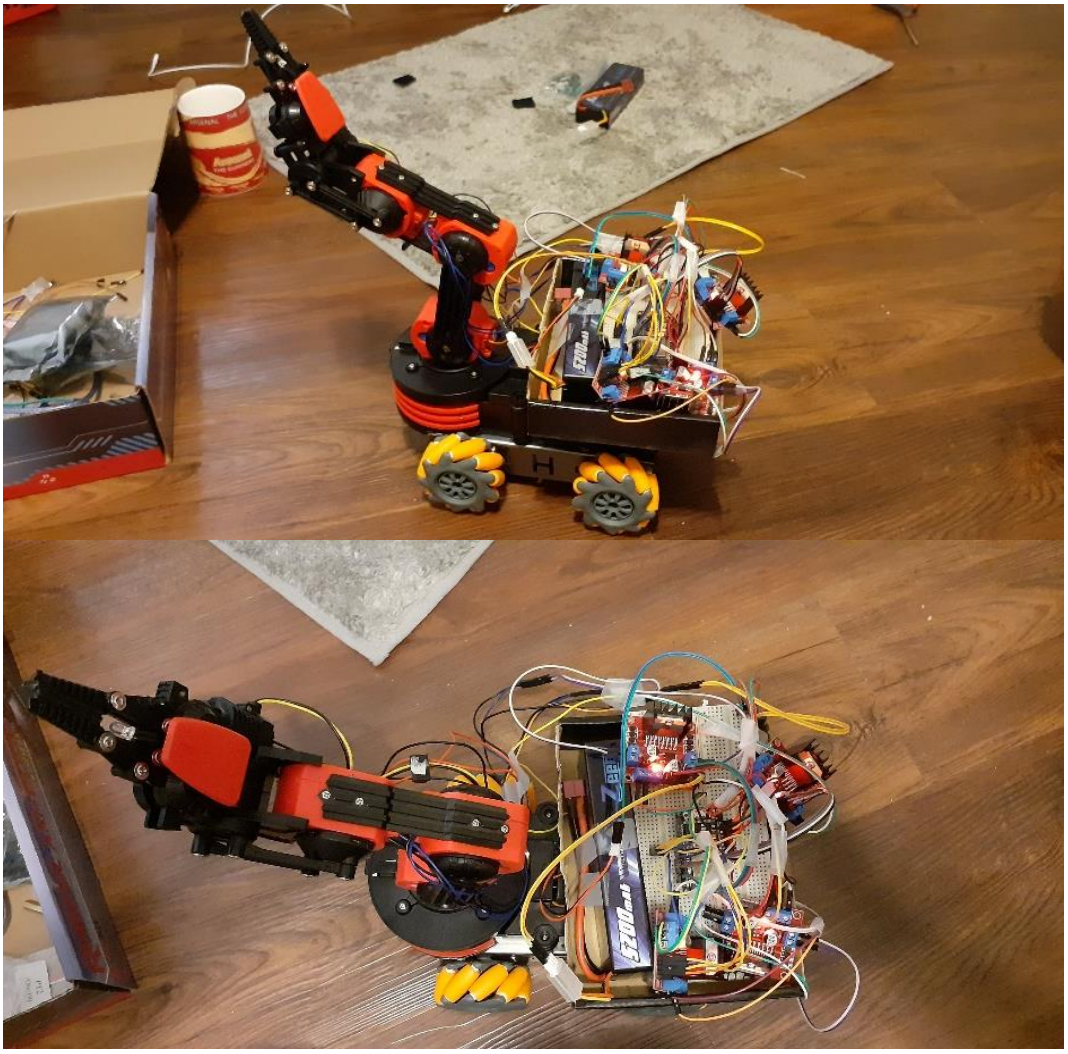

Figure 7: Overall Circuit

6.2 Code for Android app and ESP32 Connection

```
#include <Arduino.h>
#ifdef ESP32
#include <WiFi.h>
#include <AsyncTCP.h>
#elif defined(ESP8266)
#include <ESP8266WiFi.h>
#include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>

#define UP 'g'
#define DOWN 'b'
```

```
#define LEFT 'l'
#define RIGHT 'r'
#define STOP 's'

#define fistOpen 'o'
#define fistClose 'c'
#define wristLeft '0'
#define wristRight '1'
#define armLeft '2'
#define armRight '3'
#define elbowLeft '4'
#define elbowRight '5'
#define baseOn '6'
#define baseOff '7'

#define RIGHT_MOTOR 0
#define LEFT_MOTOR 1

#define fist_motor 2
#define wrist_motor 3
#define arm_motor 4
#define elbow_motor 5

#define FORWARD 1
#define BACKWARD -1

#define rRotate 1
#define lRotate -1

struct MOTOR_PINS {
  int pinIN1;
  int pinIN2;
};

std::vector<MOTOR_PINS> motorPins = {
  { 16, 17 },  //FRONT_RIGHT_MOTOR  //RIGHT_MOTOR
  //{ 18, 19 },  //BACK_RIGHT_MOTOR
```

```
  { 13, 12 },  //FRONT_LEFT_MOTOR     //LEFT_MOTOR
  //{25, 33},  //BACK_LEFT_MOTOR

  { 18, 19 },  //fist/grip
  { 14, 27 },  //wrist
  { 26, 25 },  //Arm
  { 21, 22 },  //elbow
  { 33, 32 }   //Base (start and reset)
};

const char* ssid = "Finity";
const char* password = "Zpqmbty1@";

WiFiServer server(80);



void rotateMotor(int motorNumber, int motorDirection) {
  if (motorDirection == FORWARD) {
    digitalWrite(motorPins[motorNumber].pinIN1, HIGH);
    digitalWrite(motorPins[motorNumber].pinIN2, LOW);
  } else if (motorDirection == BACKWARD) {
    digitalWrite(motorPins[motorNumber].pinIN1, LOW);
    digitalWrite(motorPins[motorNumber].pinIN2, HIGH);
  } else {
    digitalWrite(motorPins[motorNumber].pinIN1, LOW);
    digitalWrite(motorPins[motorNumber].pinIN2, LOW);
  }
}

void processAuto(char c_) {
  char temp[16] = { 'g', 'c', 'c', 'b', 'b', 'r', 'o', 'o', 'l' };
  for (int i = 0; i < 16; i++) {
    processCarMovement(temp[i]);
    delay(200);
    processCarMovement(STOP);
  }
```

```
}

void processCarMovement(char inputValue) {
  //Serial.println(inputValue);
  switch (inputValue) {

    case UP:
      rotateMotor(RIGHT_MOTOR, FORWARD);
      rotateMotor(LEFT_MOTOR, FORWARD);
      break;

    case DOWN:
      rotateMotor(RIGHT_MOTOR, BACKWARD);
      rotateMotor(LEFT_MOTOR, BACKWARD);
      break;

    case LEFT:
      rotateMotor(RIGHT_MOTOR, FORWARD);
      rotateMotor(LEFT_MOTOR, BACKWARD);
      break;

    case RIGHT:
      rotateMotor(RIGHT_MOTOR, BACKWARD);
      rotateMotor(LEFT_MOTOR, FORWARD);
      break;

    case STOP:
      rotateMotor(RIGHT_MOTOR, STOP);
      rotateMotor(LEFT_MOTOR, STOP);
      rotateMotor(fist_motor, STOP);
      rotateMotor(wrist_motor, STOP);
      rotateMotor(arm_motor, STOP);
      rotateMotor(elbow_motor, STOP);
      break;

    case fistOpen:
      rotateMotor(fist_motor, FORWARD);
```

```
  break;

case fistClose:
  rotateMotor(fist_motor, BACKWARD);
  break;

case wristLeft:
  rotateMotor(wrist_motor, FORWARD);
  break;

case wristRight:
  rotateMotor(wrist_motor, BACKWARD);
  break;

case armLeft:
  rotateMotor(arm_motor, FORWARD);
  break;

case armRight:
  rotateMotor(arm_motor, BACKWARD);
  break;

case elbowLeft:
  rotateMotor(elbow_motor, FORWARD);
  break;

case elbowRight:
  rotateMotor(elbow_motor, BACKWARD);
  break;

default:
  rotateMotor(RIGHT_MOTOR, STOP);
  rotateMotor(LEFT_MOTOR, STOP);
  rotateMotor(fist_motor, STOP);
  rotateMotor(wrist_motor, STOP);
  rotateMotor(arm_motor, STOP);
  rotateMotor(elbow_motor, STOP);
```

```
      break;
    }
  }



void setUpPinModes() {
  for (int i = 0; i < motorPins.size(); i++) {
    pinMode(motorPins[i].pinIN1, OUTPUT);
    pinMode(motorPins[i].pinIN2, OUTPUT);
    rotateMotor(i, STOP);
  }
}



void setup(void) {
  setUpPinModes();
  Serial.begin(115200);
  pinMode(2, OUTPUT);  // set the LED pin mode

  delay(10);

  // We start by connecting to a WiFi network

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
```

```
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  server.begin();

  Serial.println("HTTP server started");

  Serial.println("********************");
  Serial.println("Robot Commands");
  Serial.println("Menu...");
  Serial.println("Enter: 0-7 for robot arm, o/c for fist open and close,
g/s/l/r/b for wheels ");
  Serial.println("********************");
}

void loop() {
  WiFiClient client = server.available();  // listen for incoming clients

  if (client) {                     // if you get a client,
    Serial.println("New Client.");  // print a message out the serial port
    String currentLine = "";        // make a String to hold incoming data
from the client
    while (client.connected()) {    // loop while the client's connected
      if (client.available()) {     // if there's bytes to read from the client,
        char c_ = client.read();    // read a byte, then
        Serial.write(c_);           // print it out the serial monitor
        if (c_ == '\n') {           // if the byte is a newline character

          // if the current line is blank, you got two newline characters in
a row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g.
HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a
blank line:
```

```
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println();

            // the content of the HTTP response follows the header:
            // client.print("Click <a href=\"/H\">here</a> to turn the LED
on pin 5 on.<br>");
            // client.print("Click <a href=\"/L\">here</a> to turn the LED
on pin 5 off.<br>");

            // The HTTP response ends with another blank line:
            client.println();
            // break out of the while loop:
            break;
          } else {  // if you got a newline, then clear currentLine:
            currentLine = "";
          }
        } else if (c_ != '\r') {  // if you got anything else but a carriage
return character,
          currentLine += c_;      // add it to the end of the currentLine
        }

        Serial.println();

        Serial.println("c_:");
        Serial.println(c_);

        if (c_ == '6') {
          processAuto(c_);
        } else if (c_ == '7') {
          processCarMovement(STOP);
        } else {
          processCarMovement(c_);
          //delay(200);
          //processCarMovement(STOP);
        }
      }
```

```
    }
  }
}
```

7. Advantages, Disadvantages and Applications

7.1 Advantages:

- Decent battery backup which makes it a mobile device to be used in any situation without requiring power supply from the mains.
- Mobile app control makes the model suitable for replacing direct human contact for hazardous tasks such as chemical handling, bomb disposal, etc.
- Commendable weight carrying capacity of 5kgs.
- Mecanum wheels are used in chassis which makes it suitable for operation on any terrain.
- Lightweight design which makes the model appropriate to be used in operations that require constant location change.
- WIFI control that enables real time robotic operation from a different location to the robot.
- User friendly android app interface which makes controlling the robot straightforward and uncomplicated**.**

7.2 Disadvantages:

- The robotic arm claw has only two grips which makes it unsuitable for picking up any circular objects (for example: a ball).
- Application in the industry is limited to only lightweight object lifting and maneuvering since the model has a weight carrying capacity of 5kgs.
- Uptime is reduced drastically in extreme cold climates since battery backup is reduced in such climatic conditions.
- Mobile app control limited to Android, no IOS support.
- Improper shielding makes the model unsuitable to be operated outdoors during rain, since there is potential risk of water damage to the esp32 and other sensors.

7.3 Applications:

- Suited for use by Bomb Disposal Squad.
- Remotely pick up and move unidentified objects, which can be a bomb, to a secure location away from vicinity of people for safe disposal.

- The safety of the personnel is ensured as they do not come in direct contact with the threat object.
- Our model being battery powered makes it a mobile device which can be used in any setting.
- Our model can be implemented into large warehouses to pick up and take goods to the delivery area.
- Single operator can remotely control the robot from a central control room.

8. Discussion

We used ESP32 because it was cheaper than Raspberry Pi and could perform multiple functions this reduced the cost for a microcontroller. It is also feasible to use ESP32 for low power small applications. Our system has both Automatic as well as manual mode of operation and features combination of both robotic arm and a car.

8.1 Problems Encountered

Initially, the android app was designed to control the robot via Bluetooth; however, due to the frequent disconnections of BLE to save energy we had to switch to wifi control. The whole application development was done two times. The wifi one worked very well.

There were lots of motors in our system. The battery was not sufficient to drive every motor. Motor drivers' pins were insufficient. So, we had to order Lipo batteries, as well as extra motor drivers, which came late and delayed our project.

This reduced precision in wheel control a little bit but still performed very well.

These problems delayed our project completion, but we addressed all the problems and successfully completed it.

8.2 Creative solution

The motor driver output pins and esp32 GPIO pins were limited in number. So, we reduced the number of output pins by providing the same control signals to the two motors on left and two motors on right.

Classic Bluetooth in ESP32 needed some extra authentication and security challenges in app development. With Bluetooth Low Power (BLE) in the ESP32 microcontroller, we encountered frequent disconnections (as BLE wants to save power by turning itself off and

operating only when required). We solved this problem by using Wi-Fi instead of Bluetooth.

8.3 Teamwork and Management

Consistency, motivation, desire to learn, and team coordination really helped us addressing the issues in app development as well as hardware design that require interfacing of devices with different power requirements.

9. Conclusion

A vehicular robotic arm that constantly performs the task of picking an object and moving it to the desired location was developed. It was controlled via an Android mobile app. For automatic operation, our robot was able to store the movements and then automatically repeat them using a save button on our android application. Moreover, our robot has the functionality to operate both in manual and automatic modes. This prototype may be expected to overcome the problems of picking hazardous objects or non-hazardous objects that are far away from the user and where displacement of very heavy objects is needed from one place to another as automation is required in many industries.