Mechatronics Assignment

GROUP MEMBERS:

- MUHAMMAD ABEER KHAN (ME-19221)
- MUHAMMAD RAFEY NAEEM (ME-19223)
- HUZEFA-BIN-AQEEL (ME-19225)
- OSAMA MAHFOOZ (ME-19231)
- ARHUM AHMED (ME-19248)

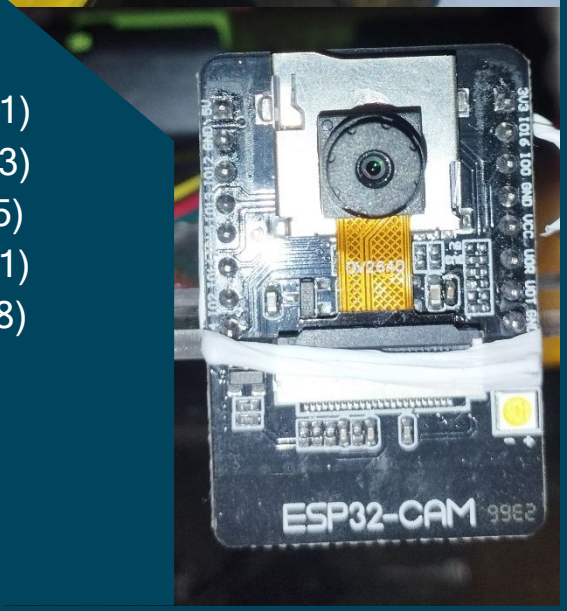


Table of Contents

SURVEILLANCE CAR USING ESP-32 CAM MODULE

Introduction:

The evolving industry of automotive is heading rapidly towards the applications of the concept of self-driven automated cars. Behind this evolving industry, the basis is being formed by the integrated systems of mechatronics. In the correlation of these mechatronics systems and the evolving automotive industry, our project aims towards a very specific feature of these systems, which is assembled and designed as a prototype for mainly the surveillance purposes. As a prototype, our car has been designed as a remote-controlled surveillance car.

The ESP32 surveillance car project aims to design and develop a versatile and efficient vehicle equipped with the ESP32 microcontroller for surveillance tasks. The ESP32 microcontroller offers powerful capabilities, including Wi-Fi connectivity, dual-core processing, and adequate GPIO pins, making it an ideal choice for integrating advanced surveillance technology.

Scope:

The project aims to create a surveillance car that can operate in various environments, providing real-time monitoring, threat detection, and situational awareness. This involves developing algorithms and implementing software that can analyze sensor data, perform object recognition, track movements, and identify potential security breaches.

Components:

4WD Car Kit:

4WD car kit is used as chassis of the car. The 4WD Car kit is a versatile platform that includes four motors, four wheels, and connectors, providing a solid foundation for building a four-wheel-drive vehicle. With its four motors, the kit offers powerful propulsion and torque, allowing the car to traverse various terrains with ease.

DC Motors:

Four DC motors are used in the maneuvering and driving of car. The voltage range of DC motors can vary depending on the specific model and manufacturer. However, commonly used DC motors typically operate within a voltage range of **3V to 12V**.

Voltage	DC 6V
Current	120mA
Reduction Ratio	48:1
RPM (Tire)	240
Tire Diameter	66mm

ESP 32:

The ESP32 is a series of low-cost, low-power System on a Chip (SoC) microcontrollers developed by Espressif Systems. It is a successor to the ESP8266 and includes a number of new features, such as a dual-core 32-bit processor, integrated **Wi-Fi and Bluetooth**, Bluetooth Low Energy (BLE), **160MHz** clock speed, **4MB** of onboard flash memory, **520KB** of **SRAM**, and support for multiple operating systems, including **Arduino**, MicroPython, and FreeRTOS.

The ESP32 is well-suited for a wide range of applications, including Internet of Things (IoT) devices, smart home devices, wearables, robotics, audio and video streaming, and industrial automation. It is a popular choice for IoT developers because of its low cost, small size, low power consumption, and ease of programming.

Low cost: The ESP32 is a low-cost microcontroller, making it a cost-effective solution for many IoT projects.

Small size: The ESP32 is a small, compact microcontroller, making it ideal for embedded applications.

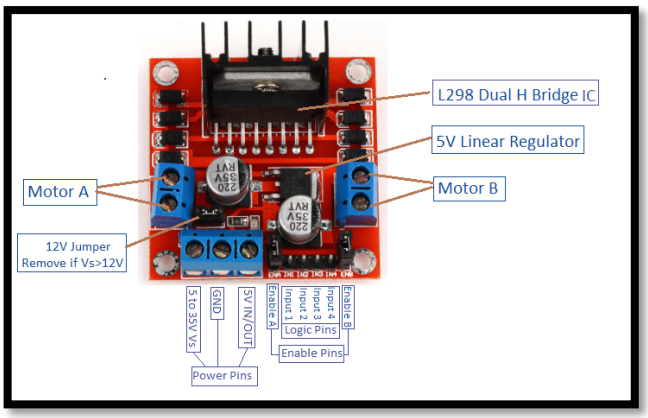
Low power consumption: The ESP32 has low power consumption, making it ideal for battery-powered devices.

Easy to program: The ESP32 is easy to program, making it a good choice for beginners.

Wide range of applications: The ESP32 can be used for a wide range of applications, making it a versatile microcontroller.

L298N motor driver module:

The L298 motor driver is a dual H-bridge motor driver that allows you to control the speed and direction of four DC motors. It is a popular choice for robotics and other projects that require precise motor control. The L298 motor driver has a number of features that make it a popular choice for motor control applications. These features include:



- **Robust integrated circuit:** The L298 motor driver is a robust integrated circuit (IC) that contains all of the circuitry necessary to control four DC motors. It has two H-bridges, each of which can be used to control a single motor. An H-bridge is a circuit that can be used to switch the direction of current flow through a motor, allowing you to control the motor's speed and direction.
- **High current output:** The L298 motor driver can output up to **2A** of current per channel, making it suitable for driving even the most powerful DC motors.
- **Low voltage operation:** The L298 motor driver can operate from a voltage range of **5V to 35V**, making it compatible with a wide range of power supplies.
- **Simple to use:** The L298 motor driver is relatively simple to use, making it a good choice for beginners.
- **Wide range of applications:** The L298 motor driver can be used in a wide range of applications, including robotics, drones, and industrial automation.

Some features of L298 motor driver module are:

- **Heatsink:** The L298 motor driver has a built-in heatsink that helps to dissipate heat, making it suitable for driving high-powered motors.
- **Protection:** The L298 motor driver has a number of protection features, including overcurrent protection, undervoltage protection, and thermal protection.
- **Availability:** The L298 motor driver is a widely available component, making it easy to purchase.

Arduino UNO:

In this project Arduino UNO is only use for the uploading of code in the ESP32 CAM module, because there is no direct way to upload code on ESP32 CAM module. Moreover, different microcontroller like NodeMCU can be use.

Batteries:

Three Lithium-ion Batteries of **3.7V** are used to provide power to the ESP32, motors, and motor driver. These batteries are placed in the battery holder to provide power.

Jumper Wires:

They are used to make the full circuitry of the whole system to drive surveillance car. Multiple male to male, male to female and female to female jumper wires are used.

Software Architecture:

We have used Arduino UNO for the uploading of code in the ESP32 CAM module. NodeMCU can also be used for the same purpose.

Following steps are followed for the uploading of code on ESP32 using Arduino UNO.

1) Latest version of **Arduino IDE** is installed and used. (https://www.arduino.cc/en/software)

2) Install **ESP32 Board** Support Package: Open the Arduino IDE and go to "File" > "Preferences." In the "Additional Boards Manager URLs" field, add the following URL: (https://dl.espressif.com/dl/package_esp32_index.json). Click "OK" to save the preferences.

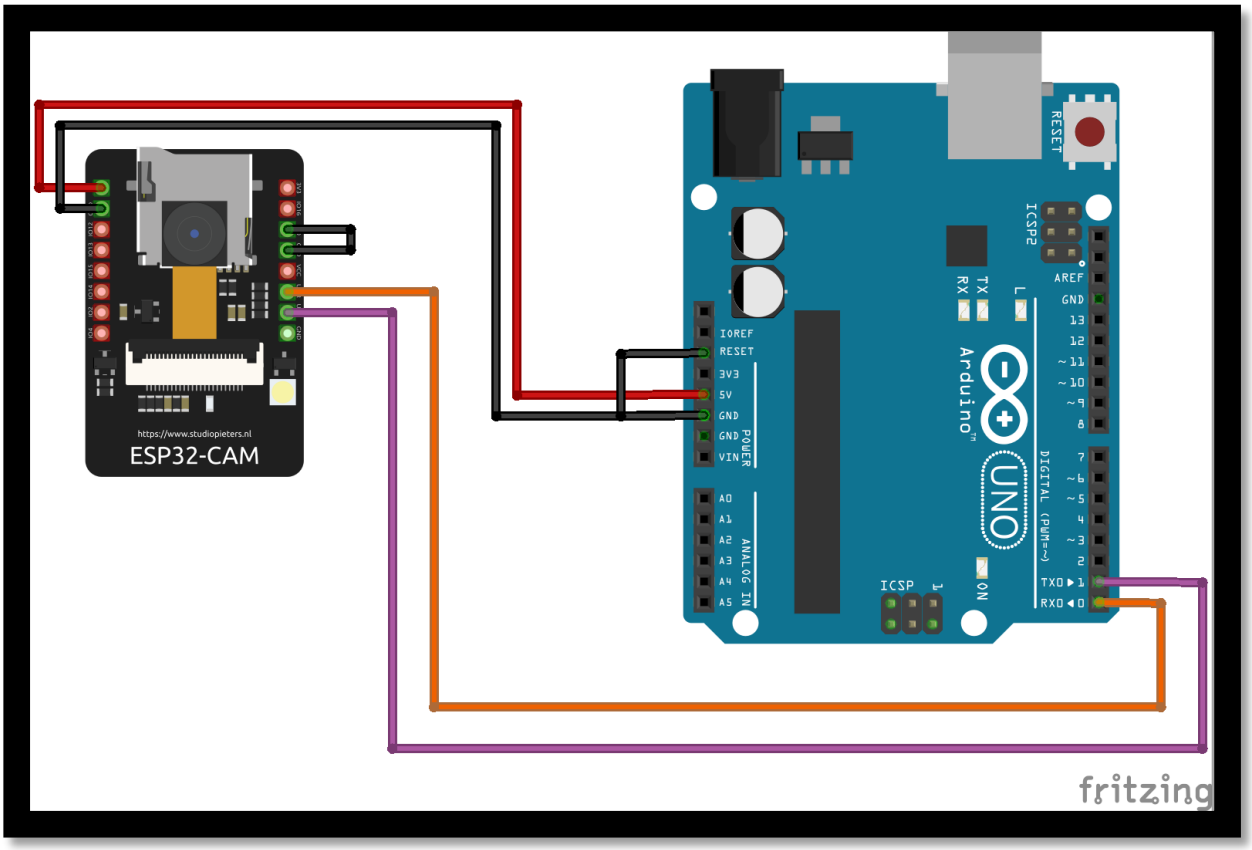
3) Install **Async TCP Library** (https://github.com/me-no-dev/AsyncTCP) Go to “sketch”> “Add.ZIP Library”> “Async TCP”

4) Install **ESPAsyncWebServer Library** (https://github.com/me-no-dev/ESPAsyncWebServer) Go to “sketch”> “Add.ZIP Library”> “ESPAsyncWebServer Library”

5) Select the ESP32 Board, go to "Tools" > "Board" and select "**ESP32 Dev Module**" board from the list.

6) Connect Arduino to the computer. Select COM and "ESP32 Dev Module".

Circuit Diagram for Code Uploading:



Code Uploading (Pin connection)	
ESP32 CAM module	**Arduino**
VOR	Rx
VOT	Tx
GND	GND
5V	5V
IO0---GND	-
-	RESET---GND

Code:

```
#include "esp_camera.h"
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <iostream>
#include <sstream>

struct MOTOR_PINS
{
  int pinEn;
  int pinIN1;
  int pinIN2;
};

std::vector<MOTOR_PINS> motorPins =
{
  {12, 13, 15},  //RIGHT_MOTOR Pins (EnA, IN1, IN2)
  {12, 14, 2},  //LEFT_MOTOR  Pins (EnB, IN3, IN4)
};
#define LIGHT_PIN 4

#define UP 1
#define DOWN 2
#define LEFT 3
#define RIGHT 4
#define STOP 0

#define RIGHT_MOTOR 0
#define LEFT_MOTOR 1

#define FORWARD 1
#define BACKWARD -1

const int PWMFreq = 1000; /* 1 KHz */
const int PWMResolution = 8;
const int PWMSpeedChannel = 2;
const int PWMLightChannel = 3;

//Camera related constants
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
```

```
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

const char* ssid     = "Gizli Boyz";
const char* password = "19248";

AsyncWebServer server(80);
AsyncWebSocket wsCamera("/Camera");
AsyncWebSocket wsCarInput("/CarInput");
uint32_t cameraClientId = 0;

const char* htmlHomePage PROGMEM = R"HTMLHOMEPAGE(
<!DOCTYPE html>
<html>
  <head>
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-
scalable=no">
    <style>
    .arrows {
      font-size:40px;
      color:red;
    }
    td.button {
      background-color:black;
      border-radius:25%;
      box-shadow: 5px 5px #888888;
    }
    td.button:active {
      transform: translate(5px,5px);
      box-shadow: none;
    }
    .noselect {
      -webkit-touch-callout: none; /* iOS Safari */
        -webkit-user-select: none; /* Safari */
         -khtml-user-select: none; /* Konqueror HTML */
           -moz-user-select: none; /* Firefox */
            -ms-user-select: none; /* Internet Explorer/Edge */
                user-select: none; /* Non-prefixed version, currently
                                      supported by Chrome and Opera */
    }
```

```
 .slidecontainer {
   width: 100%;
 }
 .slider {
   -webkit-appearance: none;
   width: 100%;
   height: 15px;
   border-radius: 5px;
   background: #d3d3d3;
   outline: none;
   opacity: 0.7;
   -webkit-transition: .2s;
   transition: opacity .2s;
 }
 .slider:hover {
   opacity: 1;
 }

 .slider::-webkit-slider-thumb {
   -webkit-appearance: none;
   appearance: none;
   width: 25px;
   height: 25px;
   border-radius: 50%;
   background: red;
   cursor: pointer;
 }
 .slider::-moz-range-thumb {
   width: 25px;
   height: 25px;
   border-radius: 50%;
   background: red;
   cursor: pointer;
 }
 </style>

</head>
<body class="noselect" align="center" style="background-color:white">

 <!--h2 style="color: teal;text-align:center;">Wi-Fi Camera &#128663; Control</h2-->

 <table id="mainTable" style="width:400px;margin:auto;table-layout:fixed" CELLSPACING=10>
   <tr>
     <img id="cameraImage" src="" style="width:400px;height:250px"></td>
   </tr>
   <tr>
     <td></td>
```

```
      <td class="button" ontouchstart='sendButtonInput("MoveCar","1")'
ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8679;</span></td>
      <td></td>
     </tr>
     <tr>
      <td class="button" ontouchstart='sendButtonInput("MoveCar","3")'
ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8678;</span></td>
      <td class="button"></td>
      <td class="button" ontouchstart='sendButtonInput("MoveCar","4")'
ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8680;</span></td>
     </tr>
     <tr>
      <td></td>
      <td class="button" ontouchstart='sendButtonInput("MoveCar","2")'
ontouchend='sendButtonInput("MoveCar","0")'><span class="arrows" >&#8681;</span></td>
      <td></td>
     </tr>
     <tr/><tr/>
     <tr>
      <td style="text-align:left"><b>Speed:</b></td>
      <td colspan=2>
       <div class="slidecontainer">
         <input type="range" min="0" max="255" value="150" class="slider" id="Speed"
oninput='sendButtonInput("Speed",value)'>
       </div>
      </td>
     </tr>
     <tr>
      <td style="text-align:left"><b>Light:</b></td>
      <td colspan=2>
       <div class="slidecontainer">
         <input type="range" min="0" max="255" value="0" class="slider" id="Light"
oninput='sendButtonInput("Light",value)'>
       </div>
      </td>
     </tr>
   </table>

   <script>
    var webSocketCameraUrl = "ws:\/\/" + window.location.hostname + "/Camera";
    var webSocketCarInputUrl = "ws:\/\/" + window.location.hostname + "/CarInput";
    var websocketCamera;
    var websocketCarInput;

    function initCameraWebSocket()
    {
     websocketCamera = new WebSocket(webSocketCameraUrl);
     websocketCamera.binaryType = 'blob';
```

```
      websocketCamera.onopen    = function(event){};
      websocketCamera.onclose   = function(event){setTimeout(initCameraWebSocket, 2000);};
      websocketCamera.onmessage = function(event)
      {
        var imageId = document.getElementById("cameraImage");
        imageId.src = URL.createObjectURL(event.data);
      };
    }

    function initCarInputWebSocket()
    {
      websocketCarInput = new WebSocket(webSocketCarInputUrl);
      websocketCarInput.onopen    = function(event)
      {
        var speedButton = document.getElementById("Speed");
        sendButtonInput("Speed", speedButton.value);
        var lightButton = document.getElementById("Light");
        sendButtonInput("Light", lightButton.value);
      };
      websocketCarInput.onclose   = function(event){setTimeout(initCarInputWebSocket, 2000);};
      websocketCarInput.onmessage = function(event){};
    }

    function initWebSocket()
    {
      initCameraWebSocket ();
      initCarInputWebSocket();
    }
    function sendButtonInput(key, value)
    {
      var data = key + "," + value;
      websocketCarInput.send(data);
    }

    window.onload = initWebSocket;
    document.getElementById("mainTable").addEventListener("touchend", function(event){
      event.preventDefault()
    });
   </script>
  </body>
</html>
)HTMLHOMEPAGE";


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 moveCar(int inputValue)
{
  Serial.printf("Got value as %d\n", 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);
      break;

    default:
      rotateMotor(RIGHT_MOTOR, STOP);
      rotateMotor(LEFT_MOTOR, STOP);
```

```
      break;
  }
}

void handleRoot(AsyncWebServerRequest *request)
{
  request->send_P(200, "text/html", htmlHomePage);
}

void handleNotFound(AsyncWebServerRequest *request)
{
    request->send(404, "text/plain", "File Not Found");
}

void onCarInputWebSocketEvent(AsyncWebSocket *server,
                      AsyncWebSocketClient *client,
                      AwsEventType type,
                      void *arg,
                      uint8_t *data,
                      size_t len)
{
  switch (type)
  {
    case WS_EVT_CONNECT:
      Serial.printf("WebSocket client #%u connected from %s\n", client->id(), client-
>remoteIP().toString().c_str());
      break;
    case WS_EVT_DISCONNECT:
      Serial.printf("WebSocket client #%u disconnected\n", client->id());
      moveCar(0);
      ledcWrite(PWMLightChannel, 0);
      break;
    case WS_EVT_DATA:
      AwsFrameInfo *info;
      info = (AwsFrameInfo*)arg;
      if (info->final && info->index == 0 && info->len == len && info->opcode == WS_TEXT)
      {
        std::string myData = "";
        myData.assign((char *)data, len);
        std::istringstream ss(myData);
        std::string key, value;
        std::getline(ss, key, ',');
        std::getline(ss, value, ',');
        Serial.printf("Key [%s] Value[%s]\n", key.c_str(), value.c_str());
        int valueInt = atoi(value.c_str());
        if (key == "MoveCar")
        {
          moveCar(valueInt);
```

```
        }
        else if (key == "Speed")
        {
          ledcWrite(PWMSpeedChannel, valueInt);
        }
        else if (key == "Light")
        {
          ledcWrite(PWMLightChannel, valueInt);
        }
      }
      break;
    case WS_EVT_PONG:
    case WS_EVT_ERROR:
      break;
    default:
      break;
  }
}

void onCameraWebSocketEvent(AsyncWebSocket *server,
                            AsyncWebSocketClient *client,
                            AwsEventType type,
                            void *arg,
                            uint8_t *data,
                            size_t len)
{
  switch (type)
  {
    case WS_EVT_CONNECT:
      Serial.printf("WebSocket client #%u connected from %s\n", client->id(), client-
>remoteIP().toString().c_str());
      cameraClientId = client->id();
      break;
    case WS_EVT_DISCONNECT:
      Serial.printf("WebSocket client #%u disconnected\n", client->id());
      cameraClientId = 0;
      break;
    case WS_EVT_DATA:
      break;
    case WS_EVT_PONG:
    case WS_EVT_ERROR:
      break;
    default:
      break;
  }
}

void setupCamera()
```

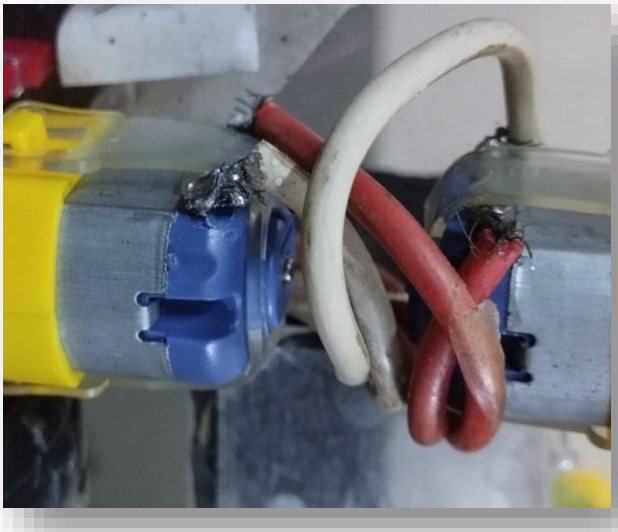
```
{
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 10;
  config.fb_count = 1;

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK)
  {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  if (psramFound())
  {
    heap_caps_malloc_extmem_enable(20000);
    Serial.printf("PSRAM initialized. malloc to take memory from psram above this size");
  }
}

void sendCameraPicture()
{
  if (cameraClientId == 0)
  {
    return;
  }
```

```
  unsigned long  startTime1 = millis();
  //capture a frame
  camera_fb_t * fb = esp_camera_fb_get();
  if (!fb)
  {
     Serial.println("Frame buffer could not be acquired");
     return;
  }

  unsigned long  startTime2 = millis();
  wsCamera.binary(cameraClientId, fb->buf, fb->len);
  esp_camera_fb_return(fb);

  //Wait for message to be delivered
  while (true)
  {
    AsyncWebSocketClient * clientPointer = wsCamera.client(cameraClientId);
    if (!clientPointer || !(clientPointer->queueIsFull()))
    {
      break;
    }
    delay(1);
  }

  unsigned long  startTime3 = millis();
  Serial.printf("Time taken Total: %d|%d|%d\n",startTime3 - startTime1, startTime2 - startTime1,
startTime3-startTime2 );
}

void setUpPinModes()
{
  //Set up PWM
  ledcSetup(PWMSpeedChannel, PWMFreq, PWMResolution);
  ledcSetup(PWMLightChannel, PWMFreq, PWMResolution);

  for (int i = 0; i < motorPins.size(); i++)
  {
    pinMode(motorPins[i].pinEn, OUTPUT);
    pinMode(motorPins[i].pinIN1, OUTPUT);
    pinMode(motorPins[i].pinIN2, OUTPUT);

    /* Attach the PWM Channel to the motor enb Pin */
    ledcAttachPin(motorPins[i].pinEn, PWMSpeedChannel);
  }
  moveCar(STOP);

  pinMode(LIGHT_PIN, OUTPUT);
  ledcAttachPin(LIGHT_PIN, PWMLightChannel);
```

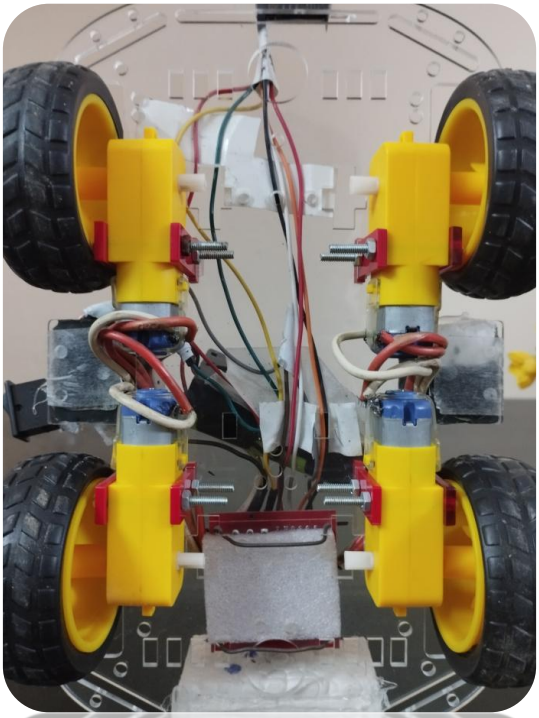
```
}


void setup(void)
{
  setUpPinModes();
  Serial.begin(115200);

  WiFi.softAP(ssid, password);
  IPAddress IP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(IP);

  server.on("/", HTTP_GET, handleRoot);
  server.onNotFound(handleNotFound);

  wsCamera.onEvent(onCameraWebSocketEvent);
  server.addHandler(&wsCamera);

  wsCarInput.onEvent(onCarInputWebSocketEvent);
  server.addHandler(&wsCarInput);

  server.begin();
  Serial.println("HTTP server started");

  setupCamera();
}

void loop()
{
  wsCamera.cleanupClients();
  wsCarInput.cleanupClients();
  sendCameraPicture();
  Serial.printf("SPIRam Total heap %d, SPIRam Free Heap %d\n", ESP.getPsramSize(),
ESP.getFreePsram());
}
```

Hardware Architecture:

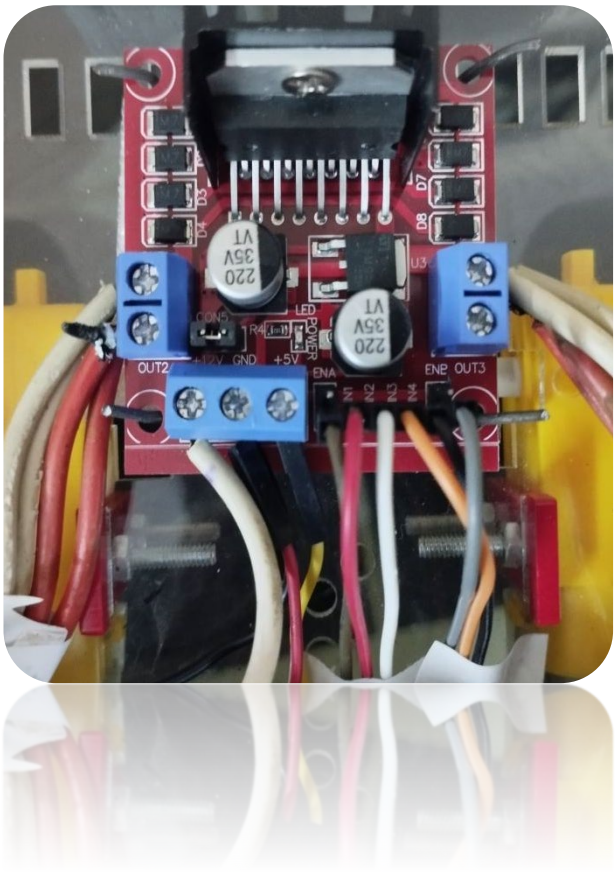
1) Firstly, all the DC motors are soldered with wires for tight and long-term connections.



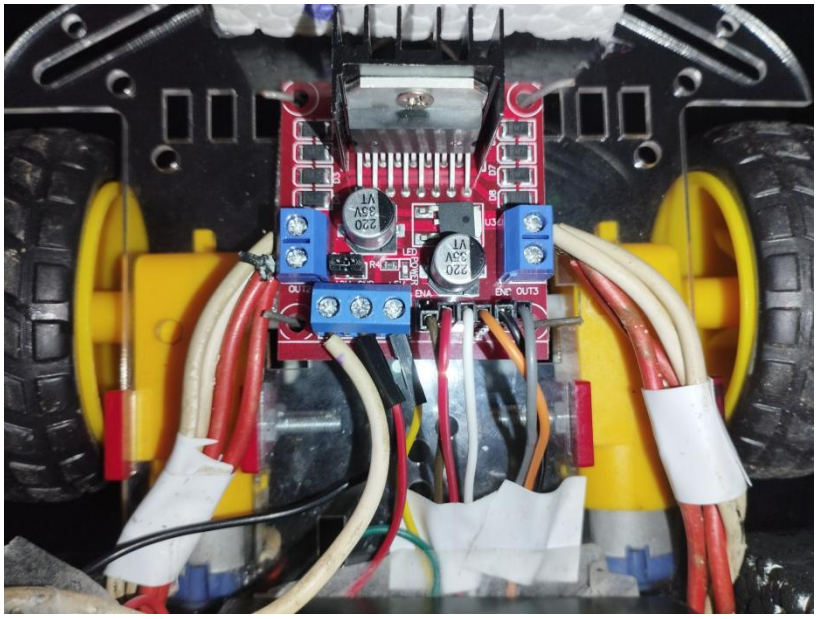
2) All the 4 motors are mounted at the opposites end of the chassis of the car.



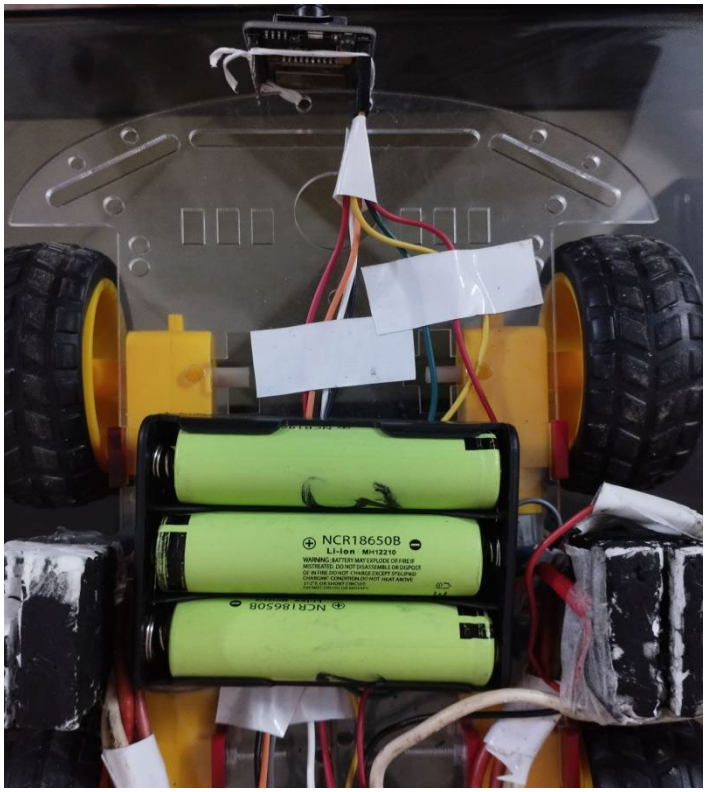
3) L298 Motor Driver is mounted on the rear end of the chassis.



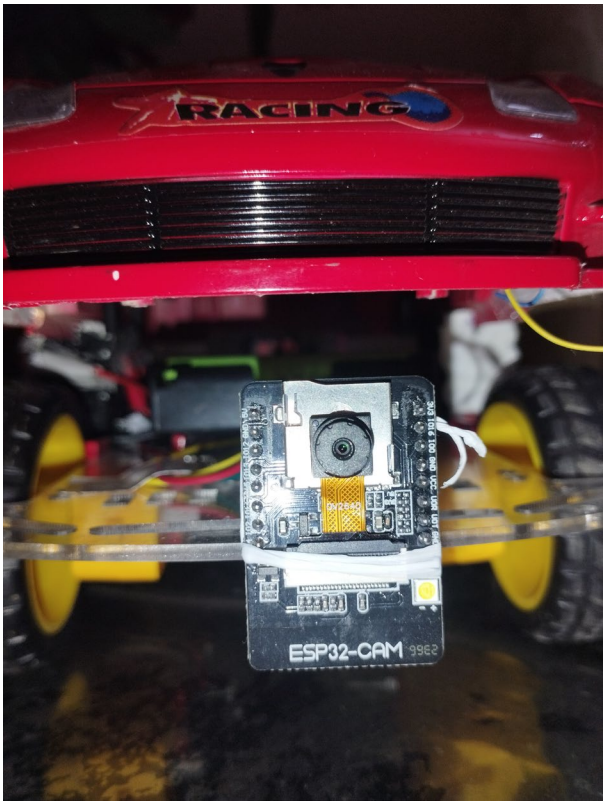
4) L298 motor connections are done with the DC motor. **Right side motors** are connected to the **OUT1** and **OUT2** pins of the motor driver. **Left side motors** are connected to the **OUT3** and **OUT4** pins of the motor driver.



5) DC Batteries' holder is assembled in the middle of chassis of car.



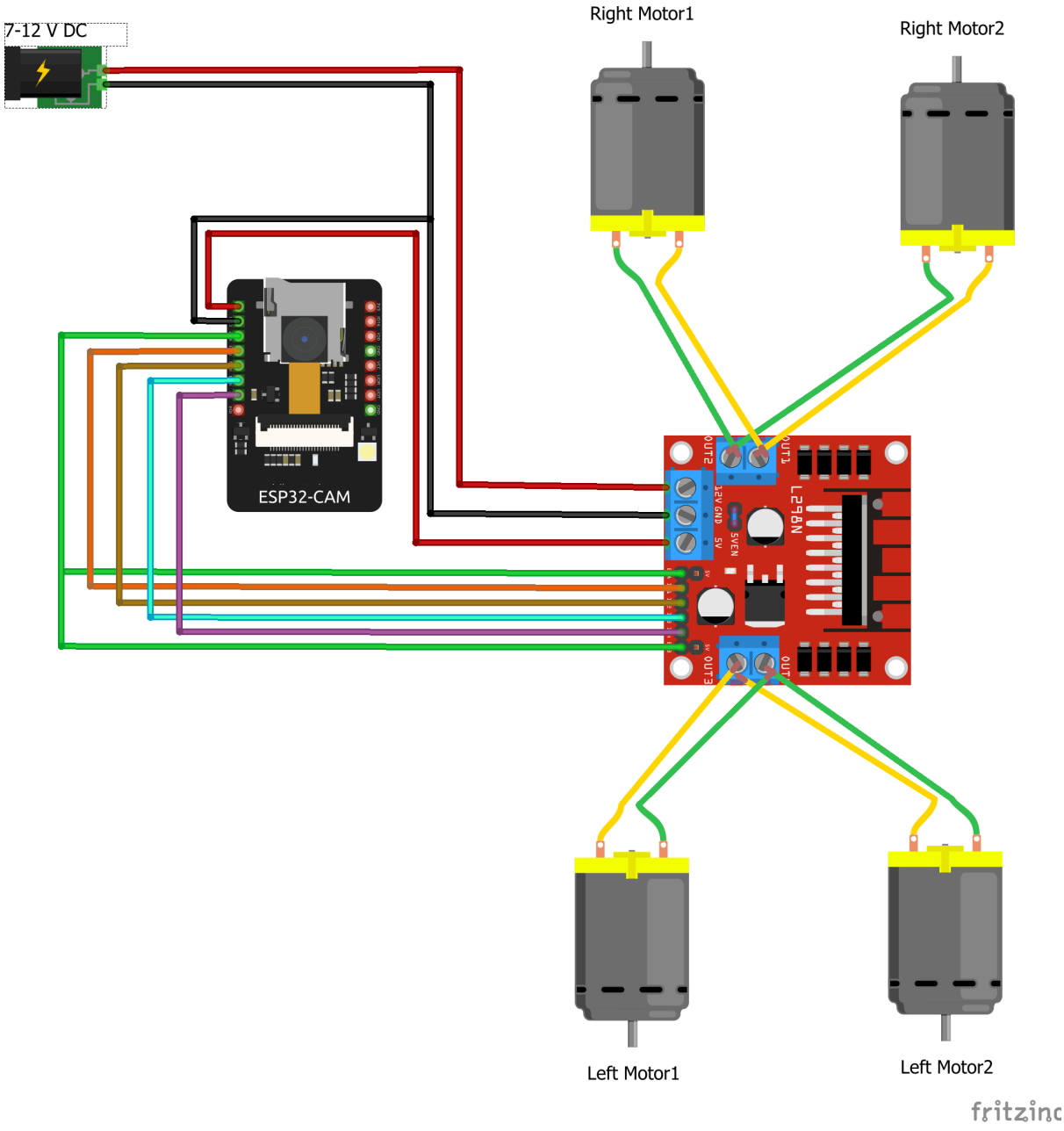
6) DC Batteries' holder wires are connected to the **5V** and **GND pin** present on the L298 motor driver.
7) Now, ESP32 module is assemble in the front of chassis and the circuit connection to motor driver are shown below in the table as well as circuit diagram.



Car Connections (Pin Connection)	
ESP32 CAM module	L298 Motor Driver
5V	5V
GND	GND
IO2	IN4
IO12	enA
IO12	enB
IO13	IN1
IO14	IN3
IN15	IN2

8) All the circuits and wire management are done and the car casing is place on the top.

Circuit Diagram for Car Connections:



Working:

- To initiate the working process for the car first we need to press the power button to start the supply to the motor driver.
- Next the motor driver module will transmit the voltage supply to the 4 motors that are installed and to the Esp cam module also and it starts.
- The next step is the connectivity in which we connect the Wi-Fi transmitted by the module to our smart devices (smartphone, laptop etc. anything with Wi-Fi receiver capability) by using the Ip address on our browser 193.168.4.1 after this a screen pops with our cam module feed along with car driving controls.
- We can now control our car through our mobile by using the controls displayed.

SCREENSHOTS FROM MOBILE: