ESP 32 Cam RC Car

Submitted To

DR. Tarek Elewa

Submitted By

Mohamed Ahmed Rabea

Mohamed Ahmed Saleh

Mohamed Ahmed Ibrahim

Muhammad Usama Aowad

Mohamed Osama Mohamed

Mohamed Saad abd El Naiem

Maher Shehta Abdo

Components

1-CAR CHASIS

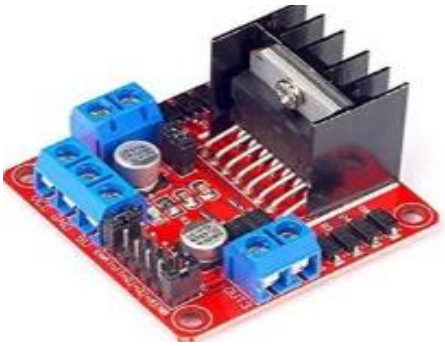
2-TT GEAR MOTORS

3-WHEELS

4-CONNECTORS&SCREWS

5-ESP32 CAM

6-L298N MOTOR DRIVER



8-ARDUINO UNO

9-JUMPER WIRES

10-BATTERIES

11- A SWITCH

ABOUT THIS PROJECT:

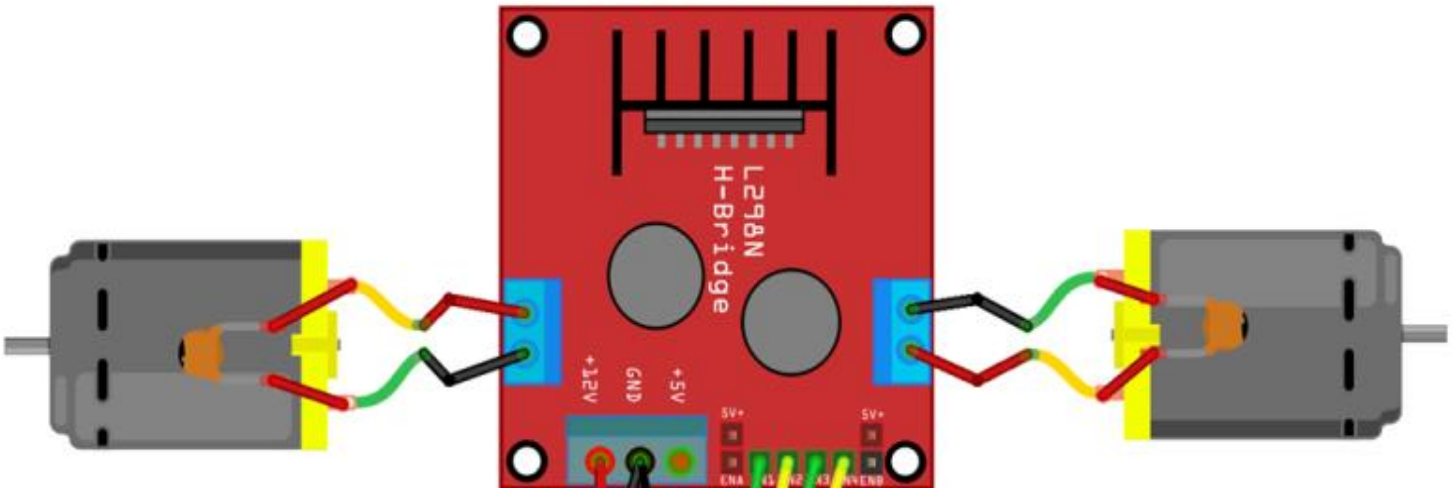
this time we are attaching a camera with the robot to make it a surveillance robot car. this web-controlled surveillance car can be easily built using the esp32-cam module. apart from the esp32-camera module, here we will use four dc motor with robot chassis and L298N motor driver module to build this robotic car. esp32 is one of the most popular boards to build IOT based projects esp32-cam doesn’t have a usb connector, so you need an Arduino uno to upload the code into esp32-cam. vcc and gnd pin of esp32 is connected with the vcc and gnd pin of the Arduino uno. TX of and RX of esp32 is connected with RX and TX of the Arduino uno. four dc motors are connected to esp32 through the L298N module. module pins are connected to io4, io2, io14, and io15 pins of esp32. before uploading the code, connect the io0 to the ground. io0 determines whether the esp32 is in flashing mode or not. when gpio 0 is connected to gnd, the esp32 is in flashing mode.

STEP 1: CONNECT GEAR MOTORS TO MOTOR DRIVER.

we first connect each two motors with each other's in each side, then we connect each connection of them to its place in the l298n driver, then we connect all the components to make our car and we are waiting for our esp to get ready to be connected.

STEP 2: CONNECT POWER SWITCH AND BATTERIES.

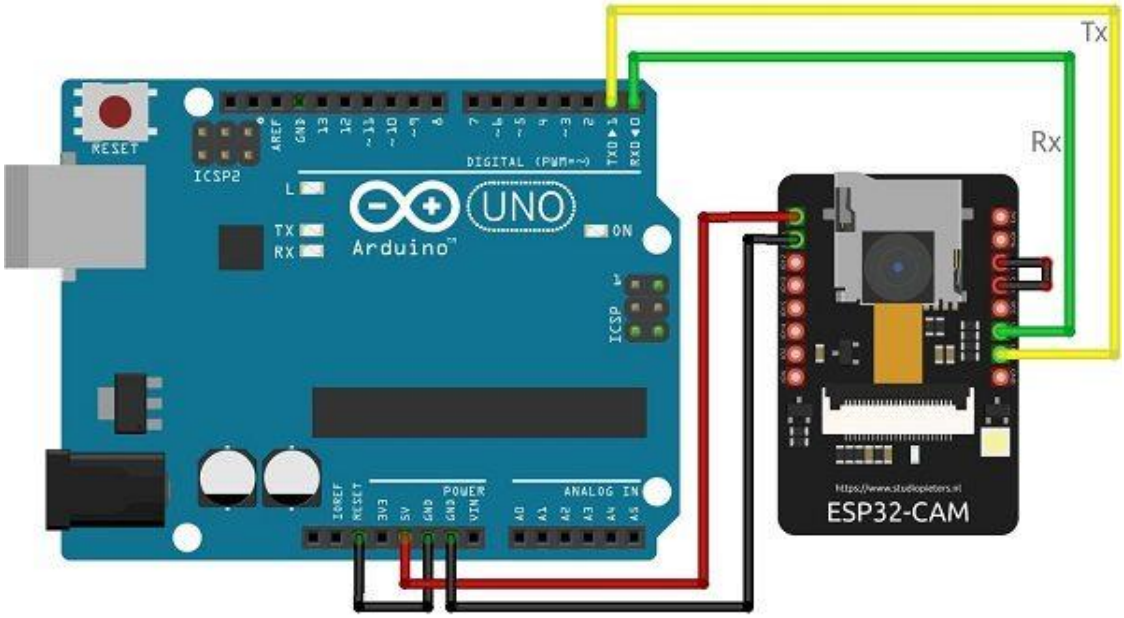
we connected the batteries to the driver as shown in the picture below and then we added the switch button



STEP 3: CONNECT ESP32 CAM TO THE ARDUINO UNO

we started by making the code that would be uploaded to the esp cam and we made sure that this code is working so we made some tests with the code to test the Wi-Fi, the web block in it, the camera, and finally the buttons that will control the car.
(You will find the code at the end of the report).

Here is the way we connect the Arduino to the esp cam.

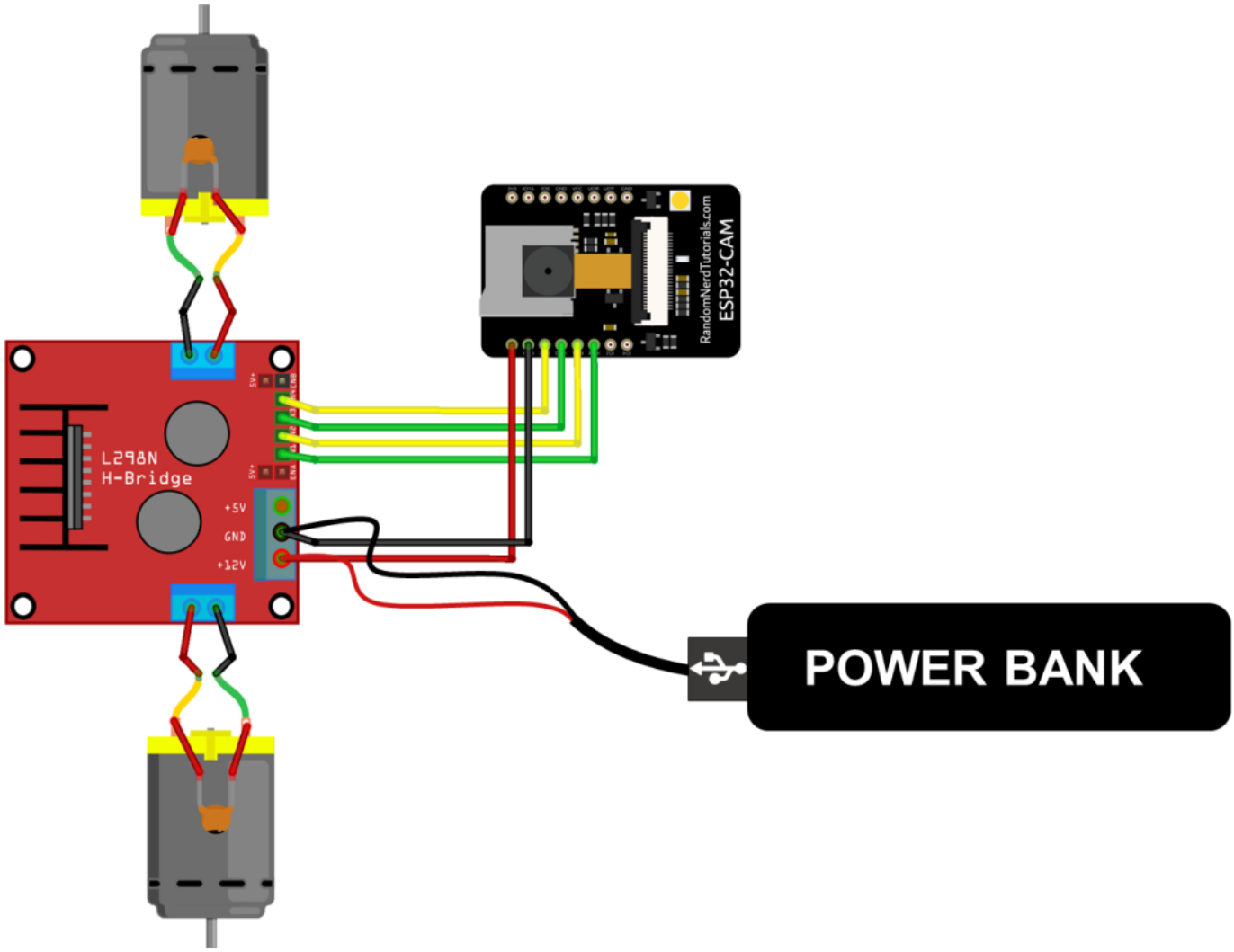


STEP 4: OPEN SERIAL MONITOR AND NOTE IP ADDRESS.

after we uploaded the code, we then need to disconnect the io0 and the ground then we save the code and push the esp button once and then we go to the serial monitor to get the Ip address of the esp Wi-Fi.

STEP 7: CONNECT ESP32 TO MOTOR DRIVER.

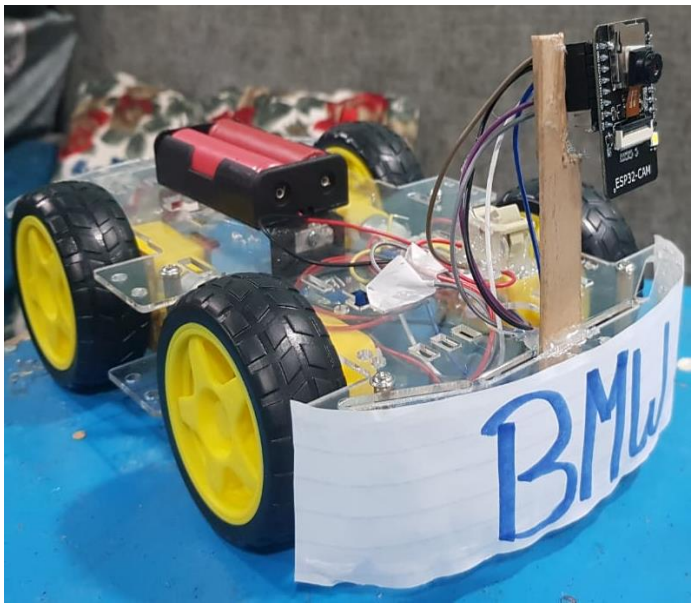
AFTER ALL OF THAT, WE THEN GET THE ESP TO CONNECT IT TO THE L298N DRIVER WITH THE CONNECTION IN THE PICTURE BELOW

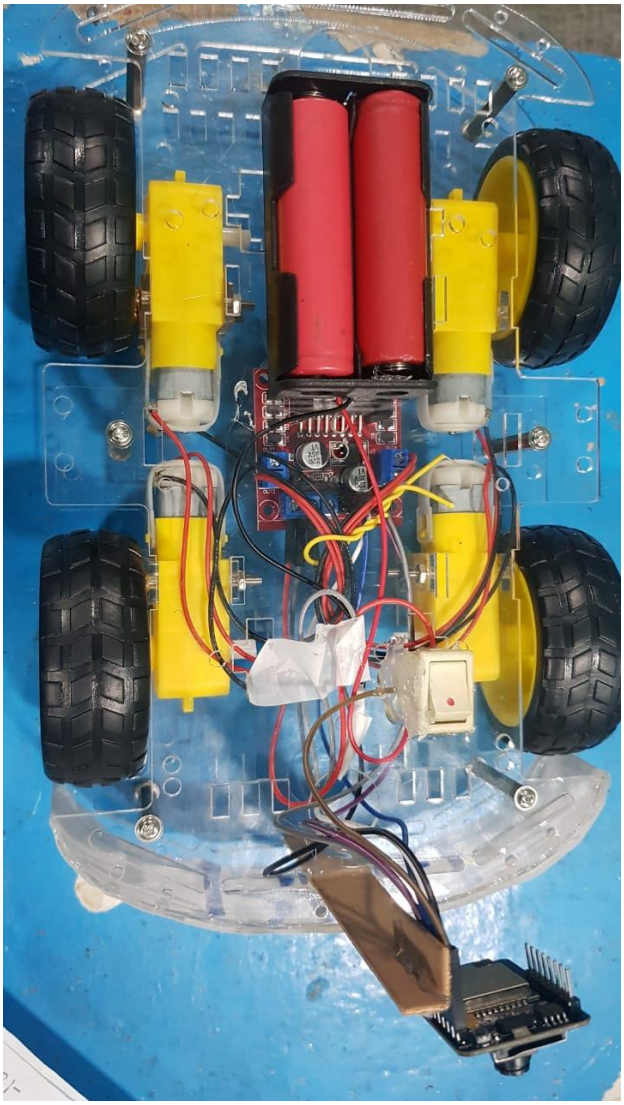


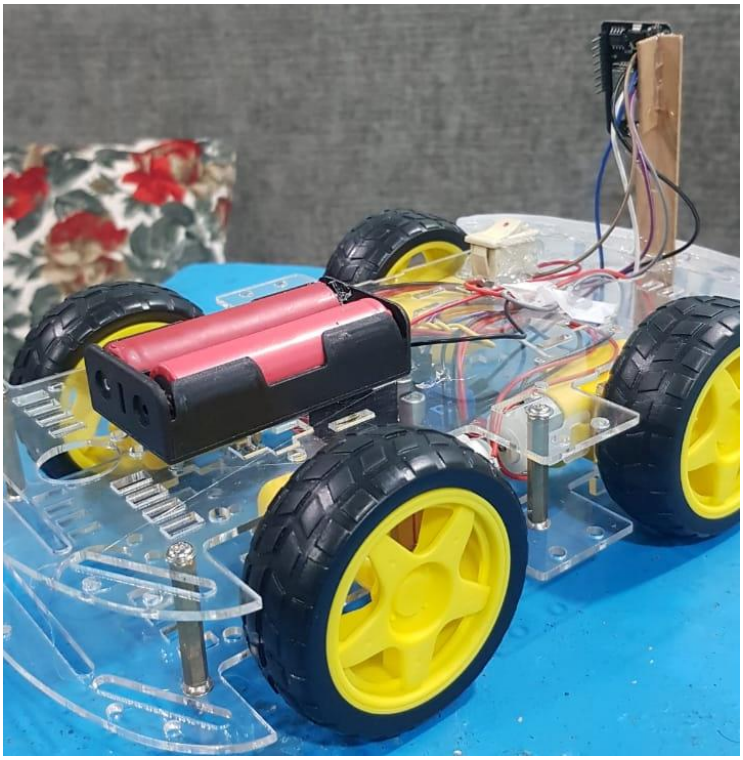
STEP 8: GRAPING ALL THE PIECES UP TOGETHER

After we finally finished the circuit with its connections , and we connected the RC car gear components together, now we need to put our circuit with the RC car and get it to work properly.

Our BMW







The Code we Used

```
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h"             // disable brownout problems
#include "soc/rtc_cntl_reg.h"    // disable brownout problems
#include "esp_http_server.h"

// Replace with your network credentials
const char* ssid = "ESP32";
const char* password = "123456789”;

#define PART_BOUNDARY "123456789000000000000987654321"

#define CAMERA_MODEL_AI_THINKER
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM_B
//#define CAMERA_MODEL_WROVER_KIT

#if defined(CAMERA_MODEL_WROVER_KIT)
  #define PWDN_GPIO_NUM    -1
  #define RESET_GPIO_NUM   -1
  #define XCLK_GPIO_NUM    21
  #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      19
  #define Y4_GPIO_NUM      18
  #define Y3_GPIO_NUM       5
  #define Y2_GPIO_NUM       4
  #define VSYNC_GPIO_NUM   25
  #define HREF_GPIO_NUM    23
  #define PCLK_GPIO_NUM    22

#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21
```

```
#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       17
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_AI_THINKER)
  #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

#elif defined(CAMERA_MODEL_M5STACK_PSRAM_B)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     22
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21

#else
  #error "Camera model not selected"
```

```
#endif

#define MOTOR_1_PIN_1    14
#define MOTOR_1_PIN_2    15
#define MOTOR_2_PIN_1    13
#define MOTOR_2_PIN_2    12

static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary="
PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length:
%u\r\n\r\n";

httpd_handle_t camera_httpd = NULL;
httpd_handle_t stream_httpd = NULL;

static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<html>
  <head>
    <title>ESP32-CAM Robot</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <style>
      body { font-family: Arial; text-align: center; margin:0px auto; padding-top:
30px;}
      table { margin-left: auto; margin-right: auto; }
      td { padding: 8 px; }
      .button {
        background-color: #2f4468;
        border: none;
        color: white;
        padding: 10px 20px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
        font-size: 18px;
        margin: 6px 3px;
        cursor: pointer;
        -webkit-touch-callout: none;
        -webkit-user-select: none;
        -khtml-user-select: none;
        -moz-user-select: none;
        -ms-user-select: none;
        user-select: none;
        -webkit-tap-highlight-color: rgba(0,0,0,0);
      }
      img {  width: auto ;
        max-width: 100% ;
        height: auto ;
      }
    </style>
  </head>
  <body>
    <h1>ESP32-CAM Robot</h1>
    <img src="" id="photo" >
    <table>
      <tr><td colspan="3" align="center"><button class="button"
onmousedown="toggleCheckbox('forward');" ontouchstart="toggleCheckbox('forward');"
onmouseup="toggleCheckbox('stop');"
ontouchend="toggleCheckbox('stop');">Forward</button></td></tr>
      <tr><td align="center"><button class="button"
onmousedown="toggleCheckbox('left');" ontouchstart="toggleCheckbox('left');"
onmouseup="toggleCheckbox('stop');"
ontouchend="toggleCheckbox('stop');">Left</button></td><td align="center"><button
class="button" onmousedown="toggleCheckbox('stop');"
ontouchstart="toggleCheckbox('stop');">Stop</button></td><td align="center"><button
class="button" onmousedown="toggleCheckbox('right');"
```

```
ontouchstart="toggleCheckbox('right');" onmouseup="toggleCheckbox('stop');"
ontouchend="toggleCheckbox('stop');">Right</button></td></tr>
      <tr><td colspan="3" align="center"><button class="button"
onmousedown="toggleCheckbox('backward');" ontouchstart="toggleCheckbox('backward');"
onmouseup="toggleCheckbox('stop');"
ontouchend="toggleCheckbox('stop');">Backward</button></td></tr>
    </table>
   <script>
   function toggleCheckbox(x) {
     var xhr = new XMLHttpRequest();
     xhr.open("GET", "/action?go=" + x, true);
     xhr.send();
   }
   window.onload = document.getElementById("photo").src = window.location.href.slice(0,
-1) + ":81/stream";
  </script>
  </body>
</html>
)rawliteral";

static esp_err_t index_handler(httpd_req_t *req){
  httpd_resp_set_type(req, "text/html");
  return httpd_resp_send(req, (const char *)INDEX_HTML, strlen(INDEX_HTML));
}

static esp_err_t stream_handler(httpd_req_t *req){
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t * _jpg_buf = NULL;
  char * part_buf[64];

  res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
  if(res != ESP_OK){
    return res;
  }

  while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      res = ESP_FAIL;
    } else {
      if(fb->width > 400){
        if(fb->format != PIXFORMAT_JPEG){
          bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
          esp_camera_fb_return(fb);
          fb = NULL;
          if(!jpeg_converted){
            Serial.println("JPEG compression failed");
            res = ESP_FAIL;
          }
        } else {
          _jpg_buf_len = fb->len;
          _jpg_buf = fb->buf;
        }
      }
    }
    if(res == ESP_OK){
      size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
      res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
```

```
if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY, strlen(_STREAM_BOUNDARY));
    }
    if(fb){
      esp_camera_fb_return(fb);
      fb = NULL;
      _jpg_buf = NULL;
    } else if(_jpg_buf){
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if(res != ESP_OK){
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}

static esp_err_t cmd_handler(httpd_req_t *req){
  char*  buf;
  size_t buf_len;
  char variable[32] = {0,};

  buf_len = httpd_req_get_url_query_len(req) + 1;
  if (buf_len > 1) {
    buf = (char*)malloc(buf_len);
    if(!buf){
      httpd_resp_send_500(req);
      return ESP_FAIL;
    }
    if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
      if (httpd_query_key_value(buf, "go", variable, sizeof(variable)) == ESP_OK) {
      } else {
        free(buf);
        httpd_resp_send_404(req);
        return ESP_FAIL;
      }
    } else {
      free(buf);
      httpd_resp_send_404(req);
      return ESP_FAIL;
    }
    free(buf);
  } else {
    httpd_resp_send_404(req);
    return ESP_FAIL;
  }

  sensor_t * s = esp_camera_sensor_get();
  int res = 0;

  if(!strcmp(variable, "forward")) {
    Serial.println("Forward");
    digitalWrite(MOTOR_1_PIN_1, 1);
    digitalWrite(MOTOR_1_PIN_2, 0);
    digitalWrite(MOTOR_2_PIN_1, 1);
    digitalWrite(MOTOR_2_PIN_2, 0);
  }
  else if(!strcmp(variable, "left")) {
    Serial.println("Left");
    digitalWrite(MOTOR_1_PIN_1, 0);
    digitalWrite(MOTOR_1_PIN_2, 1);
    digitalWrite(MOTOR_2_PIN_1, 1);
    digitalWrite(MOTOR_2_PIN_2, 0);
  }
```

```
  else if(!strcmp(variable, "right")) {
    Serial.println("Right");
    digitalWrite(MOTOR_1_PIN_1, 1);
    digitalWrite(MOTOR_1_PIN_2, 0);
    digitalWrite(MOTOR_2_PIN_1, 0);
    digitalWrite(MOTOR_2_PIN_2, 1);
  }
  else if(!strcmp(variable, "backward")) {
    Serial.println("Backward");
    digitalWrite(MOTOR_1_PIN_1, 0);
    digitalWrite(MOTOR_1_PIN_2, 1);
    digitalWrite(MOTOR_2_PIN_1, 0);
    digitalWrite(MOTOR_2_PIN_2, 1);
  }
  else if(!strcmp(variable, "stop")) {
    Serial.println("Stop");
    digitalWrite(MOTOR_1_PIN_1, 0);
    digitalWrite(MOTOR_1_PIN_2, 0);
    digitalWrite(MOTOR_2_PIN_1, 0);
    digitalWrite(MOTOR_2_PIN_2, 0);
  }
  else {
    res = -1;
  }

  if(res){
    return httpd_resp_send_500(req);
  }

  httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
  return httpd_resp_send(req, NULL, 0);
}

void startCameraServer(){
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;
  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = index_handler,
    .user_ctx  = NULL
  };

  httpd_uri_t cmd_uri = {
    .uri       = "/action",
    .method    = HTTP_GET,
    .handler   = cmd_handler,
    .user_ctx  = NULL
  };
  httpd_uri_t stream_uri = {
    .uri       = "/stream",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };
  if (httpd_start(&camera_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(camera_httpd, &index_uri);
    httpd_register_uri_handler(camera_httpd, &cmd_uri);
  }
  config.server_port += 1;
  config.ctrl_port += 1;
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &stream_uri);
  }
}
```

```
void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector

  pinMode(MOTOR_1_PIN_1, OUTPUT);
  pinMode(MOTOR_1_PIN_2, OUTPUT);
  pinMode(MOTOR_2_PIN_1, OUTPUT);
  pinMode(MOTOR_2_PIN_2, OUTPUT);

  Serial.begin(115200);
  Serial.setDebugOutput(false);

  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;

  if(psramFound()){
    config.frame_size = FRAMESIZE_VGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    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;
  }
  // Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  Serial.print("Camera Stream Ready! Go to: http://");
  Serial.println(WiFi.localIP());

  // Start streaming web server
  startCameraServer();
}

void loop() {}
```