

# Surveillence Car

Popa Maria-Eliza

## Scop:

Proiectul propus vizează crearea unei mașinuțe robot echipate cu o cameră, controlată și monitorizată prin intermediul unei aplicații mobile. Scopul acestui proiect este de a oferi o soluție interactivă și distractivă, dar și educativă, prin explorarea tehnologiilor moderne de comunicare între dispozitive. Proiectul este conceput pentru a oferi utilizatorului posibilitatea de a controla mașinuța în timp real și de a vizualiza mediul înconjurător prin intermediul camerei integrate, toate acestea fiind gestionate printr-o aplicație mobilă creată în MIT App Inventor.

Motivația alegerii acestui proiect derivă din dorința de a explora domeniul roboticilor și al comunicațiilor între dispozitive, oferind o experiență practică și hands-on în programare și tehnologie. Prin implementarea acestui proiect, se urmărește dezvoltarea cunoștințelor și abilităților în domeniul programării pentru microcontrolere, comunicării prin rețele Wi-Fi și dezvoltarea de aplicații mobile simple, accesibile oricărui utilizator.

Beneficiile acestui proiect sunt multiple:

**Educație Tehnologică:** Proiectul oferă o oportunitate de învățare în domeniul roboticilor, programării embedded și a comunicațiilor între dispozitive. Utilizatorul are posibilitatea de a înțelege și aplica concepte precum controlul motorului, capturarea și transmiterea imaginilor și dezvoltarea de aplicații mobile.

**Experiență Practică:** Proiectul oferă oportunitatea de a aplica cunoștințele teoretice într-un context real, construind și programând un sistem funcțional de la zero.

**Interactivitate și Distracție:** Mașinuța robot oferă o modalitate interactivă și distractivă de explorare a mediului înconjurător. **Utilizatorul poate explora locații inaccesibile sau periculoase prin intermediul mașinuței, controlând-o de la distanță.**

**Dezvoltarea Abilităților de Programare:** Implementarea acestui proiect implică programarea atât a microcontrolerului (ESP32), cât și a aplicației mobile MIT App Inventor, oferind ocazia de a dezvolta abilități în limbaje de programare specifice.

Preț  $\simeq$  150 RON (aprox. același preț cu alte dispozitive asemănătoare de pe AliExpress)

## Componente:



1x ESP32-CAM

1x L298N Modul cu drivere de motoare - pentru controlul motoarelor

1x Set de fire / Jumper wires

1x [Kit robot cu 4 motoare](#)

Inclus în kit:

4x Motor cu reductor 1:48 - pentru controlul roților

4x Roți

1x Comutator KCD1-11-2P - pentru pornire/oprire

1x Suport de Acumulatori 2x18650

2x Acumulatori 18650

Accesorii pentru montare

Optional:

Telefon + Conexiune la Internet/Bluetooth + 1x Tag NFC

Pistol de lipit fire + Soldering wire

Răbdare 😊

## Descriere:

### Cum funcționează:

ESP32-CAM creează un Access Point cu SSID-ul MyWifiCar și parola 12345678, și apoi transmite imaginea camerei la browser și preia comenzi din browser cu ajutorul următoarelor librării:

```
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
```

Camera face multiple poze pe secundă și le trimită asincron la server, care le afișează în browser.

Când utilizatorul apasă pe un buton, sau pe slider, browser-ul va face un request către server, iar serverul va apela funcția corespunzătoare acțiunii efectuate, pentru a pune mașina în funcțiune, sau pentru a aprinde led-ul de pe ESP32-CAM.

### Cum se utilizează:

Înțial se pornește comutatorul aflat în carcasa mașinii pentru a alimenta mașina.

Prin Browser:

Mai întâi, utilizatorul trebuie să conecteze dispozitivul mobil la rețea de internet "MyWifiCar", parola: "12345678", iar apoi, în browser-ul preferat va introduce în bara de căutare : 192.168.4.1, și va vedea în timp real ce vede camera și va putea controla mașina și direcția ei, dar și a led-ului de pe ESP32-CAM în caz de încăperea este slab iluminată.

Prin aplicație:

Utilizatorul va instala aplicația "DSFUM Car" disponibilă odată cu scanarea NFC-ului aflat pe carcasa mașinii, care este defapt APK-ul stocat în acest tag NFC. După ce va accepta toate permisiunile, va putea continua de la pasul următor.

Utilizatorul va trebui să conecteze dispozitivul mobil la ESP, prin conectarea dispozitivului mobil la rețea de internet MyWifiCar, parola: 12345678.

Apoi, utilizatorul poate apăsa pe "Continue" pentru a vedea în timp real ce vede camera, dar și pentru controlul mașinii, dar și a led-ului de pe ESP32-CAM în caz de încăperea este slab iluminată. Defapt, cum funcționează este că utilizatorul vede exact ce se află și în browser.

## Conexiuni:

Cum să conectezi componente:

Motoarele:

Partea Stângă:

Conecțăm plus la plus (cablu roșu) și minus la minus (cablu negru) între motoarele de pe partea stângă

Partea Dreaptă:

Conecțăm plus la plus (cablu roșu) și minus la minus (cablu negru) între motoarele de pe partea dreapta

Motoarele la L298N:

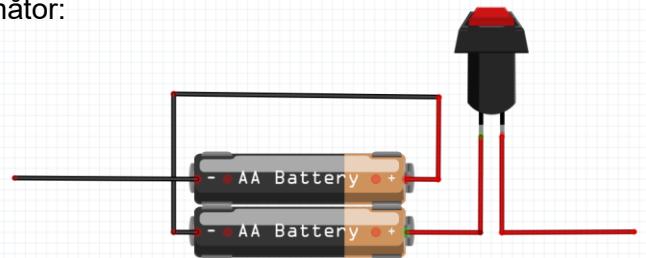
Partea Stângă:

Conecțăm plus la pinul OUT1 și minus la pinul OUT2 (sau invers, e irelevant) de pe driverul de motoare L298N

Partea Dreaptă:

Conecțăm plus la pinul OUT3 și minus la pinul OUT4 (sau invers, e irelevant) de pe driverul de motoare L298N

Dacă ați achiziționat kit-ul din secțiunea componente necesare, veți observa că suportul de baterii este făcut pentru conectarea în paralel. Însă noi avem nevoie de conectare în serie, deoarece motoarele noastre consumă între 3-6V, iar L298N mai produce o pierdere de curent din cauza felului în care a fost dezvoltat, deic avem nevoie de o sursă de curent minimă de 7V și maximă de 12V. Astfel, vom lipi firele în felul următor:



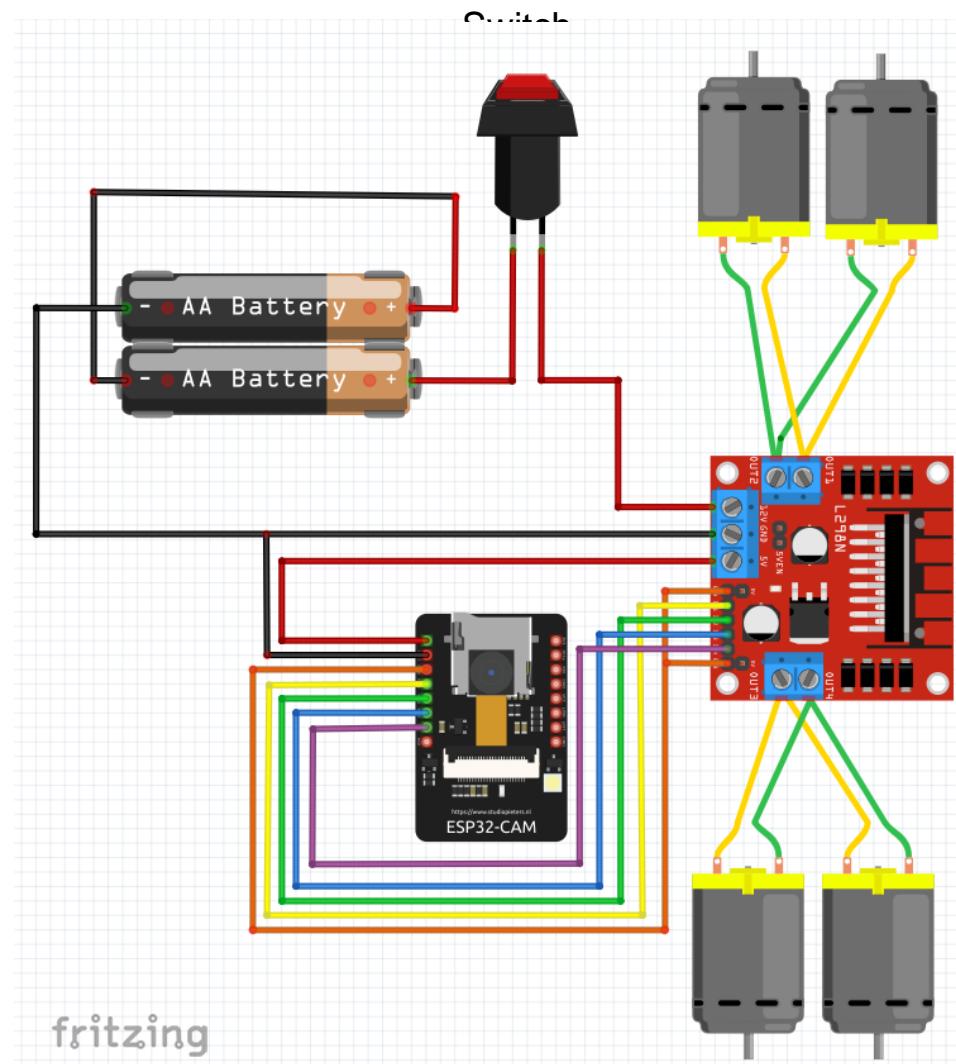
Conecțarea sursei de curent la driverul de motoare:

Se va conecta + (firul roșu) la pinul 12V al driverului, iar - (firul negru) la pinul GND

Conecțarea ESP32-CAM la L298N:

ESP32-CAM		L298N
Pinul 5V	->	Pinul 5V
Pinul GND	->	Pinul GND
Pinul IO13	->	Pinul IN1
Pinul IO15	->	Pinul IN2
Pinul IO14	->	Pinul IN3
Pinul IO2	->	Pinul IN4

Puteți vedea conexiunile aferente între componente și în următoare schemă:



## MIT App Inventor:

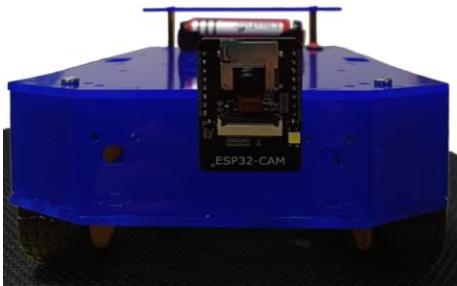
Aplicația MIT App Inventor nu este neapărat necesară, deoarece se poate controla foarte bine mașina și din browser, și, defapt, aceasta rulează imaginea din browser într-un WebViewer setat la adresa serverului creat de ESP32-CAM.

### Screen1

Aici avem o pagină de început, în care se specifică adresa Wi-Fi la care trebuie să te conectezi, și parola de acces, și cam atât. E doar o pagină informativă.

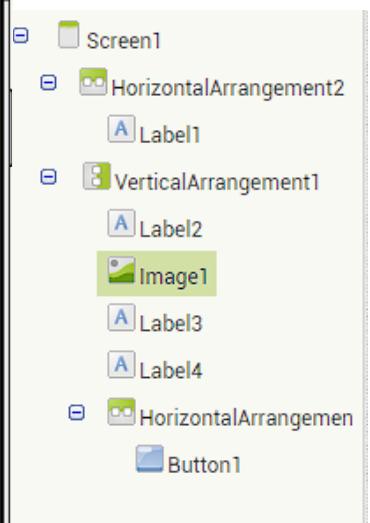
## DSFUM Car

Please start the car AND connect to the Car's WiFi in order to continue.



WiFi SSID: MyWiFiCar  
Password: 12345678

Continue



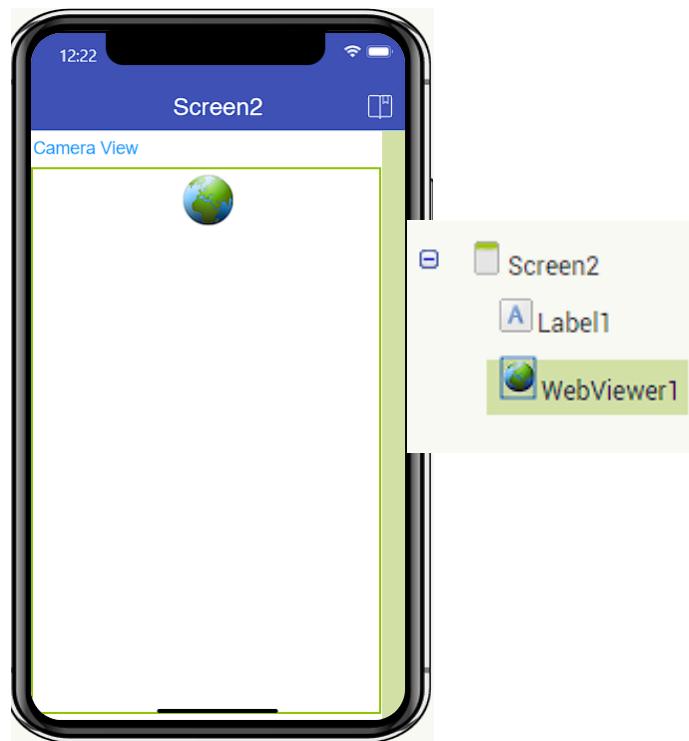
Blocks-uri utilizate pentru Screen1

```
when Button1 .Click
do open another screen screenName Screen2
    set Image1 .Picture to " jpg.car "
```

## Screen2

Aici, în schimb, avem doar un Label și un WebViewer care are setat în properties, la atributul HomeUrl, URL adresei create de ESP32-CAM, în cazul nostru: <http://192.168.4.1>

### Camera View



### Cod ESP:

```
#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 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    = "MyWiFiCar";
const char* password = "12345678";

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>
body {
    font-family: 'Arial', sans-serif; /* Replace 'Arial' with your preferred font */
    background-color: white;
    margin: 0;
    padding: 0;
}

#mainContainer {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

#mainTable {
    width: 90%;
    max-width: 400px;
    table-layout: fixed;
}

td.button {
    background-color: #116ee3ff;
    border-radius: .7em;
    height: 45px;
    font-size: 18px;
    color: white;
}

td.button:active {
    transform: translate(1px, 1px);
    box-shadow: none;
}

.slidecontainer {
    width: 100%;
}

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

```

.slider:hover {
  opacity: 1;
}

.slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 25px;
  height: 25px;
  border-radius: 50%;
  background: #116ee3ff;
  cursor: pointer;
}

.slider::-moz-range-thumb {
  width: 25px;
  height: 25px;
  border-radius: 50%;
  background: #116ee3ff;
  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:380px;margin:auto;table-layout:fixed" CELLSPACING=10>

  <tr>

    <img id="cameralImage" src="" style="width:380px;height:250px"></td>

  </tr>

  <tr>

    <td></td>

    <td class="button" ontouchstart='sendButtonInput("MoveCar", "1")' ontouchend='sendButtonInput("MoveCar", "0")'>UP</td>

    <td></td>

  </tr>

  <tr>

    <td class="button" ontouchstart='sendButtonInput("MoveCar", "3")' ontouchend='sendButtonInput("MoveCar", "0")'>LEFT</td>

    <td class="button"></td>

    <td class="button" ontouchstart='sendButtonInput("MoveCar", "4")' ontouchend='sendButtonInput("MoveCar", "0")'>RIGHT</td>

  </tr>

  <tr>

```

<td></td>
<td class="button" ontouchstart='sendButtonInput("MoveCar", "2")'
ontouchend='sendButtonInput("MoveCar", "0")'>DOWN</td>
<td></td>
</tr>
<tr><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 imgId = document.getElementById("cameralImage");
        imgId.src = URL.createObjectURL(event.data);
    };
}

function initCarInputWebSocket()
{
    websocketCarInput = new WebSocket(webSocketCarInputUrl);
    websocketCarInput.onopen = function(event)
    {
        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, BACKWARD);
    break;

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

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

case RIGHT:
    rotateMotor(RIGHT_MOTOR, BACKWARD);
    rotateMotor(LEFT_MOTOR, BACKWARD);
    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 == "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(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);
    }
    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");

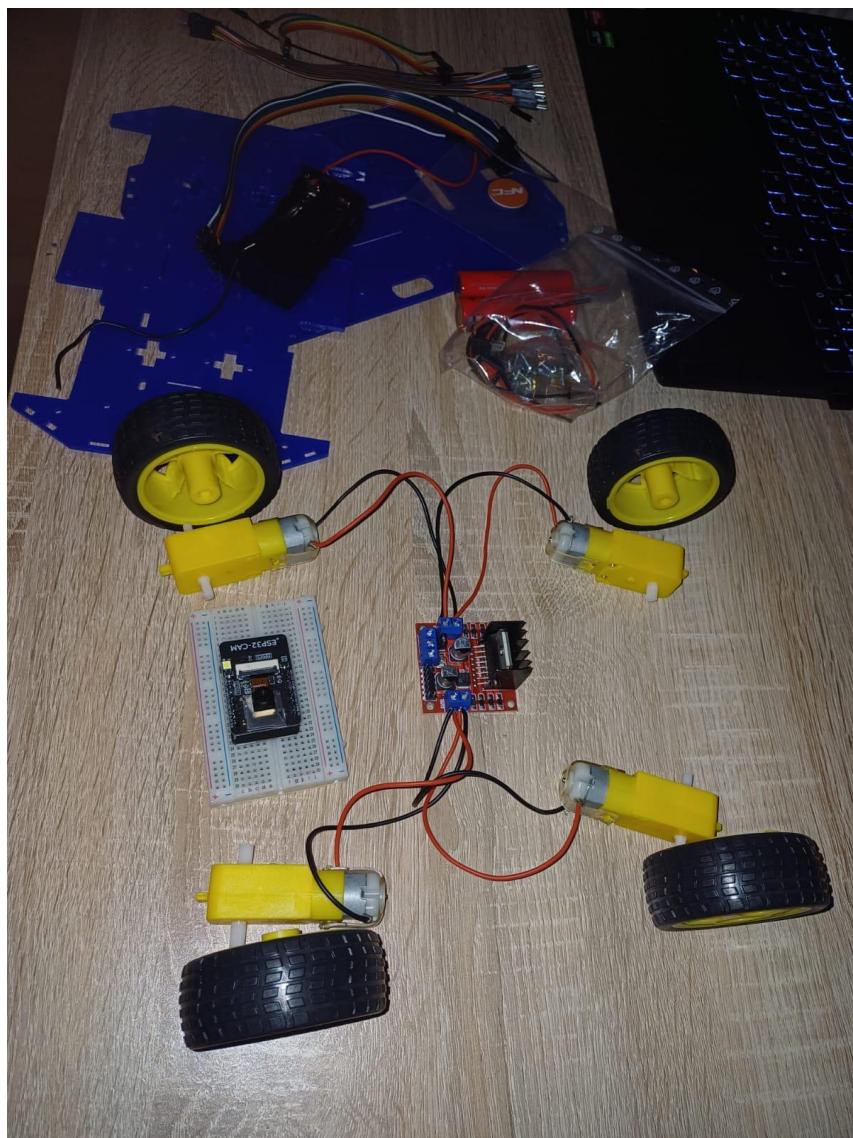
    setupCamera();
}

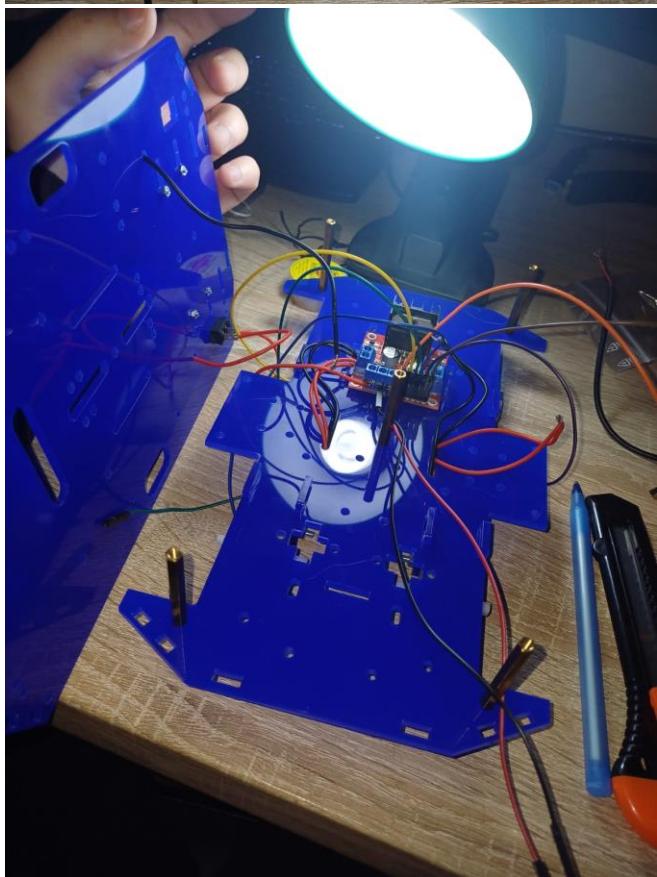
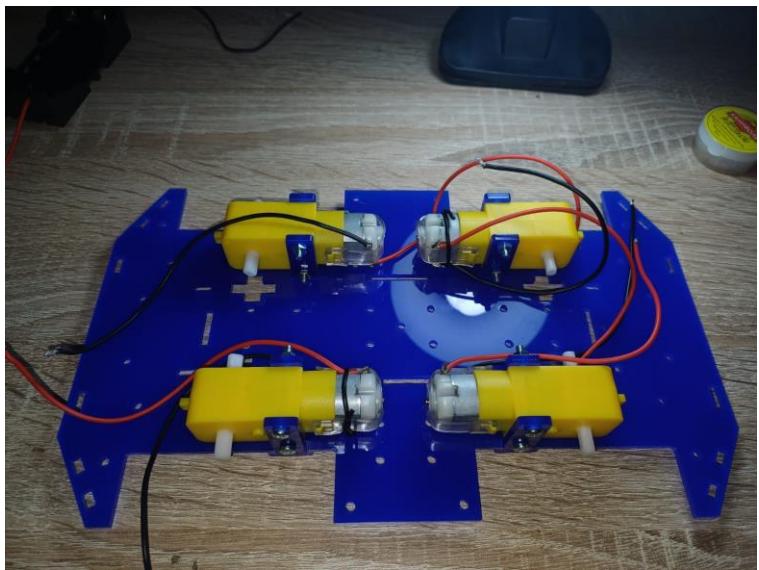
```

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

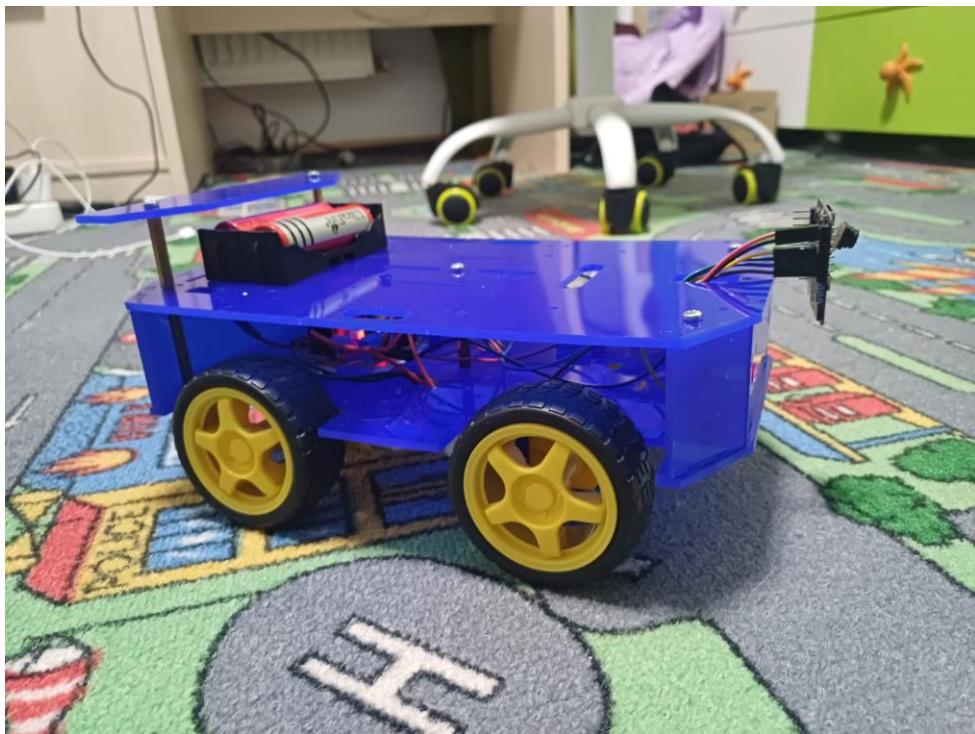
Sursa cod: <https://github.com/un0038998/CameraCar> (contribuția mea constă în anumite modificări ce țin de bug-uri și prelucrarea design-ului final)

Poze din timpul realizării:





**And the FINAL product!!**



## Imbunătățiri:

Cu siguranță, am putea îmbunătăți proiectul folosind un sistem de Pan-Tilt pentru cameră împreună cu două servo-motoare, astfel încât să fie permisă vizualizarea din mai multe unghiuri a locației în care se află mașina. Totuși costurile proiectului ar fi crescut, și nu mi-a permis mai mult bugetul :D

## Bibliografie:

(sau surse de inspirație)

<https://www.youtube.com/watch?v=HfQ7lhhqDOk>

<https://srituhobby.com/how-to-make-a-wifi-controlled-car-using-nodemcu-and-blynk-app-step-by-step-instructions/>

<https://randomnerdtutorials.com/esp32-cam-car-robot-web-server/>