Coche con ESP32-Cam

Proyectos de dos vehiculos utilizando la placa ESP32-Cam:

1º proyecto, utilizamos la ESP32-Cam para hacer fotos y guardar en Google Drive, utilizando una página Web, el vehiculo lo controlamos con una placa de arduino y una placa Motor Shield.

El vehiculo funciona de forma automática evitando obtaculos con ayuda de un sensor de ultrasonidos.

Materiales:

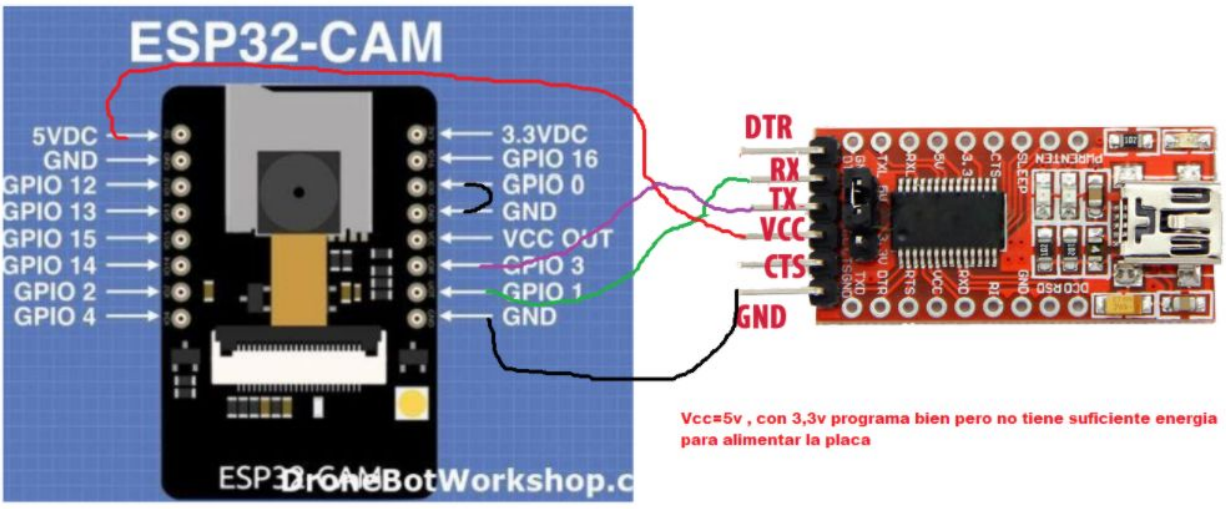
- Arduino uno.
- Arduino Motor Shield.
- ESP32-CAM.
- Servomotor.
- Interruptor de encendido
- Pulsador de paro
- Bateria de 7,3v.
- Sensor ultrasonico.
- Plataforma movil con 2 motores de CC.

Funcionamiento:

Al pulsar en Interruptor de encendido, el movil se pone en marcha , al detectar el sensor de ultrasonidos, un obtaculo a (una distancia que podemos fijar en el scrip) el servo mueve el sensor a izquierda y derecha para ver donde hay un recorrido mas libre y gira las ruedas para esquivar el obtaculo.

Al mismo tiempo podemos ver en la pagina web creada por la ESP32-Cam la imagen en vivo o hacer una foto que se guarda en Google Driver.

Programar ESP32-Cam;



Scrips:

ESP32-Cam:
Utiliza archivos:

- ESP32-CAM_Diver.ino : archivo principal
- Base64.cpp : Para guardar foto en Google Driver
- Base64.h : Para guardar foto en Google Driver
- app_httpd.cpp : Crea página web y controles (solo utilizamos los de la camara, los otros controles se pueden utilizar para controlar el movil, lo veremos en el siguiente proyecto)

ESP32-CAM_Diver.ino

```
/*
ESP32-CAM_Diver.ino
Visualiza imagen  mediante página web
Guardar foto en Driver
*/

#include "esp_wifi.h"
#include "esp_camera.h"

#include <WiFi.h>
#include "soc/soc.h"  //Desactivar problemas de brownout apagado
#include "soc/rtc_cntl_reg.h" //Desactivar problemas de brownour

#define WIFI_SSID "....."              //Completar nombre de red
#define WIFI_PASSWORD "....." //Completar contraseña de red
```

```
//---> Define parametros camara
#define 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
void startCameraServer();

//....> Pines ESP32 que se pueden  utilizar para el control del coche NO UTILIZADO
const int MotPin0 = 12;
const int MotPin1 = 13;
const int MotPin2 = 14;
const int MotPin3 = 15;

//Inicializa motores NO UTILIZADO
void initMotors() {
// Información generar PWM con ESP32
  //https://randomnerdtutorials.com/esp32-pwm-arduino-ide/

//  ledcSetup(4, 2000, 8); // Configura canal para PWM(canal 3,frecuencia 2000hz PWM, 8-bit resolution)
//  ledcSetup(3, 2000, 8);
//  ledcSetup(5, 2000, 8);
//  ledcSetup(6, 2000, 8);
//  ledcAttachPin(MotPin0, 3);
//  ledcAttachPin(MotPin1, 4); // Asigna canal PWM a pin GPIO
//  ledcAttachPin(MotPin2, 5);
//  ledcAttachPin(MotPin3, 6);
}

//--->Inicilaliza servo que controla la camara NO UTILIZADO
const int ServoPin = 2;

void initServo()  { //NO UTILIZADO
  ledcSetup(8, 50, 16);  //canal 8 50 hz PWM, 16-bit resolution, range from 3250 to 6500.
  ledcAttachPin(ServoPin, 8); // Asigna canal PWM a pin servo
}

void setup()
{
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); // prevent brownouts by silencing them

  Serial.begin(115200);
  Serial.setDebugOutput(true);//Para activar la salida de debug
  Serial.println();

  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;

  //Inicializa con especificaciones altas para preasignar búferes más grandes
  if(psramFound()){
    config.frame_size = FRAMESIZE_QVGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_QVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Fallo inicio de Camara con error 0x%x", err);
    return;
  }

 // tamaño de fotograma desplegable para una mayor velocidad inicial
  sensor_t * s = esp_camera_sensor_get();
  s->set_framesize(s, FRAMESIZE_QVGA); // VGA|CIF|QVGA|HQVGA|QQVGA   ( UXGA? SXGA? XGA? SVGA? )
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);

  // ---> Control del coche NO UTILIZADO
 // initMotors();
 // initServo();

// Flash inicial de la camara
  ledcSetup(7, 5000, 8);
  ledcAttachPin(4, 7);  //pin4 is LED


  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  delay(500);

  long int StartTime=millis();
  while (WiFi.status() != WL_CONNECTED)
  {
      delay(500);
      if ((StartTime+10000) < millis()) break;
  }


  startCameraServer();

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.print("Camara lista! en 'http://");
    Serial.print(WiFi.localIP());
    Serial.println("' to connect");
  }
  else { // Si no se conecta crea Wifi
    Serial.println("");
```

```
  Serial.println("WiFi disconnected");
  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.softAPIP());
  Serial.println("' to connect");
  char* apssid = "ESP32-CAM";
  char* appassword = "12345678";         //AP password requiere al menos 8 characteres.
  WiFi.softAP((WiFi.softAPIP().toString()+"_"+(String)apssid).c_str(), appassword);
 }



 //---> Muestra un flag inicial
 for (int i=0;i<5;i++)
 {
  ledcWrite(7,10);  // flash led
  delay(50);
  ledcWrite(7,0);
  delay(50);
 }
}

void loop() {
  delay(1000);
Serial.printf("RSSi: %ld dBm\n",WiFi.RSSI()); //Muestra intensidad señal Wifi
}
```

Base64.cpp

```
*
 * Copyright (c) 2013 Adam Rudd.
 * See LICENSE for more information
 * https://github.com/adamvr/arduino-base64
 */
#if (defined(__AVR__))
#include <avr\pgmspace.h>
#else
#include <pgmspace.h>
#endif

const char PROGMEM b64_alphabet[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                                    "abcdefghijklmnopqrstuvwxyz"
                                    "0123456789+/";

/* 'Private' declarations */
inline void a3_to_a4(unsigned char * a4, unsigned char * a3);
inline void a4_to_a3(unsigned char * a3, unsigned char * a4);
inline unsigned char b64_lookup(char c);

int base64_encode(char *output, char *input, int inputLen) {
        int i = 0, j = 0;
        int encLen = 0;
        unsigned char a3[3];
        unsigned char a4[4];

        while(inputLen--) {
                a3[i++] = *(input++);
                if(i == 3) {
                        a3_to_a4(a4, a3);

                        for(i = 0; i < 4; i++) {
                                output[encLen++] = pgm_read_byte(&b64_alphabet[a4[i]]);
                        }
```

```
                i = 0;
            }
        }

        if(i) {
            for(j = i; j < 3; j++) {
                a3[j] = '\0';
            }

            a3_to_a4(a4, a3);

            for(j = 0; j < i + 1; j++) {
                output[encLen++] = pgm_read_byte(&b64_alphabet[a4[j]]);
            }

            while((i++ < 3)) {
                output[encLen++] = '=';
            }
        }
        output[encLen] = '\0';
        return encLen;
}

int base64_decode(char * output, char * input, int inputLen) {
        int i = 0, j = 0;
        int decLen = 0;
        unsigned char a3[3];
        unsigned char a4[4];


        while (inputLen--) {
            if(*input == '=') {
                break;
            }

            a4[i++] = *(input++);
            if (i == 4) {
                for (i = 0; i <4; i++) {
                    a4[i] = b64_lookup(a4[i]);
                }

                a4_to_a3(a3,a4);

                for (i = 0; i < 3; i++) {
                    output[decLen++] = a3[i];
                }
                i = 0;
            }
        }

        if (i) {
            for (j = i; j < 4; j++) {
                a4[j] = '\0';
            }

            for (j = 0; j <4; j++) {
                a4[j] = b64_lookup(a4[j]);
            }

            a4_to_a3(a3,a4);

            for (j = 0; j < i - 1; j++) {
                output[decLen++] = a3[j];
            }
        }
        output[decLen] = '\0';
        return decLen;
```

```
}

int base64_enc_len(int plainLen) {
	int n = plainLen;
	return (n + 2 - ((n + 2) % 3)) / 3 * 4;
}

int base64_dec_len(char * input, int inputLen) {
	int i = 0;
	int numEq = 0;
	for(i = inputLen - 1; input[i] == '='; i--) {
		numEq++;
	}

	return ((6 * inputLen) / 8) - numEq;
}

inline void a3_to_a4(unsigned char * a4, unsigned char * a3) {
	a4[0] = (a3[0] & 0xfc) >> 2;
	a4[1] = ((a3[0] & 0x03) << 4) + ((a3[1] & 0xf0) >> 4);
	a4[2] = ((a3[1] & 0x0f) << 2) + ((a3[2] & 0xc0) >> 6);
	a4[3] = (a3[2] & 0x3f);
}

inline void a4_to_a3(unsigned char * a3, unsigned char * a4) {
	a3[0] = (a4[0] << 2) + ((a4[1] & 0x30) >> 4);
	a3[1] = ((a4[1] & 0xf) << 4) + ((a4[2] & 0x3c) >> 2);
	a3[2] = ((a4[2] & 0x3) << 6) + a4[3];
}

inline unsigned char b64_lookup(char c) {
	if(c >='A' && c <='Z') return c - 'A';
	if(c >='a' && c <='z') return c - 71;
	if(c >='0' && c <='9') return c + 4;
	if(c == '+') return 62;
	if(c == '/') return 63;
	return -1;
}
```

Base64.h

```
/*
 * Copyright (c) 2013 Adam Rudd.
 * See LICENSE for more information
 * https://github.com/adamvr/arduino-base64
 */
#ifndef _BASE64_H
#define _BASE64_H

/* b64_alphabet:
 *		Description: Base64 alphabet table, a mapping between integers
 *			and base64 digits
 *		Notes: This is an extern here but is defined in Base64.c
 */
extern const char b64_alphabet[];

/* base64_encode:
 *		Description:
 *			Encode a string of characters as base64
 *		Parameters:
 *			output: the output buffer for the encoding, stores the encoded string
 *			input: the input buffer for the encoding, stores the binary to be encoded
 *			inputLen: the length of the input buffer, in bytes
```

```
 *                      Return value:
 *                              Returns the length of the encoded string
 *                      Requirements:
 *                              1. output must not be null or empty
 *                              2. input must not be null
 *                              3. inputLen must be greater than or equal to 0
 */
int base64_encode(char *output, char *input, int inputLen);

/* base64_decode:
 *                      Description:
 *                              Decode a base64 encoded string into bytes
 *                      Parameters:
 *                              output: the output buffer for the decoding,
 *                                                      stores the decoded binary
 *                              input: the input buffer for the decoding,
 *                                              stores the base64 string to be decoded
 *                              inputLen: the length of the input buffer, in bytes
 *                      Return value:
 *                              Returns the length of the decoded string
 *                      Requirements:
 *                              1. output must not be null or empty
 *                              2. input must not be null
 *                              3. inputLen must be greater than or equal to 0
 */
int base64_decode(char *output, char *input, int inputLen);

/* base64_enc_len:
 *                      Description:
 *                              Returns the length of a base64 encoded string whose decoded
 *                              form is inputLen bytes long
 *                      Parameters:
 *                              inputLen: the length of the decoded string
 *                      Return value:
 *                              The length of a base64 encoded string whose decoded form
 *                              is inputLen bytes long
 *                      Requirements:
 *                              None
 */
int base64_enc_len(int inputLen);

/* base64_dec_len:
 *                      Description:
 *                              Returns the length of the decoded form of a
 *                              base64 encoded string
 *                      Parameters:
 *                              input: the base64 encoded string to be measured
 *                              inputLen: the length of the base64 encoded string
 *                      Return value:
 *                              Returns the length of the decoded form of a
 *                              base64 encoded string
 *                      Requirements:
 *                              1. input must not be null
 *                              2. input must be greater than or equal to zero
 */
int base64_dec_len(char *input, int inputLen);

#endif // _BASE64_H
```

app_httpd.cpp (en rojo los botones no utilizados para control del coche)

```
#include "dl_lib_matrix3d.h" //Para convertir fotos en redes neuronales (reconocimiento facial
#include <esp32-hal-ledc.h>  //¿configura canales PWM
int speed = 255;
int noStop = 0;


#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "Arduino.h"
#include <WiFiClientSecure.h>
#include "Base64.h"

//---> Para Guardar en Google Driver
String myScript = "/macros/s/AKfycbxtKpxBMKo...../exec";  // sustituir
const char* myDomain = "script.google.com";
String myFilename = "filename=ESP32-CAM.jpg";
String mimeType = "&mimetype=image/jpeg";
String myImage = "&data=";

int waitingTime = 30000; //Wait 30 seconds to google response.


//--->Definición de funciones
String Photo2Base64() ;
String urlencode(String str);
void saveCapturedImage();

//define estructura de datos para la foto
typedef struct {
        httpd_req_t *req;
        size_t len;
} jpg_chunking_t;

#define PART_BOUNDARY "123456789000000000000987654321"
static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
//BOUNDARY=PERÍMETRO
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 stream_httpd = NULL;
httpd_handle_t camera_httpd = NULL;
//.....................................................
static size_t jpg_encode_stream(void * arg, size_t index, const void* data, size_t len){
    jpg_chunking_t *j = (jpg_chunking_t *)arg;
    if(!index){
        j->len = 0;
    }
    if(httpd_resp_send_chunk(j->req, (const char *)data, len) != ESP_OK){
        return 0;
    }
    j->len += len;
    return len;
}
//..... HACE FOTO ..................................................
static esp_err_t capture_handler(httpd_req_t *req){

    camera_fb_t * fb = NULL;
    esp_err_t res = ESP_OK;
    int64_t fr_start = esp_timer_get_time();
```

```
    fb = esp_camera_fb_get();
    if (!fb) {
        Serial.println("Camara captura fallida");
        httpd_resp_send_500(req);
        return ESP_FAIL;
    }

    httpd_resp_set_type(req, "image/jpeg");
    httpd_resp_set_hdr(req, "Content-Disposition", "inline; filename=capture.jpg");

    size_t out_len, out_width, out_height;
    uint8_t * out_buf;
    bool s;
    {
        size_t fb_len = 0;
        if(fb->format == PIXFORMAT_JPEG){
            fb_len = fb->len;
            res = httpd_resp_send(req, (const char *)fb->buf, fb->len);
        } else {
            jpg_chunking_t jchunk = {req, 0};
            res = frame2jpg_cb(fb, 80, jpg_encode_stream, &jchunk)?ESP_OK:ESP_FAIL;
            httpd_resp_send_chunk(req, NULL, 0);
            fb_len = jchunk.len;
        }
        esp_camera_fb_return(fb);
        int64_t fr_end = esp_timer_get_time();
        Serial.printf("JPG: %uB %ums\n", (uint32_t)(fb_len), (uint32_t)((fr_end - fr_start)/1000));
       Serial.println("Guarda foto en Driver");
       saveCapturedImage();  //Guarda foto en driver
       return res;

    }

    dl_matrix3du_t *image_matrix = dl_matrix3du_alloc(1, fb->width, fb->height, 3);
    if (!image_matrix) {
        esp_camera_fb_return(fb);
        Serial.println("dl_matrix3du_alloc failed");
        httpd_resp_send_500(req);
        return ESP_FAIL;
    }

    out_buf = image_matrix->item;
    out_len = fb->width * fb->height * 3;
    out_width = fb->width;
    out_height = fb->height;

    s = fmt2rgb888(fb->buf, fb->len, fb->format, out_buf);
    esp_camera_fb_return(fb);
    if(!s){
        dl_matrix3du_free(image_matrix);
        Serial.println("to rgb888 failed");
        httpd_resp_send_500(req);
        return ESP_FAIL;
    }

    jpg_chunking_t jchunk = {req, 0};
    s = fmt2jpg_cb(out_buf, out_len, out_width, out_height, PIXFORMAT_RGB888, 90, jpg_encode_stream, &jchunk);
    dl_matrix3du_free(image_matrix);
    if(!s){
        Serial.println("JPEG compression failed");
        return ESP_FAIL;
    }

    int64_t fr_end = esp_timer_get_time();

    return res;
}
//..... VIDEO  .....................................
```

```
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];
    dl_matrix3du_t *image_matrix = NULL;

    static int64_t last_frame = 0;
    if(!last_frame) {
        last_frame = esp_timer_get_time();
    }

    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->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;
        }
        int64_t fr_end = esp_timer_get_time();
        int64_t frame_time = fr_end - last_frame;
        last_frame = fr_end;
        frame_time /= 1000;
        Serial.printf("MJPG: %uB %ums (%.1ffps)\n",
            (uint32_t)(_jpg_buf_len),
            (uint32_t)frame_time, 1000.0 / (uint32_t)frame_time
```

```
        );
    }

    last_frame = 0;
    return res;
}
//..... CONTROLES (BOTONES Y DESLIZADORES).....................................
enum state {fwd,rev,stp,drv};
state actstate = stp;

static esp_err_t cmd_handler(httpd_req_t *req)
{
    char*  buf;
    size_t buf_len;
    char variable[32] = {0,};
    char value[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, "var", variable, sizeof(variable)) == ESP_OK &&
                httpd_query_key_value(buf, "val", value, sizeof(value)) == 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;
    }

    int val = atoi(value);
    sensor_t * s = esp_camera_sensor_get();
    int res = 0;

    if(!strcmp(variable, "framesize"))    {
        Serial.println("framesize");
        if(s->pixformat == PIXFORMAT_JPEG) res = s->set_framesize(s, (framesize_t)val);
    }
    else if(!strcmp(variable, "quality"))     {
      Serial.println("quality");
      res = s->set_quality(s, val);
    }
    //****************** Controles *********************************//
    //No use el canal 1 y el canal 2
    else if(!strcmp(variable, "flash"))     {
      ledcWrite(7,val); // Led flash
    }
    else if(!strcmp(variable, "speed"))     {
      if      (val > 255) val = 255;
      else if (val <   0) val = 0;
      speed = val;
      ledcWrite(4,speed);  // pin IO13  velocidad
    }
    else if(!strcmp(variable, "nostop"))     {
      noStop = val;
```

```
    ledcWrite(3, 0);
  }
  else if(!strcmp(variable, "servo")){ // Valores para servo en automatico
    if      (val > 180) val = 180;
    else if (val < 0) val = 0;
    ledcWrite(8,val); //servo
    depurar("servo " + String(val));
  }
  else if(!strcmp(variable, "car")) {

    if (val==1) {
      Serial.println("Adelante");
      actstate = fwd;
       digitalWrite(14,HIGH);
       digitalWrite(15,HIGH);
      delay(200);
    }
    else if (val==2) {
      Serial.println("Giro izquierdas");
       digitalWrite(14,LOW);
       digitalWrite(15,HIGH);
       delay(200);
    }
    else if (val==3) {
      Serial.println("Parar");
      actstate = stp;
        ledcWrite(4,0); // IO13 velocidad 0
    }
    else if (val==4) {
      Serial.println("Giro derecha");
       digitalWrite(14,HIGH);
       digitalWrite(15,LOW);
      delay(200);

    }
    else if (val==5) {
      Serial.println("Atrás");
      actstate = rev;
       digitalWrite(14,LOW);
       digitalWrite(15,LOW);
       delay(200);
    }
    else if (val==6) {
      Serial.println("Guarda foto en driver");
      actstate = drv;
      saveCapturedImage();
      delay(200);
     }


    if (noStop!=1) {
      // Manual
       digitalWrite(12,HIGH); //Manual
    }
  }    //if car
  else   {
    Serial.println("variable");
    res = -1;
  }   //else car

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

  httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
  return httpd_resp_send(req, NULL, 0);
}
//.....................................................
static esp_err_t status_handler(httpd_req_t *req){
  static char json_response[1024];
```

```
    Serial.println("-----------------------------------");
   Serial.println("status_handler");
    sensor_t * s = esp_camera_sensor_get();
    char * p = json_response;
    *p++ = '{';

    p+=sprintf(p, "\"framesize\":%u,", s->status.framesize);
    p+=sprintf(p, "\"quality\":%u,", s->status.quality);
    *p++ = '}';
    *p++ = 0;
    httpd_resp_set_type(req, "application/json");
    httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
    return httpd_resp_send(req, json_response, strlen(json_response));
}
//.....................................................
// Pagina Web
static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<!doctype html>
<html>
   <head>
      <meta charset="utf-8">
      <meta name="viewport" content="width=device-width,initial-scale=1">
      <title>Coche con ESP32 OV2460 </title>
      <style>

body{font-family:Arial,Helvetica,sans-serif;background:#181818;color:#EFEFEF;font-size:16px}h2{font-size:18px}section.
main{display:flex}#menu,section.main{flex-direction:column}#menu{display:none;flex-wrap:nowrap;min-width:340px;back
ground:#363636;padding:8px;border-radius:4px;margin-top:-10px;margin-right:10px}#content{display:flex;flex-wrap:wrap;
align-items:stretch}figure{padding:0;margin:0;-webkit-margin-before:0;margin-block-start:0;-webkit-margin-after:0;margin-
block-end:0;-webkit-margin-start:0;margin-inline-start:0;-webkit-margin-end:0;margin-inline-end:0}figure
img{display:block;width:100%;height:auto;border-radius:4px;margin-top:8px}@media (min-width: 800px) and
(orientation:landscape){#content{display:flex;flex-wrap:nowrap;align-items:stretch}figure
img{display:block;max-width:100%;max-height:calc(100vh -
40px);width:auto;height:auto}figure{padding:0;margin:0;-webkit-margin-before:0;margin-block-start:0;-webkit-margin-after:
0;margin-block-end:0;-webkit-margin-start:0;margin-inline-start:0;-webkit-margin-end:0;margin-inline-end:0}}section#butto
ns{display:flex;flex-wrap:nowrap;justify-content:space-between}#nav-toggle{cursor:pointer;display:block}#nav-toggle-cb{o
utline:0;opacity:0;width:0;height:0}#nav-toggle-cb:checked+#menu{display:flex}.input-group{display:flex;flex-wrap:nowrap
;line-height:22px;margin:5px 0}.input-group>label{display:inline-block;padding-right:10px;min-width:47%}.input-group
input,.input-group select{flex-grow:1}.range-max,.range-min{display:inline-block;padding:0
5px}button{display:block;margin:5px;padding:0
12px;border:0;line-height:28px;cursor:pointer;color:#fff;background:#ff3034;border-radius:5px;font-size:16px;outline:0}but
ton:hover{background:#ff494d}button:active{background:#f21c21}button.disabled{cursor:default;background:#a0a0a0}inp
ut[type=range]{-webkit-appearance:none;width:100%;height:22px;background:#363636;cursor:pointer;margin:0}input[type
=range]:focus{outline:0}input[type=range]::-webkit-slider-runnable-track{width:100%;height:2px;cursor:pointer;backgrou
nd:#EFEFEF;border-radius:0;border:0 solid #EFEFEF}input[type=range]::-webkit-slider-thumb{border:1px solid
rgba(0,0,30,0);height:22px;width:22px;border-radius:50px;background:#ff3034;cursor:pointer;-webkit-appearance:none;m
argin-top:-11.5px}input[type=range]:focus::-webkit-slider-runnable-track{background:#EFEFEF}input[type=range]::-moz-r
ange-track{width:100%;height:2px;cursor:pointer;background:#EFEFEF;border-radius:0;border:0 solid
#EFEFEF}input[type=range]::-moz-range-thumb{border:1px solid
rgba(0,0,30,0);height:22px;width:22px;border-radius:50px;background:#ff3034;cursor:pointer}input[type=range]::-ms-trac
k{width:100%;height:2px;cursor:pointer;background:0
0;border-color:transparent;color:transparent}input[type=range]::-ms-fill-lower{background:#EFEFEF;border:0 solid
#EFEFEF;border-radius:0}input[type=range]::-ms-fill-upper{background:#EFEFEF;border:0 solid
#EFEFEF;border-radius:0}input[type=range]::-ms-thumb{border:1px solid
rgba(0,0,30,0);height:22px;width:22px;border-radius:50px;background:#ff3034;cursor:pointer;height:2px}input[type=range
]:focus::-ms-fill-lower{background:#EFEFEF}input[type=range]:focus::-ms-fill-upper{background:#363636}.switch{display:
block;position:relative;line-height:22px;font-size:16px;height:22px}.switch
input{outline:0;opacity:0;width:0;height:0}.slider{width:50px;height:22px;border-radius:22px;cursor:pointer;background-col
or:grey}.slider,.slider:before{display:inline-block;transition:.4s}.slider:before{position:relative;content:"";border-radius:50%;
height:16px;width:16px;left:4px;top:3px;background-color:#fff}input:checked+.slider{background-color:#ff3034}input:chec
ked+.slider:before{-webkit-transform:translateX(26px);transform:translateX(26px)}select{border:1px solid
#363636;font-size:14px;height:22px;outline:0;border-radius:5px}.image-container{position:relative;min-width:160px}.close
{position:absolute;right:5px;top:5px;background:#ff3034;width:16px;height:16px;border-radius:100px;color:#fff;text-align:c
enter;line-height:18px;cursor:pointer}.hidden{display:none}
      </style>
   </head>
   <body>
   <figure>
```

```
    <div id="stream-container" class="image-container hidden">
      <div class="close" id="close-stream">×</div>
      <img id="stream" src="">
    </div>
  </figure>
    <section class="main" align="center">
      <section id="buttons">
       <table>
        <tr>
          <td align="center">
          <button id="get-still">Guardar Foto</button>
          </td>
          <td></td>
          <td align="center">
          <button id="toggle-stream">Camara </button>
          </td>
        </tr>
        <tr>
          <td>
          <input type="checkbox" id="nostop" onclick="var noStop=0;if (this.checked)
noStop=1;fetch(document.location.origin+'/control?var=nostop&val='+noStop);">Manual/autómatico
          </td>
          <td align="center">
          <button id="forward" onclick="fetch(document.location.origin+'/control?var=car&val=1');">Adelante</button>
          </td>
          <td></td>
        </tr>
        <tr>
          <td align="center">
          <button id="turnleft" onclick="fetch(document.location.origin+'/control?var=car&val=2');">Izquierda</button>
          </td>
          <td align="center">
          <button id="stop" onclick="fetch(document.location.origin+'/control?var=car&val=3');">Stop</button>
          </td>
          <td align="center">
          <button id="turnright" onclick="fetch(document.location.origin+'/control?var=car&val=4');">Derecha</button>
          </td>
        </tr>
        <tr>
          <td></td>
          <td align="center">
          <button id="backward" onclick="fetch(document.location.origin+'/control?var=car&val=5');">Atras</button>
          </td>
          <td></td>
        </tr>
        <tr>
          <td></td>
          <td align="center">
          <button id="Gfoto" onclick="fetch(document.location.origin+'/control?var=car&val=6');">Guardar foto</button>
          </td>
          <td></td>
        </tr>
          <tr><td>Flash</td>
          <td align="center" colspan="2">
          <input type="range" id="flash" min="0" max="255" value="0"
onchange="try{fetch(document.location.origin+'/control?var=flash&val='+this.value);}catch(e){}">
          </td>
        </tr>
          <tr><td>Velocidad</td>
          <td align="center" colspan="2">
          <input type="range" id="speed" min="0" max="255" value="255"
onchange="try{fetch(document.location.origin+'/control?var=speed&val='+this.value);}catch(e){}">
          </td>
        </tr>
          <tr><td>Servo</td>
          <td align="center" colspan="2">
          <input type="range" id="servo" min="0" max="255" value="90"
onchange="try{fetch(document.location.origin+'/control?var=servo&val='+this.value);}catch(e){}">
```

```
            </td>
           </tr>
            <tr><td>Calidad</td>
            <td align="center" colspan="2">
            <input type="range" id="quality" min="10" max="63" value="10"
onchange="try{fetch(document.location.origin+'/control?var=quality&val='+this.value);}catch(e){}">
            </td>
           </tr>
            <tr><td>Resolución</td>
            <td align="center" colspan="2">
            <input type="range" id="framesize" min="0" max="6" value="5"
onchange="try{fetch(document.location.origin+'/control?var=framesize&val='+this.value);}catch(e){}">
            </td>
           </tr>
           </table>
         </section>
      </section>
      <script>
        document.addEventListener('DOMContentLoaded',function(){function b(B){let
C;switch(B.type){case'checkbox':C=B.checked?1:0;break;case'range':case'select-one':C=B.value;break;case'button':case
'submit':C='1';break;default:return;}const D=`${c}/control?var=${B.id}&val=${C}`;fetch(D).then(E=>{console.log(`request to
${D} finished, status: ${E.status}`)})}var c=document.location.origin;const
e=B=>{B.classList.add('hidden')},f=B=>{B.classList.remove('hidden')},g=B=>{B.classList.add('disabled'),B.disabled=!0},h=
B=>{B.classList.remove('disabled'),B.disabled=!1},i=(B,C,D)=>{D=!(null!=D)||D;let
E;'checkbox'===B.type?(E=B.checked,C=!!C,B.checked=C):(E=B.value,B.value=C),D&&E!==C?b(B):!D&&('aec'===B.id?
C?e(v):f(v):'agc'===B.id?C?(f(t),e(s)):(e(t),f(s)):'awb_gain'===B.id?C?f(x):e(x):'face_recognize'===B.id&&(C?h(n):g(n)))};d
ocument.querySelectorAll('.close').forEach(B=>{B.onclick=()=>{e(B.parentNode)}}),fetch(`${c}/status`).then(function(B){re
turn B.json()}).then(function(B){document.querySelectorAll('.default-action').forEach(C=>{i(C,B[C.id],!1)})});const
j=document.getElementById('stream'),k=document.getElementById('stream-container'),l=document.getElementById('get-
still'),m=document.getElementById('toggle-stream'),n=document.getElementById('face_enroll'),o=document.getElementB
yId('close-stream'),p=()=>{window.stop(),m.innerHTML='Start
Stream'},q=()=>{j.src=`${c+':81'}/stream`,f(k),m.innerHTML='Stop
Stream'};l.onclick=()=>{p(),j.src=`${c}/capture?_cb=${Date.now()}`,f(k)},o.onclick=()=>{p(),e(k)},m.onclick=()=>{const
B='Stop
Stream'===m.innerHTML;B?p():q()},n.onclick=()=>{b(n)},document.querySelectorAll('.default-action').forEach(B=>{B.onch
ange=()=>b(B)});const
r=document.getElementById('agc'),s=document.getElementById('agc_gain-group'),t=document.getElementById('gainceili
ng-group');r.onchange=()=>{b(r),r.checked?(f(t),e(s)):(e(t),f(s))};const
u=document.getElementById('aec'),v=document.getElementById('aec_value-group');u.onchange=()=>{b(u),u.checked?e(
v):f(v)};const
w=document.getElementById('awb_gain'),x=document.getElementById('wb_mode-group');w.onchange=()=>{b(w),w.chec
ked?f(x):e(x)};const
y=document.getElementById('face_detect'),z=document.getElementById('face_recognize'),A=document.getElementById('
framesize');A.onchange=()=>{b(A),5<A.value&&(i(y,!1),i(z,!1))},y.onchange=()=>{return 5<A.value?(alert('Please select
CIF or lower resolution before enabling this feature!'),void
i(y,!1)):void(b(y),!y.checked&&(g(n),i(z,!1)))},z.onchange=()=>{return 5<A.value?(alert('Please select CIF or lower
resolution before enabling this feature!'),void i(z,!1)):void(b(z),z.checked?(h(n),i(y,!0)):g(n))})};
      </script>
   </body>
</html>
)rawliteral";
//.....................................................
static esp_err_t index_handler(httpd_req_t *req){
      Serial.println("-----------------------------------");
   Serial.println("index_handler");
    httpd_resp_set_type(req, "text/html");
    return httpd_resp_send(req, (const char *)INDEX_HTML, strlen(INDEX_HTML));
}

//.....................................................

void startCameraServer()
{
       Serial.println("-----------------------------------");
   Serial.println("startCameraServer");
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();

    httpd_uri_t index_uri = {
```

```
      .uri       = "/",
      .method    = HTTP_GET,
      .handler   = index_handler,
      .user_ctx  = NULL
    };

    httpd_uri_t status_uri = {
      .uri       = "/status",
      .method    = HTTP_GET,
      .handler   = status_handler,
      .user_ctx  = NULL
    };

    httpd_uri_t cmd_uri = {
      .uri       = "/control",
      .method    = HTTP_GET,
      .handler   = cmd_handler,
      .user_ctx  = NULL
    };

    httpd_uri_t capture_uri = {
      .uri       = "/capture",
      .method    = HTTP_GET,
      .handler   = capture_handler,
      .user_ctx  = NULL
    };

   httpd_uri_t stream_uri = {
      .uri       = "/stream",
      .method    = HTTP_GET,
      .handler   = stream_handler,
      .user_ctx  = NULL
    };

    Serial.printf("Starting web server on port: '%d'\n", config.server_port);
    if (httpd_start(&camera_httpd, &config) == ESP_OK) {
      httpd_register_uri_handler(camera_httpd, &index_uri);
      httpd_register_uri_handler(camera_httpd, &cmd_uri);
      httpd_register_uri_handler(camera_httpd, &status_uri);
      httpd_register_uri_handler(camera_httpd, &capture_uri);
    }

    config.server_port += 1;
    config.ctrl_port += 1;
    Serial.printf("Starting stream server on port: '%d'\n", config.server_port);
    if (httpd_start(&stream_httpd, &config) == ESP_OK) {
      httpd_register_uri_handler(stream_httpd, &stream_uri);
    }
} //star camara
//.....................................................

//Foto a driver
void saveCapturedImage() {
//  Serial.println("Connect to " + String(myDomain));
  WiFiClientSecure client;

  if (client.connect(myDomain, 443)) {
    Serial.println("Connection successful");

    camera_fb_t * fb = NULL;
    fb = esp_camera_fb_get();
    if(!fb) {
      Serial.println("Camera capture failed");
      delay(1000);
      ESP.restart();
      return;
    }
```

```
    char *input = (char *)fb->buf;
    char output[base64_enc_len(3)];
    String imageFile = "";
    for (int i=0;i<fb->len;i++) {
      base64_encode(output, (input++), 3);
      if (i%3==0) imageFile += urlencode(String(output));
    }
    String Data = myFilename+mimeType+myImage;

    esp_camera_fb_return(fb);

    Serial.println("Send a captured image to Google Drive.");

    client.println("POST " + myScript + " HTTP/1.1");
    client.println("Host: " + String(myDomain));
    client.println("Content-Length: " + String(Data.length()+imageFile.length()));
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.println();

    client.print(Data);
    int Index;
    for (Index = 0; Index < imageFile.length(); Index = Index+1000) {
      client.print(imageFile.substring(Index, Index+1000));
    }

    Serial.println("Waiting for response.");
    long int StartTime=millis();
    while (!client.available()) {
      Serial.print(".");
      delay(100);
      if ((StartTime+waitingTime) < millis()) {
        Serial.println();
        Serial.println("No response.");
        //If you have no response, maybe need a greater value of waitingTime
        break;
      }
    }
    Serial.println();
    while (client.available()) {
      Serial.print(char(client.read()));
    }
  } else {
    Serial.println("Connected to " + String(myDomain) + " failed.");
  }
  client.stop();
}


String Photo2Base64() {
    camera_fb_t * fb = NULL;
    fb = esp_camera_fb_get();
    if(!fb) {
      Serial.println("Camera capture failed");
      return "";
    }

    String imageFile = "";
    char *input = (char *)fb->buf;
    char output[base64_enc_len(3)];
    for (int i=0;i<fb->len;i++) {
      base64_encode(output, (input++), 3);
      if (i%3==0) imageFile += urlencode(String(output));
    }

    esp_camera_fb_return(fb);

    return imageFile;
}
```

```
//https://github.com/zenmanenergy/ESP8266-Arduino-Examples/
String urlencode(String str)
{
    String encodedString="";
    char c;
    char code0;
    char code1;
    char code2;
    for (int i =0; i < str.length(); i++){
      c=str.charAt(i);
      if (c == ' '){
        encodedString+= '+';
      } else if (isalnum(c)){
        encodedString+=c;
      } else{
        code1=(c & 0xf)+'0';
        if ((c & 0xf) >9){
            code1=(c & 0xf) - 10 + 'A';
        }
        c=(c>>4)&0xf;
        code0=c+'0';
        if (c > 9){
            code0=c - 10 + 'A';
        }
        code2='\0';
        encodedString+='%';
        encodedString+=code0;
        encodedString+=code1;
        //encodedString+=code2;
      }
      yield();
    }
    return encodedString;
}

}
```

Arduino

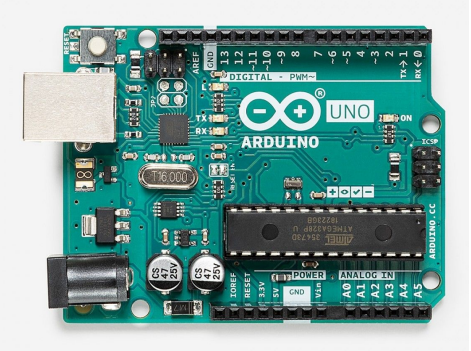
```
/*
 Control movil con arduino + wifi ESP32 Web y guarda foto en Driver
 Utiliza ESP32_CamCar_drive
PENDIENTE:
Funciona salvando obtaculos
 ESP32 solo para ver imagen  y guardar foto en Driver

*/
#include <Servo.h>

boolean inicio_paro = false ;// para iniciar programa y parar


Servo servoLook;                                // Crea un objeto para controlar el servo

#define trig  6        // sensor ultrasonidos
#define echo  7
#define pinServo 10
#define pinMAOn 3       // velocidad motor 0 - 255
#define pinMADir 12     // Sentido de giro motor
#define pinMBOn  11
#define pinMBDir 13
#define pinMAOff 9      //Paro motor a 1
#define pinMBOff 8
```

```
#define pinInt  2      // Int. paro

byte velocidad = 120;                    // Almacena velocidad motores
byte Distancia = 40;                     // Distancia objetos
byte maxDist = 90;                       // 150 Distancia máxima de detección (se ignoran los objetos más allá de esta
distancia)
byte stopDist = 60;                      // 50 Distancia mínima de un objeto a detenerse en cm
float timeOut = 2*(maxDist+10)/100/340*1000000;   // Tiempo máximo para esperar una señal de retorno

// Declaración de funciones
void Programa_automatico();
void Parar();
void Adelante();
void Atras();
void Derecha(int duracion);
void Izquierda(int duracion);
void CompruebaParo();
int CompruebaDireccion() ;
int ObtenerDistancia();


// PARO
void Parar() {
  digitalWrite(pinMAOff,HIGH);  // frena motores
  digitalWrite(pinMBOff,HIGH);
  Serial.println("Paro");
}

// AVANZA
void Adelante() {
   digitalWrite(pinMAOff,LOW);  //Quita freno
   digitalWrite(pinMBOff,LOW);
   analogWrite(pinMAOn, velocidad);  // Velocidad del motor
   analogWrite(pinMBOn, velocidad);  // Velocidad del motor
   digitalWrite(pinMADir,HIGH);
  digitalWrite(pinMBDir,HIGH);
  Serial.println("Adelante");

}
// RETROCEDE
void Atras() {
    digitalWrite(pinMAOff,LOW);  //Quita freno
    digitalWrite(pinMBOff,LOW);
   analogWrite(pinMAOn, velocidad);  // Velocidad del motor
   analogWrite(pinMBOn, velocidad);  // Velocidad del motor
  digitalWrite(pinMADir,LOW);
  digitalWrite(pinMBDir,LOW);
  Serial.println("Atras");

}

// GIRO DERECHA
void Derecha(int duracion) {
   digitalWrite(pinMAOff,LOW);  //Quita freno
   digitalWrite(pinMBOff,LOW);
   analogWrite(pinMAOn, velocidad);  // Velocidad del motor
   analogWrite(pinMBOn, velocidad);  // Velocidad del motor
  digitalWrite(pinMADir,HIGH);
  digitalWrite(pinMBDir,LOW);
  delay(duracion);
  Parar();
  Serial.println("Giro Derecha");
}

// GIRO IZQUIERDA

void Izquierda(int duracion) {
```

```
   digitalWrite(pinMAOff,LOW);  //Quita freno
   digitalWrite(pinMBOff,LOW);
   analogWrite(pinMAOn, velocidad);  // Velocidad del motor
   analogWrite(pinMBOn, velocidad);  // Velocidad del motor
  digitalWrite(pinMADir,LOW);
  digitalWrite(pinMBDir,HIGH);
  delay(duracion);
  Parar();
  Serial.println("Giro Izquierda");
}

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

  pinMode(pinServo, OUTPUT);
  servoLook.attach(pinServo);                  //Asigna pin al servo
  pinMode(trig,OUTPUT);                        //Asigna modo salida al sensor de ultrasonido
  pinMode(echo,INPUT);

  pinMode(pinInt, INPUT);                       //Interruptor Inicio/Paro A0

  inicio_paro = false;
  pinMode(pinMADir,OUTPUT);                        // MA giro
  pinMode(pinMBDir,OUTPUT);                        // MB giro
  pinMode(pinMAOff, OUTPUT);
  pinMode(pinMBOff, OUTPUT);
  Parar();                                    // Para motores
}


void Programa_automatico(){
  velocidad=255;
  servoLook.write(90);                        // Configura el servo para que mire hacia adelante
  delay(750);
  int distance = ObtenerDistancia();             // Comprueba que no haya objetos delante
  if(distance >= stopDist) {                   // Si no hay objetos dentro de la distancia de frenado, avance
      Adelante() ;
  }
  while(distance >= stopDist){                  // Siga controlando la distancia del objeto hasta que esté dentro de la distancia
mínima de frenado
    distance = ObtenerDistancia();
    CompruebaParo();
    delay(250);
  }
  Parar();                                   // Parar los motores
  int turnDir = CompruebaDireccion();            // Verifique las distancias de los objetos a la izquierda y a la derecha y
obtenga las instrucciones de giro
  Serial.print(turnDir);
  switch (turnDir){                            // Girar a la izquierda, dar la vuelta o girar a la derecha según las instrucciones
     case 0:                                  //Girar a la izquierda
      Izquierda (400);
      break;
    case 1:                                   //Giro de vuelta
      Izquierda (700);
      break;
    case 2:                                   //Girar a la derecha
      Derecha (400);
      break;
  }
}

void loop(){
   CompruebaParo();
}//loop

// Comprueba botón de paro
void CompruebaParo(){
```

```
   if (digitalRead(pinInt) == HIGH){ //PARAR
    Parar();
    exit(0);
   }
   Programa_automatico();
 }

  int ObtenerDistancia() {                    //Mide la distancia a un objeto
  unsigned long pulseTime;                    //Cree una variable para almacenar el tiempo de viaje del pulso
  int distance;                               //Crea una variable para almacenar la distancia calculada
  digitalWrite(trig, HIGH);                   //Genera un pulso de 10 microsecond
  delayMicroseconds(10);
  digitalWrite(trig, LOW);
  pulseTime = pulseIn(echo, HIGH, timeOut);         //Mide el tiempo que tarda el pulso en regresar
  distance = (float)pulseTime * 340 / 2 / 10000;
  Serial.println(distance);
  return distance;
}

int CompruebaDireccion() {                    // Verifique las direcciones izquierda y derecha y decida en qué dirección
girar
  int distances [2] = {0,0};                  // Distancias izquierda y derecha
  int turnDir = 1;                            // Dirección de giro, 0 izquierda, 1 marcha atrás, 2 derecha
  servoLook.write(150);                       // Gira el servo para mirar a la izquierda
  delay(500);
  distances [0] = ObtenerDistancia();         // Obtener la distancia del objeto a la izquierda
  servoLook.write(30);                        // Gira el servo para mirar a la derecha
  delay(1000);
  distances [1] = ObtenerDistancia();         // Obtenga la distancia correcta del objeto
  if (distances[0]>=Distancia && distances[1]>=Distancia)      // Si ambas direcciones están despejadas, gire a la
izquierda
    turnDir = 0;
  else if (distances[0]<=stopDist && distances[1]<=stopDist)    // Si ambas direcciones están bloqueadas, da la vuelta
    turnDir = 1;
  else if (distances[0]>=distances[1])          // Si a la izquierda le queda más espacio, gire a la izquierda
    turnDir = 0;
  else if (distances[0]<distances[1])           // Si la derecha tiene más espacio, gire a la derecha
    turnDir = 2;
  return turnDir;
}
```

Pines Arduino:

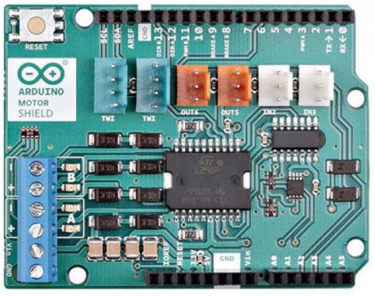


0	
1	

2	Pulsador Paro/inicio
3	MA on/off
4	
5	
6	Trigger
7	Echo
8	MB Freno
9	MA Freno Servo
10	Servo
11	MB on/off
12	MA giro
13	MB giro
A0	MA Corriente
A1	MB Corriente
A2	
A3	
A4	
A5	

Para detener los motores se puede utilizar como salidas D9 y D8 poniendolas a 1 y a cero para quitar freno.
Tambien se puede parar los motores poniendo Las salidas de arduino PWM D3 D11 a 0.

Funcionamiento de los motores:



Pines Arduino	Paro	Avanza	Retrocede	Giro Izq.	Giro Der.
D3 (MA on/off)	0 ó D9=1	0..255	0..255	0..255	0..255
D11 (MB on/off)	0 ó D8=1	0..255	0..255	0..255	0..255
D12 (MA giro)		HIGH	LOW	LOW	HIGH
D13 (MB giro)		HIGH	LOW	HIGH	LOW

Enlaces con información utilizados:

- Control movil con ESP32-Cam:
  https://github.com/alfajor144/ESP32-proyects/tree/master/ESP32CAM
  Videos:
  https://www.youtube.com/watch?v=b7Gz73nUAXU

- Camara subir fotos FTP
  https://www.gsampallo.com/2020/04/20/subir-fotos-a-un-servidor-ftp-con-esp32-cam/

- Generar PWM:
  https://randomnerdtutorials.com/esp32-pwm-arduino-ide/

- Subir fotos a Google Driver:
  https://github.com/raphaelbs/esp32-cam-ai-thinker/tree/master/examples/google_storage

- Control de obtaculos:
  https://www.the-diy-life.com/arduino-based-obstacle-avoiding-robot-car/

Fotos: