Práctica: Web Socket y Detección facial

Objetivo:

- Emplear un web-socket como medio de comunicación entre los datos ingresados a través de stream de video y pos-procesamiento para detección facial

Introducción

Una de las mayores tendencias dentro de la computación es la inteligencia artificial, desde algoritmos de reconocimiento facial para aplicaciones médicas e incluso cuestiones de seguridad hasta algoritmos de predicción de datos, son algunos ejemplos que hoy en día son una realidad.

Descripción:

En la presente práctica se abordará la detección de rostros a través del uso de un web socket instanciado dentro del mismo ESP32-CAM, sin embargo, para poder entender cómo se realizará a continuación se muestra un diagrama de descripción de procesos:

Tabla de entradas y salidas:

Debido a que en este caso solamente se hará uso de la cámara de nuestro ESP32, no es necesario definir otros pines, sin embargo, debemos saber que pines son los que se usan para la adquisición de datos a través de la cámara.

A continuación, se muestra la declaración de estos puertos dentro de nuestro código:

```
#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
```

Para más información acerca de los puertos dedicados exclusivamente para la cámara del ESP32, puede revisarse el datasheet del ESP32 - AI module o el del Wrove-module.

Diagrama de conexiones:

A continuación, se muestra el diagrama de conexiones:

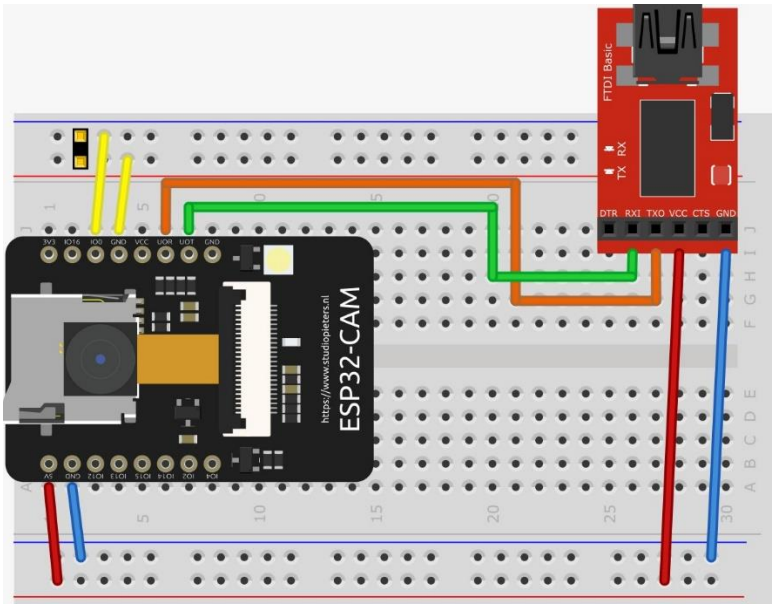

Imagen 1. Diagrama de conexiones en el ESP32

Recordemos que, una vez descargado el código en memoria, es necesario desconectar el pin 0 del 16.

Listado del programa y descripción:

A continuación, se muestra paso a paso cada una de las partes que conforman el proyecto en código de la presente práctica, el cual está constituido de 4 archivos, el cam_web_server.ino, app_httpd.cpp, y 2 bibliotecas para definición de puertos y aspectos de la cámara

1. **Definición de puertos y configuración de la cámara**

Para empezar, debemos hacer explicito los puertos a utilizar de nuestro ESP32 cam para lo que el fabricante recomienda el uso de la biblioteca denominada como “camera_pins.h” debido a que su uso está planteado para cualquier variante de ESP32:

cam__WebServer | app_httpd.cpp | camera_index.h | camera_pins.h

```
#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_ESP_EYE)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM    4
#define SIOD_GPIO_NUM    18
#define SIOC_GPIO_NUM    23

#define Y9_GPIO_NUM      36
#define Y8_GPIO_NUM      37
#define Y7_GPIO_NUM      38
#define Y6_GPIO_NUM      39
```

Imagen 2. Biblioteca genérica de Expressif

Como segundo recurso auxiliar debemos emplear la biblioteca de “camera_index.h” debido a que esta es la que tiene el binario compatible para el stream de video además de aspectos como filtros añadidos para la corrección de aberración cromática, entre otros:

cam__WebServer | app_httpd.cpp | camera_index.h | camera_pins.h

```
//File: index_ov2640.html.gz, Size: 4316
#define index_ov2640_html_gz_len 4316
const uint8_t index_ov2640_html_gz[] = {
 0x1F 0x8B 0x08 0x08 0x50 0x5C 0xAE 0x5C 0x00 0x03 0x69 0x6E 0x64 0x65 0x78 0x5F
 0x6F 0x76 0x32 0x36 0x34 0x30 0x2E 0x68 0x74 0x6D 0x6C 0x00 0xE5 0x5D 0x7B 0x73
 0xD3 0xC6 0x16 0xFF 0x9F 0x4F 0x21 0x04 0x25 0xF6 0x34 0x76 0x6C 0xC7 0x84 0xE0
 0xDA 0xE2 0x42 0x08 0xD0 0x19 0x5E 0x25 0x2D 0x74 0xA6 0xD3 0x81 0xB5 0xB4 0xB2
 0x55 0x64 0xC9 0x95 0x56 0x76 0x52 0x26 0x9F 0xE3 0x7E 0xA0 0xFB 0xC5 0xEE 0xD9
 0x87 0xA4 0x95 0xBC 0x7A 0xD8 0x26 0x36 0x97 0xEB 0xCC 0x14 0xD9 0xDA 0x73 0xF6
 0x9C 0xF3 0x3B 0xAF 0x5D 0x3D 0x3A 0xBC 0x6D 0xF9 0x26 0xB9 0x9A 0x63 0x6D 0x4A
 0x66 0xAE 0x71 0x6B 0xC8 0xFF 0xD1 0xE0 0x33 0x9C 0x62 0x64 0xF1 0x43 0xF6 0x75
 0x86 0x09 0xD2 0xCC 0x29 0x0A 0x42 0x4C 0x46 0x7A 0x44 0xEC 0xD6 0xA9 0x9E 0x3F
 0xED 0xA1 0x19 0x1E 0xE9 0x0B 0x07 0x2F 0xE7 0x7E 0x40 0x74 0xCD 0xF4 0x3D 0x82
 0x3D 0x18 0xBE 0x74 0x2C 0x32 0x1D 0x59 0x78 0xE1 0x98 0xB8 0xC5 0xBE 0x1C 0x3A
 0x9E 0x43 0x1C 0xE4 0xB6 0x42 0x13 0xB9 0x78 0xD4 0x95 0x79 0x11 0x87 0xB8 0xD8
 0x38 0xBF 0x78 0x7B 0xDC 0xD3 0xDE 0xBC 0xEF 0xF5 0x4F 0x3A 0xC3 0x23 0xFE 0x5B
 0x3A 0x26 0x24 0x57 0xF2 0x77 0xFA 0x19 0xFB 0xD6 0x95 0xF6 0x25 0xF3 0x13 0xFD
 0xD8 0x20 0x44 0xCB 0x46 0x33 0xC7 0xBD 0x1A 0x68 0x8F 0x03 0x98 0xF3 0xF0 0x05
 0x76 0x17 0x98 0x38 0x26 0x3A 0x0C 0x91 0x17 0xB6 0x42 0x1C 0x38 0xF6 0x4F 0x2B
 0x84 0x63 0x64 0x7E 0x9E 0x04 0x7E 0xE4 0x59 0x03 0xED 0x4E 0xF7 0x94 0xFE 0xAD
 0x0E 0x32 0x7D 0xD7 0x0F 0xE0 0xFC 0xF9 0x33 0xFA 0xB7 0x7A 0x9E 0xCD 0x1E 0x3A
 0xFF 0xE0 0x81 0xD6 0x3D 0x99 0x5F 0x66 0xCE 0x5F 0xDF 0xCA 0x7C 0x9D 0xF6 0x8A
 0xA4 0x17 0xF4 0xA7 0xE5 0xF4 0x21 0x36 0x89 0xE3 0x7B 0xED 0x19 0x72 0x3C 0x05
 0x27 0xCB 0x09 0xE7 0x2E 0x02 0x1B 0xD8 0x2E 0x2E 0xE5 0x73 0x67 0x86 0xBD 0xE8
 0xB0 0x82 0x1B 0x65 0xD2 0xB2 0x9C 0x80 0x8F 0x1A 0x50 0x3B 0x44 0x33 0xAF 0x92
 0x6D 0x99 0x5C 0x9E 0xEF 0x61 0x85 0x01 0xE9 0x44 0xCB 0x00 0xCD 0xE9 0x00 0xFA
 0xEF 0xEA 0x90 0x99 0xE3 0x71 0xA7 0x1A 0x68 0xC7 0xFD 0xCE 0xFC 0xB2 0x02 0xCA
 0xE3 0x13 0xFA 0xB7 0x3A 0x68 0x8E 0x2C 0xCB 0xF1 0x26 0x03 0xED 0x54 0xC9 0xC2
 0x0F 0x2C 0x1C 0xB4 0x02 0x64 0x39 0x51 0x38 0xD0 0xFA 0xAA 0x31 0x33 0x14 0x4C
```

Imagen 3. Como se puede observar la biblioteca solamente tiene binario en un arreglo que es usado en el código principal del proyecto

Cabe destacar que en futuros proyectos ya no se hará explicito el uso de esta biblioteca, sin embargo, debemos entender cuál es su funcionalidad además de qué es lo que contiene.

2. Código principal y configuración del web socket para transmisión de video

Para el desarrollo del web socket y una interfaz gráfica de usuario web (GUI) se empleó como plantilla el código del web -socket desarrollado por la misma Expressif, por lo que a continuación se mencionan meramente los cambios contemplados:

2.1 Web socket y desarrollo web

Para este apartado de manera general se partirá el código en dos segmentos principales, el primero es la creación del Web-Socket y la segunda es el desarrollo web de la interfaz con la que

se interactuará para la detección facial:

A continuación, se muestran las bibliotecas generales que deben usarse para el stream de video, creación del web-socket además de la conversión de imágenes para juckings temporales:

```
#include "esp_http_server.h"
#include "esp_timer.h"
#include "esp_camera.h"
#include "img_converters.h"
#include "camera_index.h"
#include "Arduino.h"

#include "fb_gfx.h"
#include "fd_forward.h"
#include "fr_forward.h"

#define ENROLL_CONFIRM_TIMES 5
#define FACE_ID_SAVE_NUMBER 7

#define FACE_COLOR_WHITE  0x00FFFFFF
#define FACE_COLOR_BLACK  0x00000000
#define FACE_COLOR_RED    0x000000FF
#define FACE_COLOR_GREEN  0x0000FF00
#define FACE_COLOR_BLUE   0x00FF0000
#define FACE_COLOR_YELLOW (FACE_COLOR_RED | FACE_COLOR_GREEN)
#define FACE_COLOR_CYAN   (FACE_COLOR_BLUE | FACE_COLOR_GREEN)
#define FACE_COLOR_PURPLE (FACE_COLOR_BLUE | FACE_COLOR_RED)

typedef struct {
        size_t size; //number of values used for filtering
        size_t index; //current value index
        size_t count; //value count
        int sum;
        int * values; //array to be filled with values
} ra_filter_t;

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;
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";
```

Imagen 4. Bibliotecas y recursos generales para el proyecto

Por otro lado y como bien se mencionó anteriormente, hay varias funciones que tiene este archivo qyue previamente fueron hechas para poder hacer el stream de video como son el caso de ra_filter_run , rgb_print y rgb_printf que se encargan de hacer posible desde la adquisición hasta la conversión de lo obtenido del sensor de la cámara a un video legible por el web socket, sin embargo, para poder hacer la detección de rostros lo primero por lo que se partió fue hacer un cuadro de color para asimilar cada rostro mediante un ID reconocible dentro de la interfaz web

como mediante el monitor serial:

```
78  > static int ra_filter_run(ra_filter_t * filter, int value){···
92
93  > static void rgb_print(dl_matrix3du_t *image_matrix, uint32_t color, const char * str){···
102
103 > static int rgb_printf(dl_matrix3du_t *image_matrix, uint32_t color, const char *format, ...){···
127
128 static void draw_face_boxes(dl_matrix3du_t *image_matrix, box_array_t *boxes, int face_id){
129     int x, y, w, h, i;
130     uint32_t color = FACE_COLOR_YELLOW;
131     if(face_id < 0){
132         color = FACE_COLOR_RED;
133     } else if(face_id > 0){
134         color = FACE_COLOR_GREEN;
135     }
136     fb_data_t fb;
137     fb.width = image_matrix->w;
138     fb.height = image_matrix->h;
139     fb.data = image_matrix->item;
140     fb.bytes_per_pixel = 3;
141     fb.format = FB_BGR888;
142     for (i = 0; i < boxes->len; i++){
143         // rectangle box
144         x = (int)boxes->box[i].box_p[0];
145         y = (int)boxes->box[i].box_p[1];
146         w = (int)boxes->box[i].box_p[2] - x + 1;
147         h = (int)boxes->box[i].box_p[3] - y + 1;
148         fb_gfx_drawFastHLine(&fb, x, y, w, color);
149         fb_gfx_drawFastHLine(&fb, x, y+h-1, w, color);
150         fb_gfx_drawFastVLine(&fb, x, y, h, color);
151         fb_gfx_drawFastVLine(&fb, x+w-1, y, h, color);
152 #if 0
153         // landmark
154         int x0, y0, j;
155         for (j = 0; j < 10; j+=2) {
156             x0 = (int)boxes->landmark[i].landmark_p[j];
157             y0 = (int)boxes->landmark[i].landmark_p[j+1];
158             fb_gfx_fillRect(&fb, x0, y0, 3, 3, color);
159         }
160 #endif
161     }
```

Imagen 5. Dibujo de los rostros

Por otro lado, también se hizo una función dedicada para el reconocimiento facial que si bien puede parecer lo mismo que la detección facial no es lo mismo, la diferencia radica que mientras la detección radica meramente en asimilar la forma humana del rostro, el reconocimiento asimila un ID a cada rostro que va detectando:

```
164 static int run_face_recognition(dl_matrix3du_t *image_matrix, box_array_t *net_boxes){
165     dl_matrix3du_t *aligned_face = NULL;
166     int matched_id = 0;
167 
168     aligned_face = dl_matrix3du_alloc(1, FACE_WIDTH, FACE_HEIGHT, 3);
169     if(!aligned_face){
170         Serial.println("Could not allocate face recognition buffer");
171         return matched_id;
172     }
173     if (align_face(net_boxes, image_matrix, aligned_face) == ESP_OK){
174         if (is_enrolling == 1){
175             int8_t left_sample_face = enroll_face(&id_list, aligned_face);
176 
177             if(left_sample_face == (ENROLL_CONFIRM_TIMES - 1)){
178                 Serial.printf("Enrolling Face ID: %d\n", id_list.tail);
179             }
180             Serial.printf("Enrolling Face ID: %d sample %d\n", id_list.tail, ENROLL_CONFIRM_TIMES - left_sample_face);
181             rgb_printf(image_matrix, FACE_COLOR_CYAN, "ID[%u] Sample[%u]", id_list.tail, ENROLL_CONFIRM_TIMES - left_sample_face);
182             if (left_sample_face == 0){
183                 is_enrolling = 0;
184                 Serial.printf("Enrolled Face ID: %d\n", id_list.tail);
185             }
186         } else {
187             matched_id = recognize_face(&id_list, aligned_face);
188             if (matched_id >= 0) {
189                 Serial.printf("Match Face ID: %u\n", matched_id);
190                 rgb_printf(image_matrix, FACE_COLOR_GREEN, "Hello Subject %u", matched_id);
191             } else {
192                 Serial.println("No Match Found");
193                 rgb_print(image_matrix, FACE_COLOR_RED, "Intruder Alert!");
194                 matched_id = -1;
195             }
196         }
197     } else {
198         Serial.println("Face Not Aligned");
199         //rgb_print(image_matrix, FACE_COLOR_YELLOW, "Human Detected");
200     }
201 
202     dl_matrix3du_free(aligned_face);
203     return matched_id;
```

Imagen 6. Reconocimiento facial mediante el uso de matrices de colores y formas

Por último y como parte de la creación del web socket y conexión con los datos previos, continuación se muestran las funciones encargadas del stram de video y además la función que inicia e instancia la cámara ya en un servidor:

```
206 > static size_t jpg_encode_stream(void * arg, size_t index, const void* data, size_t len){···
217
218 > static esp_err_t capture_handler(httpd_req_t *req){···
304
305 > static esp_err_t stream_handler(httpd_req_t *req){···
451
452 > static esp_err_t cmd_handler(httpd_req_t *req){···
538
539 > static esp_err_t status_handler(httpd_req_t *req){···
580
581 > static esp_err_t index_handler(httpd_req_t *req){···
590
591   void startCameraServer(){
592       httpd_config_t config = HTTPD_DEFAULT_CONFIG();
593
594       httpd_uri_t index_uri = {
595           .uri       = "/",
596           .method    = HTTP_GET,
597           .handler   = index_handler,
598           .user_ctx  = NULL
599       };
600
601       httpd_uri_t status_uri = {
602           .uri       = "/status",
603           .method    = HTTP_GET,
604           .handler   = status_handler,
605           .user_ctx  = NULL
606       };
607
608       httpd_uri_t cmd_uri = {
609           .uri       = "/control",
610           .method    = HTTP_GET,
611           .handler   = cmd_handler,
612           .user_ctx  = NULL
613       };
```

Imagen 7. Funciones para manejo del stream de video

2.2 Instancia de estructuras y declaración de ambiente

Como parte final del código, en el archivo “.ino” del proyecto solamente se realizaron las instancias de cada recurso, estructura y función de los archivos previos, cabe resaltar que para este caso en concreto la función loop no contiene nada por la forma en que fue realizado el código:

```
#include "esp_camera.h"
#include <WiFi.h>
#define CAMERA_MODEL_AI_THINKER

#include "camera_pins.h"

const char* ssid = "murryFly";
const char* password = "oQEeCponFG";

void startCameraServer();

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  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;
```

Imagen 8. Declaración de puerto y uso de las bibliotecas previas

```
38
39    if(psramFound()){
40      config.frame_size = FRAMESIZE_UXGA;
41      config.jpeg_quality = 10;
42      config.fb_count = 2;
43    } else {
44      config.frame_size = FRAMESIZE_SVGA;
45      config.jpeg_quality = 12;
46      config.fb_count = 1;
47    }
48
49  #if defined(CAMERA_MODEL_ESP_EYE)
50    pinMode(13, INPUT_PULLUP);
51    pinMode(14, INPUT_PULLUP);
52  #endif
53
54    esp_err_t err = esp_camera_init(&config);
55    if (err != ESP_OK) {
56      Serial.printf("Camera init failed with error 0x%x", err);
57      return;
58    }
59
60    sensor_t * s = esp_camera_sensor_get();
61    if (s->id.PID == OV3660_PID) {
62      s->set_vflip(s, 1);
63      s->set_brightness(s, 1);
64      s->set_saturation(s, -2);
65    }
66    s->set_framesize(s, FRAMESIZE_QVGA);
67
68  #if defined(CAMERA_MODEL_M5STACK_WIDE)
69    s->set_vflip(s, 1);
70    s->set_hmirror(s, 1);
71  #endif
72
73    WiFi.begin(ssid, password);
74
75    while (WiFi.status() != WL_CONNECTED) {
76      delay(500);
77      Serial.print(".");
78    }
79    Serial.println("");
80    Serial.println("WiFi connected");
81
82    startCameraServer();
83
84    Serial.print("Camera Ready! Use 'http://");
85    Serial.print(WiFi.localIP());
86    Serial.println("' to connect");
87  }
88
89  void loop() {
90    delay(10000);
91  }
92
```

Imagen 9. Código principal del proyecto