EJEMPLO COMUNICACIÓN TCP ENTRE PC y servidor de vídeo http

Mi PC, tfno móvil, etc. (cliente)

WiFi connect to Access Point

WiFiAp Request password

WiFi Send password

WiFi Accept

IEEE 802.11 protocol

DHCP get dynamic ip address

DHCP send dynamic ip address

DHCP protocol

TCP Connect
ip 192.168.23.1 port 81

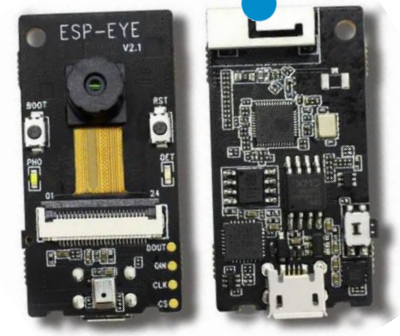
TCP accept

TCP protocol

HTTP Send "GET ...."

HTTP Send image

HTTP over TCP protocol



ESP32-CAM O ESP-EYE

Access Point MyESPCam
DHCP server
IP: 192.168.23.1
Video stream server port: 81
Video stream format: http/image (jpeg)

PROGRAMA PRINCIPAL PC (Qt-Widgets)

```
// EJEMPLO DE APLICACION PARA OBTENER IMAGENES DE UN SERVIDOR WEB (http)
// 1) Conecta con el servidor http (usar 182.168.23.1), puerto 81
// 2) Prepara solicitud para la pagina /stream
// 3) Envia la solicitud y recibe el texto html, donde la imagen está embebida
// 4) Extrae la imagen, la descomprime y la guarda en un archivo

#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>

void EndProgram(const char* msg,int err);

int main()
{
    WSADATA wsaData;
    WORD wVersionRequerida = MAKEWORD (2, 2);
    SOCKET sock;
    struct sockaddr_in local,remoto;
    char txt_snd[1024],txt_rcv[65536];
    char txt_page[1024],txt_host[1024];
    int n;

    // Necesario en Windows para inicializar sockets para la aplicación
    WSAStartup (wVersionRequerida, &wsaData);

    // Obtener identificador para un socket TCP
    sock=socket(AF_INET,SOCK_STREAM,0);
    ... Check sock ...
    // Preparar estructura con datos de IP/puerto local y realizar bind()
    local.sin_family=AF_INET;
    local.sin_addr.S_un.S_addr=0;   // 0 = cualquier dirección IP disponible
    local.sin_port=htons(0);        // 0 = cualquier puerto disponible
    memset(local.sin_zero,0,8*sizeof(char));
    if (bind(sock,(struct sockaddr*) &local,sizeof(struct sockaddr_in))!=0)
    {
        printf("No se puede asignar dirección local. Programa terminado\n");
        return -2;
    }

    printf("Pulse RETURN para conectar con el servidor de stream: ");
    getchar();
```

```
    // Preparar estructura con datos de IP/puerto remoto y realizar connect()
    remoto.sin_family=AF_INET;
    remoto.sin_addr.S_un.S_addr=inet_addr("192.168.23.1");
    remoto.sin_port=htons(81);
    memset(remoto.sin_zero,0,8*sizeof(char));

    if (connect(sock,(struct sockaddr*) &remoto,sizeof(struct sockaddr_in))!=0) {
       EndProgram("No se puede conectar a servidor remoto",-2);
    }
    printf("Conectado con ESP-CAM\n\n");

    // Preparar campos para GET: página a visitar y servidor
    strcpy(txt_snd,"GET /stream HTTP/1.1\r\n"
       "Host: 192.168.23.1:81\r\n"
       "Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8\r\n"
       "Connection: keep-alive\r\n\r\n"     );

       // Enviar texto al remoto
       n=send(sock,txt_snd,strlen(txt_snd),0);
       ... Check n ...

       // Esperar respuesta
       n=WaitAndReceive(sock,txt_rcv,65536);

       // Obtener campos relevantes de respuesta
       int answerIsOk=0;
       const char *ptType=GetContentsAfter(txt_rcv,"Content-Type:");
        ... Check ptType is "image/jpeg" ...

        int len=-1;
        const char* ptLen=GetContentsAfter(txt_rcv,"Content-Length:");
        ... Check ptLen is valid ...
        len=atoi(ptLen);  // check ptLen

        char *ptStart,*ptEnd;
        ptStart=strstr(txt_rcv,"\r\n\r\n\r\n");
        ... Check ptStart is valid ...
        ptStart=strstr(ptStart+6,"\r\n");
        ... Check ptStart is valid ...

        FILE* fid=fopen("imagen_camara.jpg","wb");
        if (fid!=NULL) {
            fwrite(ptStart+2,1,len,fid);
            fclose(fid);
        }
        ... Close socket and end program ...
```

PROGRAMA PRINCIPAL ESP32-CAM

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

const char* ssid_ap = "MyESPCam"; // TBD BY USER. Clients must connect with this SSID
const char* password_ap = "my-password-for-this-access-point"; // TBD BY USER
IPAddress local_ip(192,168,23,1);   // TBD BY USER. Clients should connect to
                                    // sockets in ports 80/81 this IP address
IPAddress gateway(192,168,23,1);    // TBD BY USER
IPAddress subnet(255,255,255,0);    // TBD BY USER


void setup()
{
   ...
   // Init camera
   camera_config_t config;
   config.pin_vsync= ...;
   ...
   esp_camera_init(&config);

   WiFi.mode(WIFI_AP);
   WiFi.softAP(ssid_ap, password_ap);        //Start Acces point mode
   WiFi.softAPConfig(local_ip, gateway, subnet);

   startCameraServer();
}

void loop() {
   ...
}
```

```
httpd_handle_t stream_httpd = NULL;
...

void startCameraServer() {
   httpd_config_t config_stream = HTTPD_DEFAULT_CONFIG();
   config_stream.server_port=81;
   httpd_start(&stream_httpd, &config_stream);
   ...
   httpd_uri_t stream_uri = {
      .uri = "/stream",
      .method = HTTP_GET,
      .handler = stream_handler,
      .user_ctx = NULL
    };
   httpd_register_uri_handler(camera_httpd, &stream_uri);
   ...
}
```

Función callback:
se activa cuando se recibe una solicitud del cliente

```
esp_err_t stream_handler(httpd_req_t *req) {
   // GET received → acquire and send image

   camera_fb_t *fb=esp_camera_fb_get();
   frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
   esp_camera_fb_return(fb);

   httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
   httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
   httpd_resp_set_hdr(req, "X-Framerate", "60");

   httpd_resp_send_chunk(req, _STREAM_BOUNDARY, strlen(_STREAM_BOUNDARY));
   size_t hlen = snprintf((char *)part_buf, 128, _STREAM_PART, _jpg_buf_len,
                          _timestamp.tv_sec, _timestamp.tv_usec);
   httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
   httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);

   esp_camera_fb_return(fb);
}
```