

Aplicaciones con ESP32_Cam y TTGO_Camera

Problema al enviar foto con librería UniversalTelegramBot

<https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot/issues/68>

Problema de la librería, la solución es modificar:

En `UniversalTelegramBot.cpp` `UniversalTelegramBot :: sendMultipartFormDataToTelegram (..)` agregué después de cada `"cliente-> print" / "cliente-> println" / "cliente-> write"` un `"delay (50)"`.

No <https://www.survivingwithandroid.com/telegram-esp32-cam-send-image-arduino-ide/>

<https://randomnerdtutorials.com/esp32-esp8266-publish-sensor-readings-to-google-sheets/>

ESP32_Cam_sensor_fotoEmail

TTGO_Camera_sensor_fotoEmail

<https://docs.google.com/document/d/1fPTioYQup7pGxAULI0UmpHfhzbrj5JbaHAQMtx-oC0/edit#>

Manda Una foto por email a una cuenta de Gmail cuando se dispara sensor PIR, usando un servidor SMTP.

La última foto tomada se guarda temporalmente en la memoria SPIFFS de ESP32.

Componentes (se pueden adquirir en Amazon o más baratos en Aliexpress):

- Placa ESP32-Cam con camara (AI Thinker ESP32-Cam).
- Programador USB para ESP32-Cam.
- Sensor de proximidad PIR.
- Placa protoboard o diseñada con EasyEda.

Para enviar correos electrónicos con ESP32-CAM , usaremos la biblioteca ESP32 MailClient . Esta biblioteca permite que el ESP32 envíe y reciba correos electrónicos con o sin archivos adjuntos a través de servidores SMTP e IMAP.

Una vez creada la cuenta en Gmail si no existe, debemos permitir que las aplicaciones menos seguras accedan a esta cuenta de Gmail, para que pueda enviar correos electrónicos.

Lo podemos hacer desde el siguiente enlace: <https://myaccount.google.com/lesssecureapps?pli=1>

Configuración del servidor SMTP de Gmail

Estos son los valores para configurar la cuenta de gmail, en negritas los que utilizaremos.

SMTP Server: **smtp.gmail.com**
SMTP username: **dirección completa de Gmail**
SMTP password: **contraseña de Gmail**
SMTP port (TLS): 587
SMTP port (SSL): **465**
SMTP TLS/SSL required: **si**

Mas información:

Rui Santos <https://randomnerdtutorials.com/esp32-cam-send-photos-email/>

ESP32-control_telegram

Solución en viar foto

<https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot/issues/68>

Problema de la librería, la solución es modificar:

En `UniversalTelegramBot.cpp` `UniversalTelegramBot :: sendMultipartFormDataToTelegram (..)` agregué después de cada `"cliente-> print" / "cliente-> println" / "cliente-> write"` un `"delay (50)"`.

ESP32_cam_telegram_vigilancia

TTGO_telegram_vigilancia

Aplicación para control utilizando Telegram.

Componentes (se pueden adquirir en Amazon o más baratos en Aliexpress):

- Placa ESP32-Cam con camara (AI Thinker ESP32-Cam).
- Sensor de proximidad PIR.
- Placa protoboard o diseñada con EasyEda.

Crear Bot en Telegram:

Ver pagina oficial: <https://core.telegram.org/bots#6-botfather>

En resumen una vez descargada la APP de Telegram y creada nuestra cuenta:

- Buscar BotFather. Con **/start**, nos muestra los comandos utilizables.
- Mandar mensaje **/newbot**
- Nos pide nombre del Bot (ejemplo esp32Cam)
- Nos pide usuario del Bot, que debe terminar en bot (ejemplo **j1234_bot**).
- Nos crea el bot y no muestra el token para acceder mediante HTTP
Api:**12732xxxxx:xxxxxxxxxxW9pFjxxxxxxxxgx4** (que utilizaremos en el scrip)
- Para conocer el chatId buscar Bot **IDBot** y mandar mensaje **/getid**
- Añadimos el Bot creado **t.me/j1234_bot**

Librerías:

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "soc/soc.h"
```

```
#include "soc/rtc_cntl_reg.h"
#include "esp_camera.h"
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
```

Credenciales de red

Inserte sus credenciales de red en las siguientes variables.

```
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

ID de usuario de Telegram

Inserta tu ID de chat. El que tienes del IDBot.

```
String chatId = "XXXXXXXXXX";
```

Token de Telegram Bot

Inserte el token de Telegram Bot que obtuvo de Botfather en la variable

`BOTtoken`

```
String BOTtoken = "XXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
String mensaje_inicial ;
Crea un nuevo cliente WiFi con WiFiClienteSeguro.
WiFiClientSecure clientTCP;
Cree un bot con el token y el cliente definidos anteriormente.
UniversalTelegramBot bot(BOTtoken, clientTCP);
```

Las variables `botRequestDelay` y `lastTimeBotRan` se utilizan para comprobar si hay nuevos mensajes de Telegram cada x número de segundos. En este caso, el código buscará nuevos mensajes cada segundo (1000 milisegundos). Puede cambiar ese tiempo de retraso en el `botRequestDelay` variable.

```
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;
```

Inicialización ESP32-CAM

Las siguientes líneas asignan los pines ESP32-CAM AI-Thinker. Si está utilizando un modelo de cámara ESP32 diferente, no olvide cambiar el pinout (lea [Tableros de cámara ESP32-CAM: Guía de asignación de pines y GPIO](#)).

```
//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
```

los `configInitCamera ()` La función inicializa la cámara ESP32.

```
void configInitCamera(){
    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;
//init with high specs to pre-allocate larger buffers
if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10; //0-63 lower number means higher quality
    config.fb_count = 2;
} else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12; //0-63 lower number means higher quality
    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);
    delay(1000);
    ESP.restart();
}
// Drop down frame size for higher initial frame rate
sensor_t * s = esp_camera_sensor_get();
s->set_framesize(s, FRAMESIZE_CIF); //
UXGA|SXGA|XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA
}

```

Inicialización TTGO-CAM

```

void inicializaCamara(){
    // #define T_Camera_V162_VERSION

#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
#define Y5_GPIO_NUM       35
#define Y4_GPIO_NUM       14
#define Y3_GPIO_NUM       13

```

```

#define Y2_GPIO_NUM      34
#define VSYNC_GPIO_NUM   5
#define HREF_GPIO_NUM    27
#define PCLK_GPIO_NUM    25
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;
//init with high specs to pre-allocate larger buffers
if (psramFound()) {
    config.frame_size = FRAMESIZE_VGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
} else {
    config.frame_size = FRAMESIZE_VGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
}

// defined(T_Camera_V162_VERSION)
/* IO13, IO14 is designed for JTAG by default,
 * to use it as generalized input,
 * firstly declair it as pullup input */
pinMode(13, INPUT_PULLUP);
pinMode(14, INPUT_PULLUP);

// camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x\n", err);
//    return false;
}

sensor_t *s = esp_camera_sensor_get();

```

```

    //initial sensors are flipped vertically and colors are a bit saturated
    if (s->id.PID == OV3660_PID) {
        s->set_vflip(s, 1); //flip it back
        s->set_brightness(s, 1); //up the blightness just a bit
        s->set_saturation(s, -2); //lower the saturation
    }
    //drop down frame size for higher initial frame rate
    s->set_framesize(s, FRAMESIZE_VGA); //FRAMESIZE_QVGA

//  defined(T_Camera_V162_VERSION)
    s->set_vflip(s, 1);
    s->set_hmirror(s, 1);

//    return true;
}

```

handleNewMessages ()

los `handleNewMessages ()` La función maneja lo que sucede cuando llegan nuevos mensajes.

```

void handleNewMessages(int numNewMessages) {

    Serial.print("Handle New Messages: ");
    Serial.println(numNewMessages);

    for (int i = 0; i < numNewMessages; i++) {
        String chat_id = String(bot.messages[i].chat_id);

        String text = bot.messages[i].text;
        Serial.println(text);

        String from_name = bot.messages[i].from_name;

        if (text.equalsIgnoreCase("/opciones")){
            bot.sendMessage(chatId, mensaje_inicial , "Markdown");
        }

        else if (text.equalsIgnoreCase("/foto")){
            bot.sendMessage(chatId, "sendPhotoTelegram", "Markdown");
            enviarFotoTelegram();
        }

        else {
            bot.sendMessage(chatId, text + ": Comando no reconocido", "");
        }
    }
}

```


Enviar un mensaje al bot es muy sencillo. Solo necesitas usar el `sendMessage()` en el objeto bot y pasar como argumentos el ID de chat del destinatario, el mensaje y el modo de análisis.

```
bool sendMessage(String chat_id, String text, String parse_mode = "");
```

En nuestro ejemplo, enviaremos el mensaje a la ID almacenada en el `chatId` variable (que corresponde a su ID de chat personal) y envíe el mensaje guardado en la variable de bienvenida.

```
bot.sendMessage(chatId, "sendPhotoTelegram", "Markdown");
```

Si recibe el mensaje / **foto** , llama a la función `enviarFotoTelegram()` , que toma foto y la envia a Telegram

```
if (text.equalsIgnoreCase("/foto")){  
    bot.sendMessage(chatId, "Enviar Foto a Telegram", "");  
    enviarFotoTelegram();  
}
```

enviarFotoTelegram ()

La función toma una foto con la camara y la envia a Telegram

```
String enviarFotoTelegram() {  
  
    //Toma foto:  
  
    const char* myDomain = "api.telegram.org";  
    String getAll = "";  
    String getBody = "";  
  
    fb = esp_camera_fb_get();  
    if(!fb) {  
        Serial.println("Captura de camara fallida");  
        delay(1000);  
        ESP.restart();  
        return "Captura de camara fallida";  
    }  
  
    Serial.println("Conectado a " + String(myDomain));  
  
    // Envia foto:
```

```
if (clientTCP.connect(myDomain, 443)) {  
    Serial.println("Conectado");
```

```
    String head = "--Jarp\r\nContent-Disposition: form-data; name=\"chat_id\";\r\n\r\n" + chatId + "\r\n--Jarp\r\nContent-Disposition: form-data;\r\nname=\"photo\"; filename=\"TTGO-cam.jpg\"\r\nContent-Type: image/jpeg\r\n\r\n";  
    String tail = "\r\n--Jarp--\r\n";
```

```
    uint16_t imageLen = fb->len;  
    uint16_t extraLen = head.length() + tail.length();  
    uint16_t totalLen = imageLen + extraLen;
```

```
    clientTCP.println("POST /bot"+BOTtoken+"/sendPhoto HTTP/1.1");  
    clientTCP.println("Host: " + String(myDomain));  
    clientTCP.println("Content-Length: " + String(totalLen));  
    clientTCP.println("Content-Type: multipart/form-data; boundary=Jarp");  
    clientTCP.println();  
    clientTCP.print(head);
```

```
    uint8_t *fbBuf = fb->buf;  
    size_t fbLen = fb->len;  
    for (size_t n=0;n<fbLen;n=n+1024) {  
        if (n+1024<fbLen) {  
            clientTCP.write(fbBuf, 1024);  
            delay(50); //AÑADIDO  
            fbBuf += 1024;  
        }  
        else if (fbLen%1024>0) {  
            size_t remainder = fbLen%1024;  
            clientTCP.write(fbBuf, remainder);  
            delay(50); //AÑADIDO  
        }  
    }  
}
```

```
    clientTCP.print(tail);
```

```
    esp_camera_fb_return(fb);
```

```
    int waitTime = 10000;    // 10s  
    long startTimer = millis();  
    boolean state = false;
```

```
    while ((startTimer + waitTime) > millis()){  
        Serial.print(".");  
        delay(100);  
        while (clientTCP.available()) {  
            char c = clientTCP.read();  
            if (c == '\n') {  
                if (getAll.length()==0) state=true;  
                getAll = "";
```

```

    }
    else if (c != '\r')
        getAll += String(c);
    if (state==true) getBody += String(c);
    startTimer = millis();
    }
    if (getBody.length()>0) break;
}
clientTCP.stop();
Serial.println(getBody);

}
else {
    getBody="Error al conectar a api.telegram.org.";
    Serial.println(getBody);
}
bot.sendMessage(chatId, getBody, "");
return getBody;
}

```

setup()

```

WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
// Inicialice Serial Monitor.
Serial.begin(115200);

// Conecta a WiFi
WiFi.mode(WIFI_STA);
Serial.println();
Serial.print("Conectado a WIFI: ");
Serial.println(ssid);
WiFi.begin(ssid, password);

// Añade certificado para api.telegram.org
clientTCP.setCACert(TELEGRAM_CERTIFICATE_ROOT);
while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
}
Serial.println();
Serial.print("TTGO-CAM IP Address: ");
Serial.println(WiFi.localIP());
IPAddress ipAddress = WiFi.localIP();
//convierte Ip a cadena
String miip =String(ipAddress[0]) + String(".") + String(ipAddress[1]) +
String(".") + String(ipAddress[2]) + String(".") + String(ipAddress[3]) ;

```

```

inicializaCamara();

//Inicializa sensor PIR
esp_err_t err = gpio_isr_handler_add(GPIO_NUM_19, &detectsMovement, (void
*) AS312_PIN);
if (err != ESP_OK){
    Serial.printf("handler NO  añadido con error 0x%x \r\n", err);
}
err = gpio_set_intr_type(GPIO_NUM_19, GPIO_INTR_POSEDGE);
if (err != ESP_OK){
    Serial.printf("Fallo al establecer el tipo de intr  con error 0x%x \r\n",
err);
}

//Envia mensaje Inicial
mensaje_inicial = "TTGO_telegram_vigilancia conectada a IP:" + miip;
mensaje_inicial += "\n\nopciones : comandos \n\n";
mensaje_inicial += "/foto : toma una nueva foto\n\n";
mensaje_inicial += "Recibirás una foto cada vez que se detecte
movimiento.\n";
Serial.println("TTGO_telegram_vigilancia conectada");
bot.sendMessage(chatId, mensaje_inicial , "Markdown");

```

loop()

Comprueba si se ha disparado el sensor, mediante interrupción, y llama a `enviarFotoTelegram()` para capturar foto y mandar a telegram.

```

if(disparo){
    Serial.println("Movimiento detectado");
    bot.sendMessage(chatId, "Movimiento detectado" , "Markdown");
    enviarFotoTelegram();
    bot.sendMessage(chatId, "Foto Enviada detector" , "Markdown");
    disparo= false;
}

```

También comprueba si hay mensajes nuevos cada segundo.

```

if (millis() > lastTimeBotRan + botRequestDelay) {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    while (numNewMessages) {
        Serial.println("got response");
        handleNewMessages(numNewMessages);
        numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
}

```

```
}  
lastTimeBotRan = millis();  
}
```

Cuando llegue un mensaje nuevo, llame al `handleNewMessages ()` función.

```
while (numNewMessages) {  
    Serial.println("got response");  
    handleNewMessages(numNewMessages);  
    numNewMessages = bot.getUpdates(bot.last_message_received + 1);  
}
```