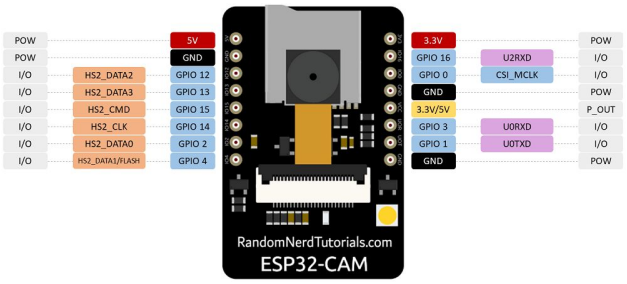
ESP32 manda foto por Gmail

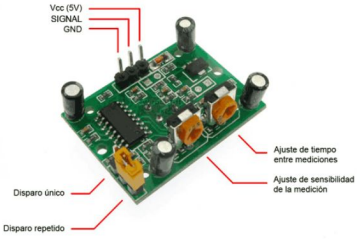
Manda Una foto por email a una cuenta de Gmail cuando se dispara el sensor PIR, usando un servidor SMTP.
La última foto tomada se guarda temporalmente en la memoria SPIFFS de ESP32.

ESP32_Cam_sensor_fotoEmail

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

- Placa ESP32-Cam con camara (Al 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**

Cómo funciona el código:

Bibliotecas utilizadas

```
#include "esp_camera.h"
#include "SPI.h"
#include "driver/rtc_io.h"
#include "ESP32_MailClient.h"
#include <FS.h>
#include <SPIFFS.h>
#include <WiFi.h>
```

Valores de WiFI

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

Ajustes del correo electrónico

```
#define emailSenderAccount   "EXAMPLE_SENDER_ACCOUNT@gmail.com"
#define emailSenderPassword  "SENDER_ACCOUNT_PASSWORD"
#define emailRecipient "YOUR_EMAIL_RECIPIENT@gmail.com"

#define smtpServer "smtp.gmail.com"
#define smtpServerPort 465

//Variable que almacena el asunto del email.

#define emailSubject "ESP32-CAM Foto enviada por disparo sensor"
```

// Disparo Sensor

```
bool motionDetected = false;
```

// Indica cuando se dispara el sensor

```
static void IRAM_ATTR detectsMovement(void * arg){
  motionDetected = true;
  Serial.println("Disparo sensor");
}
```

//Crear un STMPData objeto llamado smtpData que contiene los datos y la configuración del correo electrónico a enviar.

```
SMTPData smtpData;
```

//Guardamos la foto tomada por la cámara ESP32 en SPIFFS con el nombre foto.jpg.

```
define FILE_PHOTO "/foto.jpg"
```

```
//Definimos los pines de cámara a utilizar  AI-Thinker ESP32-CAM

#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
```

En setup()

```
//Conecta el ESP32 a Wi-Fi.
WiFi.begin(ssid, password);
Serial.print("Connecting to WiFi...");
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println();

//Inicializa la memoria SPIFFS para guardar la foto tomada con la ESP32-CAM.
if (!SPIFFS.begin(true)) {
  Serial.println("An Error has occurred while mounting SPIFFS");
  ESP.restart();
}
else {
  delay(500);
  Serial.println("SPIFFS mounted successfully");
}

// Muestar  la dirección IP de ESP32-CAM:
Serial.print("IP Address: http://");
Serial.println(WiFi.localIP());

//Configura y establecen los ajustes de la cámara:
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;

if(psramFound()){
  config.frame_size = FRAMESIZE_UXGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}

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

// PIR Modo de sensor de movimiento INPUT_PULLUP
    err = gpio_isr_handler_add(GPIO_NUM_13, &detectsMovement, (void *) 13);
  if (err != ESP_OK){
    Serial.printf("handler NO  añadido con error 0x%x \r\n", err);
  }
  err = gpio_set_intr_type(GPIO_NUM_13, GPIO_INTR_POSEDGE);
  if (err != ESP_OK){
    Serial.printf("Fallo al establecer el tipo de intr  con error 0x%x \r\n", err);
  }
```

En loop()

```
// Comprueba si se ha disparado el sensor llama a capturar foto y envia foto
  if(motionDetected){
    Serial.println("Sensor disparado");
       capturaFotoGuardaSpiffs();
      enviaFoto();
    motionDetected = false;
  }
```

función capturaFotoGuardaSpiffs ()

```
//La función captura una foto y la guarda en la memoria SPIFFS de ESP32_Cam.
fb = esp_camera_fb_get();
if (!fb) {
  Serial.println("Camera capture failed");
  return;
}

//Crea un nuevo archivo en SPIFFS donde se almacena la foto.

Serial.printf("Picture file name: %s\n", FILE_PHOTO);
File file = SPIFFS.open(FILE_PHOTO, FILE_WRITE);


//Comprueba si el archivo se creó correctamente. Si no es así, muestra un mensaje de error.

if (!file) {
  Serial.println("Falló la captura de la cámara");
}

//Una vez creado archivo sin error, guarda la imagen dell búfer al archivo.
 File file = SPIFFS.open(FILE_PHOTO, FILE_WRITE);

file.write(fb->buf, fb->len); // payload (image), payload length
Serial.print("The picture has been saved in ");
Serial.print(FILE_PHOTO);
Serial.print(" - Size: ");
Serial.print(file.size());
Serial.println(" bytes");


//Cierra el archivo y borra el búfer utilizado.
file.close();
esp_camera_fb_return(fb);
```

//Verifica si la foto se tomó y guardó correctamente. Lo hace comprobando que el tamaño del archivo de la fotosea mayor de 100bytes. para esto utilizamos la función **checkFoto** () .

```
ok = checkFoto(SPIFFS);


bool checkPhoto( fs::FS &fs ) {
  File f_pic = fs.open( FILE_PHOTO );
  unsigned int pic_sz = f_pic.size();
  return ( pic_sz > 100 );
}
```

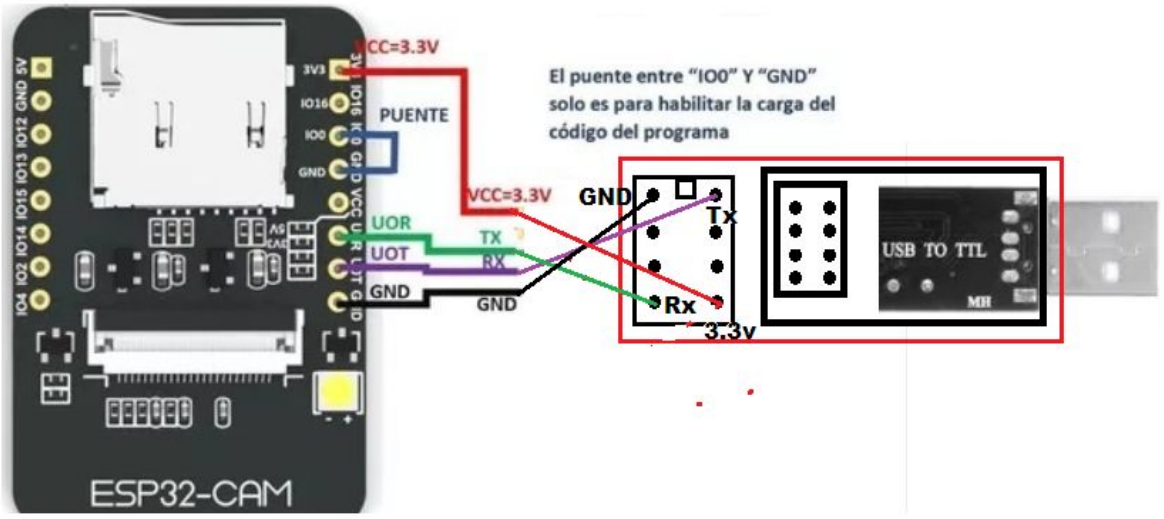
En caso de que el tamaño de la foto sea inferior a 100 bytes, la variable **OK** seguirá siendo 0 y seguirá intentando tomar una nueva foto y guardarla.

// Envia la foto tomada por email

```
void sendPhoto( void ) {

//Configura el host, el puerto, la cuenta y la contraseña del correo electrónico del servidor SMTP
smtpData.setLogin(smtpServer, smtpServerPort, emailSenderAccount, emailSenderPassword);

// Establece  el nombre del remitente y el correo electrónico
  smtpData.setSender("ESP32-CAM", emailSenderAccount);

// Establece la prioridad o importancia del correo electrónico Alta, Normal, Baja o de 1 a 5 (1 es la más alta)
  smtpData.setPriority("High");

 // Establece el asunto del email
  smtpData.setSubject(emailSubject);

  // Establece el mensaje de correo electrónico en formato de texto (con true formato HTML)
  smtpData.setMessage("Sensor disparado, foto capturada with ESP32-CAM.", false);

  // Agregar destinatarios, puede agregar más de un destinatario
  smtpData.addRecipient(emailRecipient);
  smtpData.addRecipient("xxxxxxx@gmail.com");

  // Agregar archivos adjuntos de SPIFFS
  smtpData.addAttachFile(FILE_PHOTO, "image/jpg");
```

```
  // Configura el tipo de almacenamiento para adjuntar archivos en su correo electrónico(SPIFFS)
  smtpData.setFileStorageType(MailClientStorageType::SPIFFS);

// Función sendCallback de devolución de llamada para obtener el estado de envío de correo electrónico
  smtpData.setSendCallback(sendCallback);
   // Comienza a enviar el correo electrónico, se puede configurar la función de devolución de llamada para rastrear el estado
  if (!MailClient.sendMail(smtpData))
    Serial.println("Error sending Email, " + MailClient.smtpErrorReason());

  // Borra todos los datos del objeto de correo electrónico para liberar memoria
  smtpData.empty();
```

Para programar desde el entorno de arduino configuramos la placa como AI Thinker ESP32-CAM



Codigo completo:

ESP32_Cam_sensor_fotoEmail.ino

```
/*********
 * ESP32_Cam_sensor_fotoEmail
*  Manda foto por Gmail cuando se dispara sensor PIR
* jarp 2020

 Basado en  Rui Santos   https://randomnerdtutorials.com/esp32-cam-send-photos-email/

 *********/

#include "esp_camera.h"
#include "SPI.h"
#include "driver/rtc_io.h"
#include "ESP32_MailClient.h"
#include <FS.h>
#include <SPIFFS.h>
#include <WiFi.h>
```

```
// Datos wifi
const char* ssid = "xxxxxx";   // Actualizar datos
const char* password = xxxxxxx";  // Actualizar datos

// Configuración para Gmail
// DEBE HABILITAR la opción de aplicación menos segura https://myaccount.google.com/lesssecureapps?pli=1
#define emailSenderAccount    "xxxxxxx@gmail.com"   // Actualizar datos
#define emailSenderPassword   "xxxxxxx"                  // Actualizar datos
#define smtpServer            "smtp.gmail.com"
#define smtpServerPort         465
#define emailSubject          "ESP32-CAM Disparo sensor, envia foto"
#define emailRecipient        "xxxxxx@gmail.com"         // Actualizar datos

#define CAMERA_MODEL_AI_THINKER

#if defined(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
#else
  #error "Modelo de cámara no seleccionado"
#endif

// Disparo  Sensor
bool motionDetected = false;

// Indica cuando se dispara el sensor
static void IRAM_ATTR detectsMovement(void * arg){
  motionDetected = true;
  Serial.println("Disparo sensor");
}

// El objeto de datos de envío de correo electrónico contiene la configuración y los datos para enviar
SMTPData smtpData;

// Nombre de archivo de foto para guardar SPIFFS
#define FILE_PHOTO "/foto.jpg"

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); // deshabilitar el detector de fallo de alimentación

  Serial.begin(115200);
  Serial.println();

  // Conecta Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Conectando a WiFi ...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  if (!SPIFFS.begin(true)) {
```

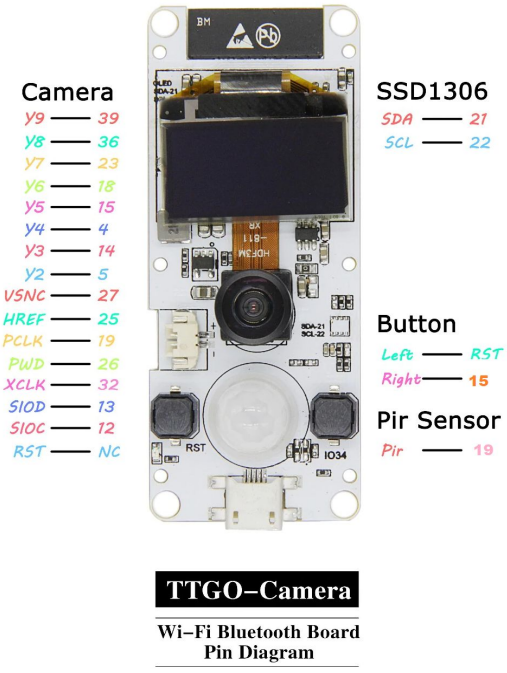
```
    Serial.println("Ha ocurrido un error durante el montaje SPIFFS");
    ESP.restart();
  }
  else {
    delay(500);
    Serial.println("SPIFFS montado con éxito");
  }

  // Muestra dirección IP Local de ESP32_Cam
  Serial.print("dirección IP: http://");
  Serial.println(WiFi.localIP());

  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;

  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

  // Inicializa camara
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Error de inicio de cámara con error 0x%x", err);
    return;
  }
    // PIR Modo de sensor de movimiento INPUT_PULLUP
     err = gpio_isr_handler_add(GPIO_NUM_13, &detectsMovement, (void *) 13);
  if (err != ESP_OK){
    Serial.printf("handler NO  añadido con error 0x%x \r\n", err);
  }
  err = gpio_set_intr_type(GPIO_NUM_13, GPIO_INTR_POSEDGE);
  if (err != ESP_OK){
    Serial.printf("Fallo al establecer el tipo de intr  con error 0x%x \r\n", err);
  }

}

void loop() {

   if(motionDetected){
    Serial.println("Sensor disparado");
   capturaFotoGuardaSpiffs();
   enviaFoto();
```

```
    motionDetected = false;
  }

}

// Verifica si la captura de fotos fue exitosa
bool checkFoto( fs::FS &fs ) {
  File f_pic = fs.open( FILE_PHOTO );
  unsigned int pic_sz = f_pic.size();
  return ( pic_sz > 100 );
}


// Capturar foto y guardarla en memoria SPIFFS
void capturaFotoGuardaSpiffs;( void ) {
  camera_fb_t * fb = NULL; // pointer
  bool ok = 0; // Boolean indicando si la foto se ha tomado correctamente

  do {
    // Toma una foto con la cámara
    Serial.println("Tomando una foto...");

    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Falló la captura de la cámara");
      return;
    }

    // Nombre de archivo de foto
    Serial.printf("Nombre del archivo de imagen: %s\n", FILE_PHOTO);
    File file = SPIFFS.open(FILE_PHOTO, FILE_WRITE);

    // Inserta los datos de la foto en el archivo .
    if (!file) {
      Serial.println("No se pudo abrir el archivo en modo de escritura");
    }
    else {
      file.write(fb->buf, fb->len); // payload (imagen), payload length
      Serial.print("La imagen se ha guardado en ");
      Serial.print(FILE_PHOTO);
      Serial.print(" - Tamaño: ");
      Serial.print(file.size());
      Serial.println(" bytes");
    }
    // Cerrar el archivo
    file.close();
    esp_camera_fb_return(fb);

    // comprueba si el archivo se ha guardado correctamente en SPIFFS
    ok = checkFoto(SPIFFS);
  } while ( !ok );
}

void  enviaFoto( void ) {
  // Preparando email
  Serial.println("Enviando email...");

  //Configura el host, el puerto, la cuenta y la contraseña del correo electrónico del servidor SMTP
  smtpData.setLogin(smtpServer, smtpServerPort, emailSenderAccount, emailSenderPassword);

  // Establece  el nombre del remitente y el correo electrónico
  smtpData.setSender("ESP32-CAM", emailSenderAccount);

  // Establece la prioridad o importancia del correo electrónico Alta, Normal, Baja o de 1 a 5 (1 es la más alta)
  smtpData.setPriority("High");

  //  Establece el asunto del email
  smtpData.setSubject(emailSubject);
```

```
  //  Establece el mensaje de correo electrónico en formato HTML
//  smtpData.setMessage("<h2>Sensor  ESP32-CAM and attached in this email.</h2>", true);
  //  Establece el mensaje de correo electrónico en formato de texto
  smtpData.setMessage("Sensor disparado, foto capturada with ESP32-CAM.", false);

  // Agregar destinatarios, puede agregar más de un destinatario
  smtpData.addRecipient(emailRecipient);
  smtpData.addRecipient("xxxxxxxx@gmail.com");    // Actualizar datos

  // Agregar archivos adjuntos de SPIFFS
  smtpData.addAttachFile(FILE_PHOTO, "image/jpg");

  // Configura el tipo de almacenamiento para adjuntar archivos en su correo electrónico(SPIFFS)
  smtpData.setFileStorageType(MailClientStorageType::SPIFFS);
  smtpData.setSendCallback(sendCallback);

  // Comienza a enviar el correo electrónico
  if (!MailClient.sendMail(smtpData))
    Serial.println("Error sending Email, " + MailClient.smtpErrorReason());

  // Borra todos los datos del objeto de correo electrónico para liberar memoria
  smtpData.empty();
}

// Función de devolución de llamada para obtener el estado de envío de correo electrónico
void sendCallback(SendStatus msg) {
  //Imprime el estado actual
  Serial.println(msg.info());
}
```

TTGO_sensor_fotoEmail

Podemos utlilizar la placa TTGO- Camera, que ya tiene incluido la camara, el sensor PIR y un botón entre otras cosa.



Según la versión de la placa y la camara incluida asignamos los pines de la camara, del sensor y del botón, ver:

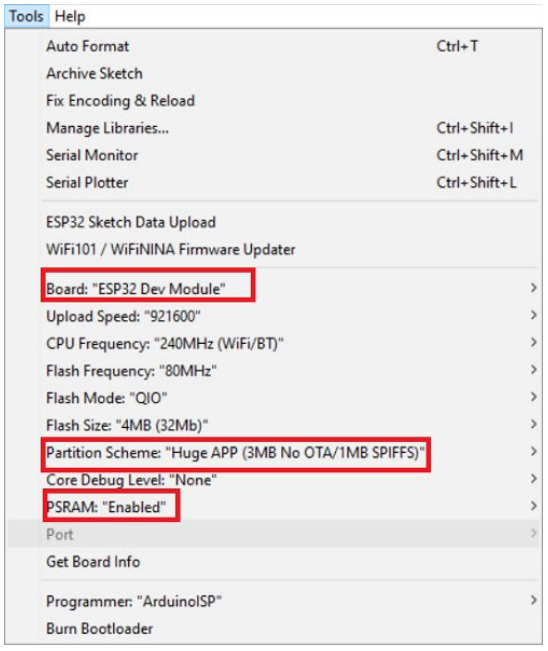
https://github.com/Xinyuan-LilyGO/LilyGo-Camera-Series

Para la versión utilizada T_Camera_V162_VERSION , los pines son:

```
#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

//LCD
#define I2C_SDA             21
#define I2C_SCL             22

//Botón
#define BUTTON_1            15

 // LCD
#define SSD130_MODLE_TYPE   0   // LCD 0 : GEOMETRY_128_64  // 1: GEOMETRY_128_32

//PIR sensor
#define AS312_PIN           19

//MIC
#define IIS_SCK             26
#define IIS_WS              32
#define IIS_DOUT            33

#define ENABLE_IP5306
```

Para cargar el scrip con el entorno de arduino, configuramos la placa:



Codigo completo:
TTGO_sensor_fotoEmail.ino

```
/*********
 * TTGO_sensor_fotoEmail
 * Manda foto por Gmail cuando se dispara sensor PIR
 * * jarp 2020
 *
```

```
 * Información pines: https://github.com/Xinyuan-LilyGO/LilyGo-Camera-Series

 *********/

#include "esp_camera.h"
#include "driver/rtc_io.h"
#include "soc/soc.h"           //Solución de error de caída
#include "soc/rtc_cntl_reg.h"  // Solución de error de caída
#include "ESP32_MailClient.h"
#include <FS.h>
#include <SPIFFS.h>
#include <WiFi.h>

// Datos wifi
const char* ssid = "xxxxxxx";
const char* password = "xxxxxxx";

// Configuración para Gmail
// DEBE HABILITAR la opción de aplicación menos segura https://myaccount.google.com/lesssecureapps?pli=1
#define emailSenderAccount    "xxxxxx@gmail.com"  // Actualizar datos
#define emailSenderPassword   "xxxxxxx"  // Actualizar datos
#define smtpServer            "smtp.gmail.com"
#define smtpServerPort         465
#define emailSubject          "ESP32-TTGO  Disparo sensor, envia foto"
#define emailRecipient        "xxxxxxx@gmail.com"  // Actualizar datos

//  T_Camera_V162

#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

// Disparo  Sensor

volatile bool disparo = false; //volatile para que el compilador no la borre?

static void IRAM_ATTR detectsMovement(void * arg){
  disparo = true;
  Serial.println("Disparo sensor");
}


bool inicializaCamara(){

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

    pinMode(13, INPUT_PULLUP);
    pinMode(14, INPUT_PULLUP);


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

    sensor_t *s = esp_camera_sensor_get();
    //tamaño de fotograma desplegable para una mayor velocidad de fotogramas inicial
    s->set_framesize(s, FRAMESIZE_VGA);

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

  return true;
}



// El objeto de datos de envío de correo electrónico contiene la configuración y los datos para enviar
SMTPData smtpData;

const int buttonPin = 15;
// Nombre de archivo de foto para guardar SPIFFS
#define FILE_PHOTO "/fotoemail.jpg"

void setup() {
//  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); // deshabilitar el detector de fallo de alimentación

  Serial.begin(115200);
  Serial.println();

  // Conecta Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Conectando a WiFi ...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
```

```
  if (!SPIFFS.begin(true)) {
    Serial.println("Ha ocurrido un error durante el montaje SPIFFS");
    ESP.restart();
  }
  else {
    delay(500);
    Serial.println("SPIFFS montado con éxito");
  }
  SPIFFS.format();
  // Muestra dirección IP Local de ESP32
  Serial.print("dirección IP: http://");
  Serial.println(WiFi.localIP());

   inicializaCamara();
//digitalPinToInterrupt(PIRPin);
//  pinMode(PIRPin, INPUT);//_PULLUP
//  attachInterrupt(PIRPin, int_disparo,  FALLING); //RISING 0-1, FALLING 1-0,
    // PIR Modo de sensor de movimiento INPUT_PULLUP
   esp_err_t  err = gpio_isr_handler_add(GPIO_NUM_19, &detectsMovement, (void *) 19);
  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);
  }
   pinMode(buttonPin, INPUT);

    SPIFFS.remove(FILE_PHOTO);// borra foto anterior
  Serial.println("TTGO_sensor_fotoEmail");

}

void loop() {

   if( disparo){
    Serial.println("Sensor disparado");
      capturaFotoGuardaSpiffs();
      enviaFoto();
    disparo = false;
  }
      if (digitalRead(buttonPin) == LOW) {
         Serial.println("Boton disparado");
      capturaFotoGuardaSpiffs();
      enviaFoto();
      }
}

// Verifica si la captura de fotos fue exitosa
bool checkFoto( fs::FS &fs ) {
  File f_pic = fs.open( FILE_PHOTO );
  unsigned int pic_sz = f_pic.size();
  return ( pic_sz > 100 );
}

// Capturar foto y guardarla en memoria SPIFFS
void  capturaFotoGuardaSpiffs() {
  camera_fb_t * fb = NULL; // pointer
  bool ok = 0; // Boolean indicando si la foto se ha tomado correctamente

  do {
    // Toma una foto con la cámara
    Serial.println("Tomando una foto...");

    fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Falló la captura de la cámara");
      return;
```

```
    }

    // Nombre de archivo de foto
    Serial.printf("Nombre del archivo de imagen: %s\n", FILE_PHOTO);
    File file = SPIFFS.open(FILE_PHOTO, FILE_WRITE);

    // Inserta los datos de la foto en el archivo .
    if (!file) {
      Serial.println("No se pudo abrir el archivo en modo de escritura");
    }
    else {
      file.write(fb->buf, fb->len); // payload (imagen), payload length
      Serial.print("La imagen se ha guardado en ");
      Serial.print(FILE_PHOTO);
      Serial.print(" - Tamaño: ");
      Serial.print(file.size());
      Serial.println(" bytes");
    }
    // Cerrar el archivo
    file.close();
    esp_camera_fb_return(fb);

    // comprueba si el archivo se ha guardado correctamente en SPIFFS
    ok = checkFoto(SPIFFS);
  } while ( !ok );
}

void enviaFoto() {
  // Preparando email
  Serial.println("Enviando email...");
  //Configura el host, el puerto, la cuenta y la contraseña del correo electrónico del servidor SMTP
  smtpData.setLogin(smtpServer, smtpServerPort, emailSenderAccount, emailSenderPassword);

  // Establecer el nombre del remitente y el correo electrónico
  smtpData.setSender("ESP32-TTGO", emailSenderAccount);

  // Establezca la prioridad o importancia del correo electrónico Alta, Normal, Baja o de 1 a 5 (1 es la más alta)
  smtpData.setPriority("High");

  // Establecer el asunto del email
  smtpData.setSubject(emailSubject);

  // Establecer el mensaje de correo electrónico en formato HTML
//  smtpData.setMessage("<h2>Sensor  ESP32-CAM and attached in this email.</h2>", true);
  // Establecer el mensaje de correo electrónico en formato de texto
  smtpData.setMessage("Sensor disparado, foto capturada with ESP32-TTGO.", false);

  // Agregar destinatarios, puede agregar más de un destinatario
  smtpData.addRecipient(emailRecipient);
  smtpData.addRecipient("xxxxxxx@gmail.com");  // Actualizar datos

  // Agregar archivos adjuntos de SPIFFS
  smtpData.addAttachFile(FILE_PHOTO, "image/jpg");
  // Configura el tipo de almacenamiento para adjuntar archivos en su correo electrónico(SPIFFS)
  smtpData.setFileStorageType(MailClientStorageType::SPIFFS);

  smtpData.setSendCallback(sendCallback);

  // Comienza a enviar el correo electrónico, se puede configurar la función de devolución de llamada para rastrear el
estado
  if (!MailClient.sendMail(smtpData))
    Serial.println("Error sending Email, " + MailClient.smtpErrorReason());

  // Borra todos los datos del objeto de correo electrónico para liberar memoria
  smtpData.empty();
}

// Función de devolución de llamada para obtener el estado de envío de correo electrónico
```

```
void sendCallback(SendStatus msg) {
  //Imprime el estado actual
  Serial.println(msg.info());
}
```

Información:

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

biblioteca https://github.com/mobizt/ESP32-Mail-Client

Información pines TTGO : https://github.com/Xinyuan-LilyGO/LilyGo-Camera-Series

Ejemplos TTGO
https://makeradvisor.com/esp32-ttgo-t-camera-pir-sensor-oled/
https://github.com/Xinyuan-LilyGO/TTGO_Camera_Mini