ESP32-CAM

Dr. Rubén Estrada Marmolejo

Fecha: 1 de Octubre del 2021 v1.

correo: ruben.estrada@hetpro.com.mx

Contenido

Descripción general

El **ESP32-CAM** es una tarjeta de desarrollo que incluye un procesador de 32-bits corriendo a 160 Mhz y 600 DMIPS. Incluye una capacidad de memoria SRAM de 520 KB y una memoria externa de 4M PSRAM. Incluye una cámara OV2640 que puede suministrar una imagen UXGA de hasta 1600x1200px (se tiene que configurar). Incluso también se cuenta con un puerto WiFi y Bluetooth. La tarjeta ESP-32-CAM se puede programar en Arduino y su velocidad Serial BAUD por default es de 115200. Para poder ser programada se requiere configurar el IDE de Arduino así como contar con un convertidor USB-Serial externo. En los siguientes tutoriales, te indicaré el proceso para utilizar la ESP32-CAM.

Especificaciones

- Diseño esquemático.
- Dos Leds para el usuario:
    - LED Flash. Conectado en el pin GPIO-4.
    - Led conectado en el GPIO-32.
    - Cámara OV2640.
    - Conector uSD.
    - Botón de reset.
    - Memoria externa PSRAM 4M.
    - Conexión para programador Serial (U0TXD, U0RXD).
    - Voltaje de alimentación: 5vdc.

Configurar el IDE de Arduino para la ESP-32 CAM

1. Abrir el menú Archivo -> Preferencias.
2. Agregar los siguientes repositorios en el menú de “Gestor de URLs Adicionales de Tarjetas”.
    a. Escribir los siguientes repositorios separados por comas y seleccionar el botón “Ok”.
    b. https://dl.espressif.com/dl/package_esp32_index.json, http://arduino.esp8266.com/stable/package_esp8266com_index.json

Paso # 1.

Para configurar la tarjeta ESP-32 Cam en el IDE de Arduino, se requiere agregar las siguientes URL en la configuración de Arduino:

https://dl.espressif.com/dl/package_esp32_index.json,
http://arduino.esp8266.com/stable/package_esp8266com_index.json

Estas direcciones URL son los repositorios donde se encuentran el conjunto de librerías y elementos necesarios para realizar la configuración de Arduino para la ESP-32. Estas las vamos a escribir abriendo el menú “File” y “Preferences” o Archivo y preferencias según el idioma instalado. En la Figura 2 y 3, se muestran dichos menús, así como el apartado donde se coloca el texto anterior. Estos valores es importante separarlos mediante una coma. Una vez ingresado el texto se procede a seleccionar el botón de “ok”.

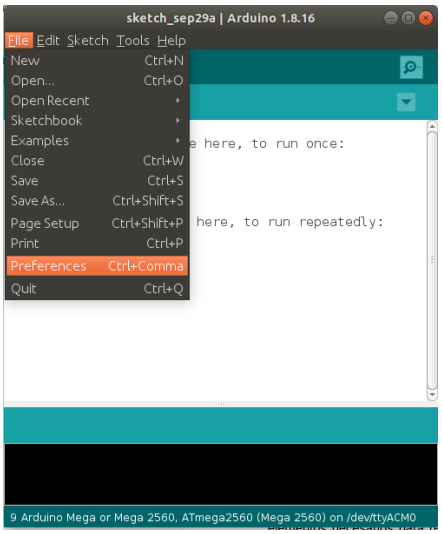

Figura 2. El menú preferencias del IDE de Arduino donde se podrá configurar la ESP32.

Paso # 2.

Una vez agregado los repositorios al IDE de Arduino, se procede a abrir el menú “tools” o herramientas para buscar la opción de “boards” o tarjetas y el menú “boards manager” o administrador de tarjetas. En la Figura 4, se muestra esta opción. A continuación se abrirá una ventana donde aparecerá una barra de exploración donde podremos buscar el texto: “ESP32”, una vez seleccionada la opción que aparece, solo seleccionamos el botón de instalar y al terminar cerramos dicha ventana.

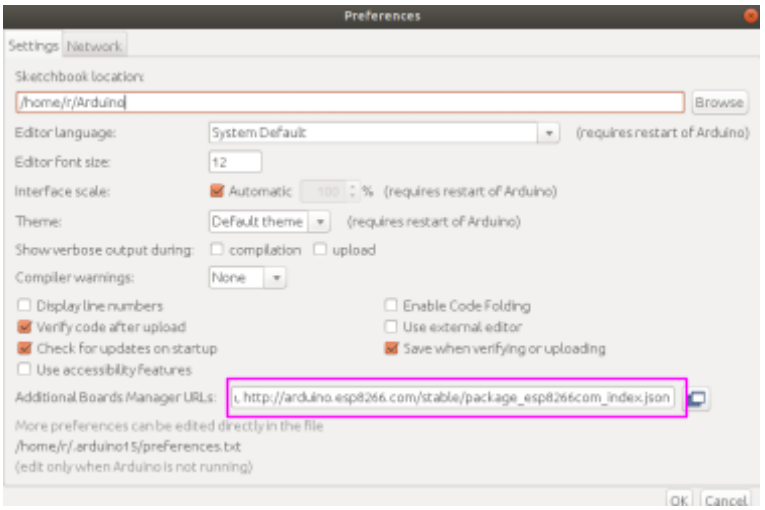

Figura 3. Ventana donde se debe de configurar las URL de los repositorios para la ESP-32-CAM.

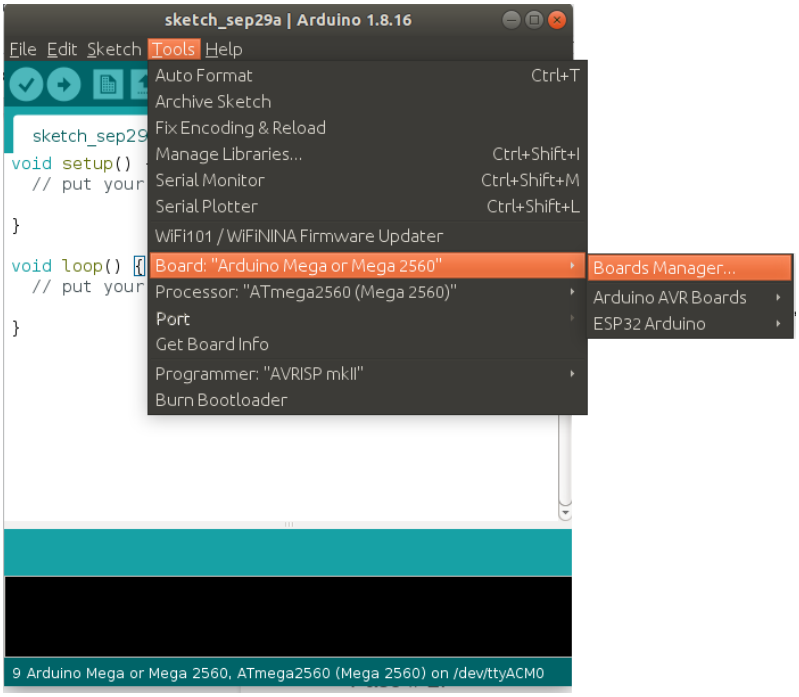

Figura 4. Paso · 2 en la configuración de la ESP32-CAM y el IDE de Arduino 1.8.16

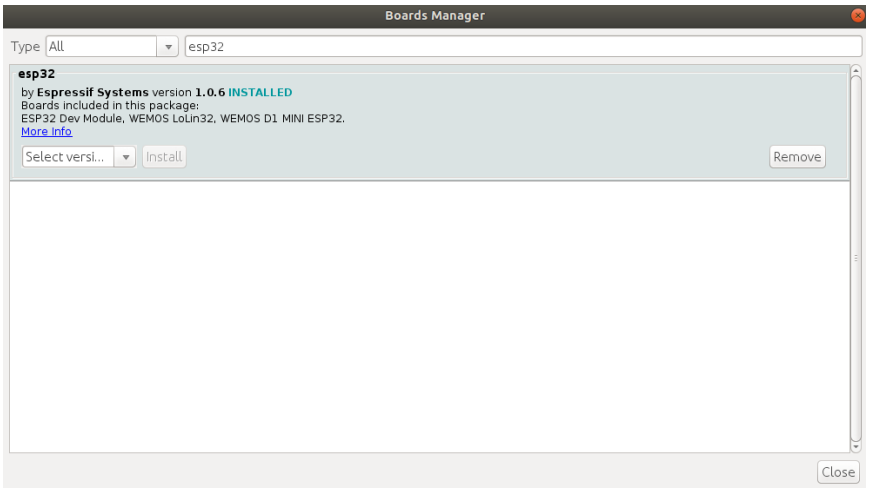

Figura 5. Instalación de la tarjeta esp32 en el IDE de Arduino y su administrador de tarjetas.

Programación de la tarjeta ESP-32 CAM

Para programar la ESP32-CAM se requiere de tener instalado el software de Arduino así como el paquete de la esp32. A diferencia de la mayoría de las tarjetas Arduino, esta tarjeta tiene un programador externo. Existen dos formas de programarla:

- Convertidor USB a Serial.
- Programador para ESP32-CAM.

En la Figura 6 y Figura 7 se muestran una imagen de las dos opciones de programador para la ESP32-CAM.

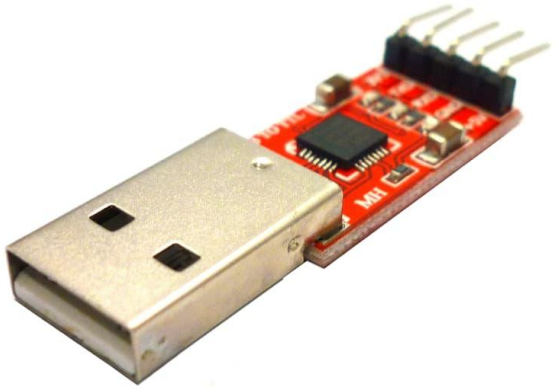

Figura 6. Convertidor USB-Serial CP2102.

Figura 7. Programador ESP32-CAM-MB.

Conexión programador con la ESP32-CAM

Las conexiones para la ESP32-CAM y un programador o convertidor USB a serial, solo son 5 y se realizan de la siguiente manera:

Esp-32-CAM	Convertidor USB a Serial.
GND	GND
VCC (5Vdc)	VCC (5Vdc)
RX	TX
TX	RX
CSI_MCLK	GND

Tabla-1 Conexiones de una ESP32-CAM con un convertidor USB a serial.

El proceso para programar y correr una aplicación:

1. Realizar las conexiones como la Tabla-1.
2. Seleccionar en Arduino la tarjeta: ***AI-Thinker-ESP32-CAM***
3. Codificar el programa.
4. Descargar el programa a la tarjeta.
5. Quitar la conexión de CSI_MCLK de GND.
6. Presionar el botón reset, para correr la aplicación.

Ejemplo 1 Blink led flash de la ESP32-CAM

El programa que permite parpadear el led del FLASH de la ESP32-CAM es muy sencillo, solo tenemos que saber en qué pin se encuentra conectado dicho led. En la Figura 7, se muestra el diagrama esquemático de la ESP32-CAM referente al LED del FLASH. Como podemos apreciar en la Figura 7, el led se encuentra conectado a un transistor S8050 y la base del mismo al pin GPIO-4, por lo tanto ese número de pin (4) será nuestra referencia en Arduino para poder controlarlo digitalmente.

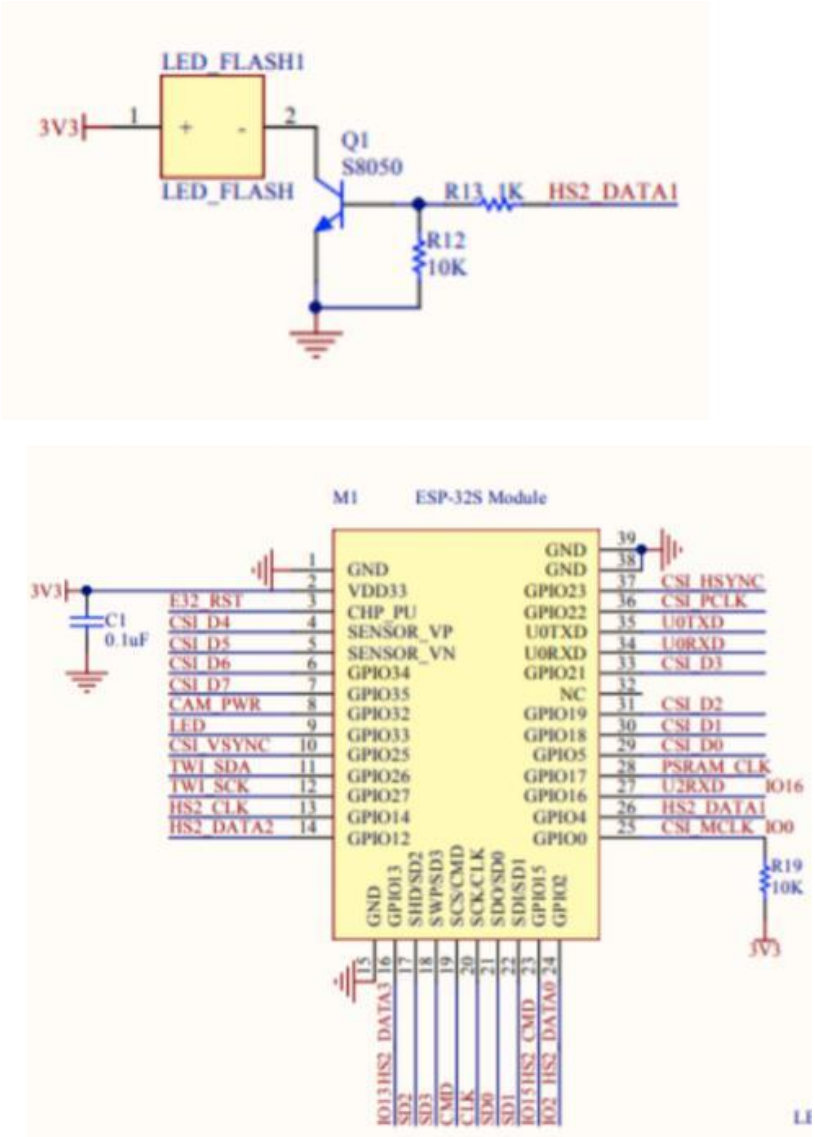

Figura 7. Diagrama esquemático del Led flash de la ESP32-CAM.

Código del Ejemplo 1

```
#define led_FLASH 4

void setup() {
  pinMode(led_FLASH, OUTPUT);
}

void loop() {
  digitalWrite(led_FLASH, LOW);
            delay (1000);
  digitalWrite(led_FLASH, HIGH);
            delay (1000);
}
```

Código 1. Blink del led flash de la ESP32-CAM.
https://gist.github.com/esmarr58/f368c9d765ed934db33431eacaad2096

Ejemplo 2 - ESP32-CAM como servidor Web con cámara

Se puede configurar a la ESP32-CAM para funcionar como un servidor web que permita visualizar una imagen tomada con la cámara OV2640 conectada al ESP32. En este tutorial te mostraré un programa que permite visualizar una imagen tomada desde la cámara de la ESP32-CAM y te la muestra en una url local.

El código es bastante sencillo, para configurarlo basta con indicar la red y contraseña de la red que se utilizara, esto se puede modificar en las variables ssid y password. Adicionalmente existe una variable llamada dns, esta sirve para cambiar el nombre del dns de la esp32. Un DNS es un nombre que se le asignará al dispositivo, aunque solo lo reconocen computadoras con Linux o IOs instalado y en caso de usar windows, se requiere instalar un software adicional que permita reconocer los DNS locales. En este caso el dns configurado se llama “cámara” por lo que para acceder a la imagen de la cámara, se utilizará la siguiente URL http://camara.local/imagen.jpg. Entonces esta es la URL que se utilizará para probar el codigo. En caso de que no tengas Linux o IOs, entonces tendrás que abrir el monitor serial de Arduino para averiguar que IP te asignó la red y en ese caso tendrás que acceder a http://IP-que-asigno-la-red/imagen.jpg. En la Figura E2.1, se muestra un ejemplo de esta configuración.

```
Connectando a la red:
.......
WiFi conectado
Direccion IP:
192.168.100.169
Direccion MAC:
C8:C9:A3:F9:31:F0
Servidor iniciado
```

Figura E2.1. Monitor serial del programa al iniciar la configuración, en este caso la red asignó la dirección 192.168.100.169, aunque cabe resaltar que esta IP cambia de acuerdo a las condiciones de su propia red.

Resoluciones válidas de la ESP32-CAM

En este caso las resoluciones probadas con la ESP32-CAM son las siguientes:

- HQVGA (240px x 160px).
- QVGA (320px, 240px).
- WQVGA (400px 240px).
- VGA(640px, 480px).
- SVGA(800px,600px);

Existen otras resoluciones que se podrían compilar pero sin embargo no producen una buena calidad de imagen, como lo es la UXGA 1600px 1200px, la XGA(1024px x 768px). En teoría para aplicaciones de machine learning es posible utilizar las resoluciones anteriores con la ESP32-CAM.

Funciones

- **conectaWiFi()**. Permite realizar la conexión con la red wifi y las variables públicas ssid y password.
- **error404()**. Es la función que se ejecuta cuando el servidor detecta que se quiere ingresar a una url que el servidor de la ESP32 no reconoce.
- **servirImagen()**. En esta función está el código necesario para suministrar al servidor web de una imagen en formato jpg. Se toma la imagen y se le proporciona al cliente.
- **setup()**. Configuraciones iniciales del programa, como la url donde puede verse la imagen, la configuración del wifi y de la cámara.
- **loop()**. Es el loop principal donde se está esperando por las conexiones entrantes.

Configuración de setup

Para poder utilizar la camara de la esp32, se realizan las siguientes instrucciones:

- using namespace esp32cam; . Es el nombre del espacio de las funciones y clases de la biblioteca esp32cam.h.
- Config configuracion; La variable de configuración.
- configuracion.setPins(pins::AiThinker); Se configuran los pines de conexión de la ESP32-CAM.
- configuracion.setResolution(QVGA); Se establece la resolución.
- configuracion.setBufferCount(5); . Establece el número de imágenes en el buffer
- configuracion.setJpeg(80); Se configura la calidad de la imagen.
- bool ok = Camera.begin(configuracion);. Se establece la configuración a la cámara.

Código.

```
#include <WebServer.h>
#include <WiFi.h>
#include <ESPmDNS.h>
#include "esp_system.h"
#include <esp32cam.h>

static auto HQVGA = esp32cam::Resolution::find(240, 160);
static auto QVGA = esp32cam::Resolution::find(320, 240);
static auto WQVGA = esp32cam::Resolution::find(400, 240);
static auto VGA = esp32cam::Resolution::find(640, 480);
static auto SVGA = esp32cam::Resolution::find(800, 600);

const char *ssid = "****";
const char *password = "****";
const char *dns = "camara";



WebServer servidor(80);


void conectaWiFi(){

   Serial.println();
   Serial.print("Connectando a la red: ");
   Serial.println(ssid);

   WiFi.persistent(false); //Permite que la configuracion del Wifi no sea afectada.
   WiFi.mode(WIFI_STA);    //Configura el Wifi en modo estacion
   WiFi.begin(ssid,password);

   while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
   }


   Serial.println("");
   Serial.println("WiFi conectado");
   Serial.println("Direccion IP: ");
   Serial.println(WiFi.localIP());
   Serial.println("Direccion MAC: ");
   Serial.println(WiFi.macAddress());
   //http://camara.local   //Servidor dns que sustituye a la IP (solo funciona con Linux o
```

```
IOS) Para windows se requiere software adicional

    if (!MDNS.begin(dns)) {
          Serial.print("Se configuro el DNS: http://");
          Serial.print(dns);
          Serial.println(".local/");

    }
    else{
        MDNS.addService("http", "tcp", 80);
    }

}

void error404() {
    servidor.send(200, "text/plain", "Error pagina no encontrada");
}

void servirImagen() {

  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    servidor.send(503, "", "");

    return;
  }
   Serial.printf("IMAGEN TOMADA %dx%d %db\n", frame->getWidth(),
frame->getHeight(),
               static_cast<int>(frame->size()));

  servidor.setContentLength(frame->size());
  servidor.send(200, "image/jpeg");
  WiFiClient client = servidor.client();
  frame->writeTo(client);


}

void setup() {
  Serial.begin(115200);
  using namespace esp32cam;
  Config configuracion;
  configuracion.setPins(pins::AiThinker);
  configuracion.setResolution(QVGA);
  configuracion.setBufferCount(5);  //Establece el numero de imagenes en el buffer
  configuracion.setJpeg(80);
  bool ok = Camera.begin(configuracion); //Inicializa la camara con la configuracion previa
  Serial.println(ok ? "CAMERA OK" : "Error al inicializar la camara");
  conectaWiFi();
  servidor.on("/imagen.jpg", servirImagen); //En la funcion servirImagenes se creara la
pagina /imagen.jpg
  servidor.onNotFound(error404);
  servidor.begin();
  Serial.println("Servidor iniciado");
```

```
}

void loop() {
  servidor.handleClient();
}
```

Enlace a github

https://gist.github.com/esmarr58/265cfbccea675c4a6330f2647bc8a7c5