Guia de pinagem do ESP32-CAM AI-Thinker: explicação do uso de GPIOs

A ESP32-CAM é uma placa de desenvolvimento com chip ESP32-S, câmera OV2640, slot para cartão microSD e vários GPIOs para conectar periféricos. Neste guia, veremos os GPIOs ESP32-CAM e como usá-los.

Diagrama de pinagem

A imagem a seguir mostra o diagrama de pinagem do ESP32-CAM AI-Thinker .

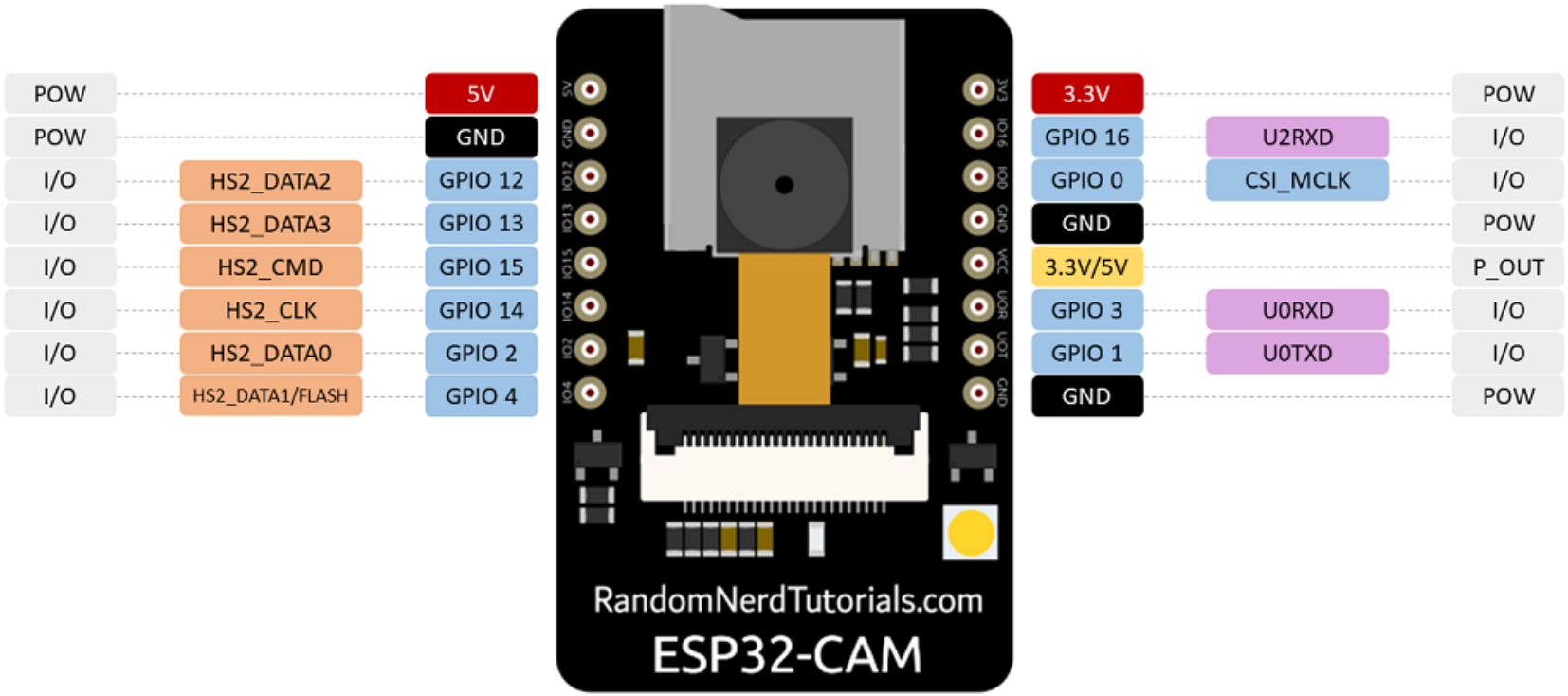
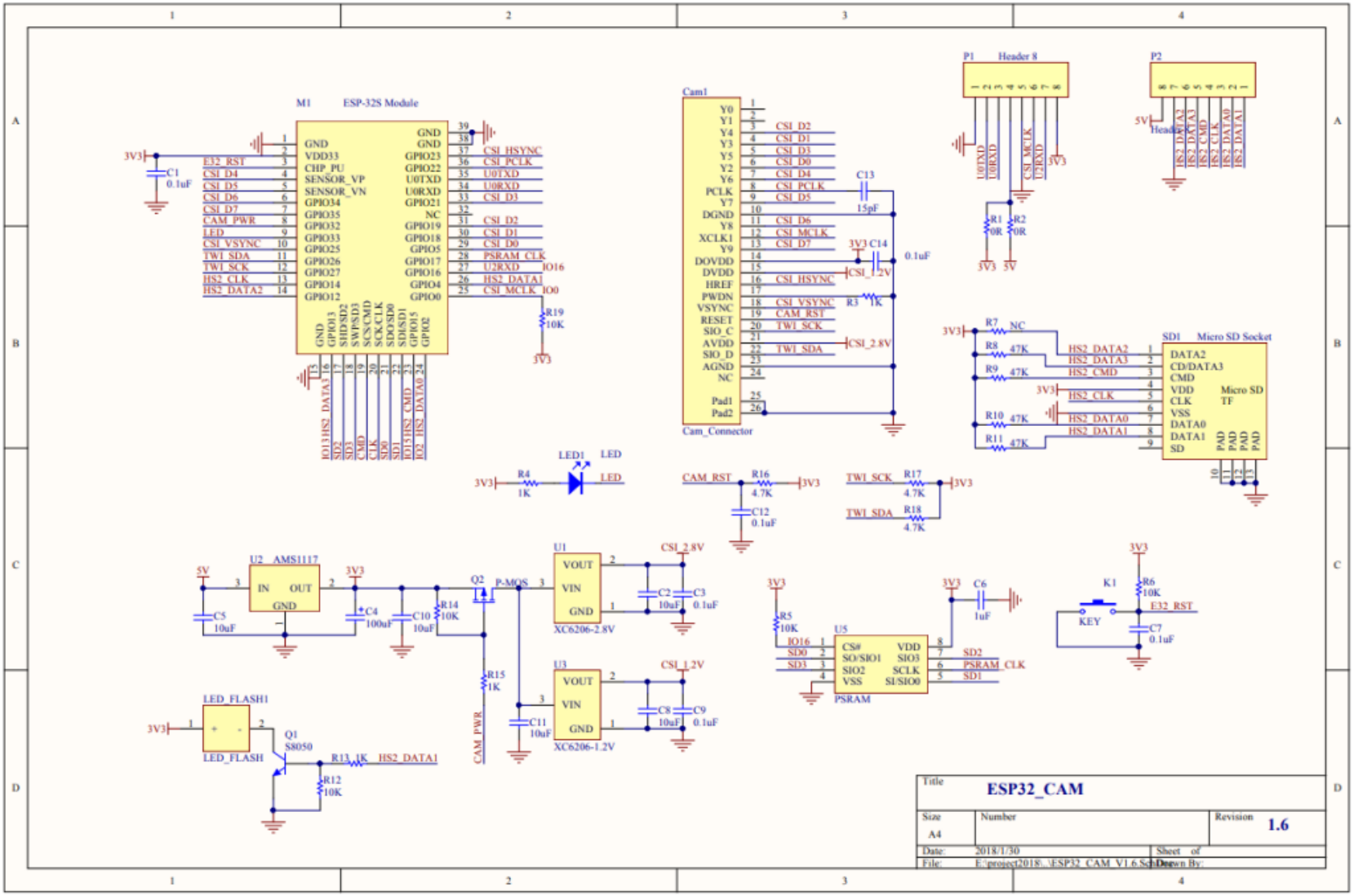


Diagrama esquemático

A figura a seguir mostra o diagrama esquemático para o ESP32-CAM.



Fonte da imagem

Você pode baixar um arquivo PDF com melhor resolução neste repositório GitHub .

Pinos de alimentação

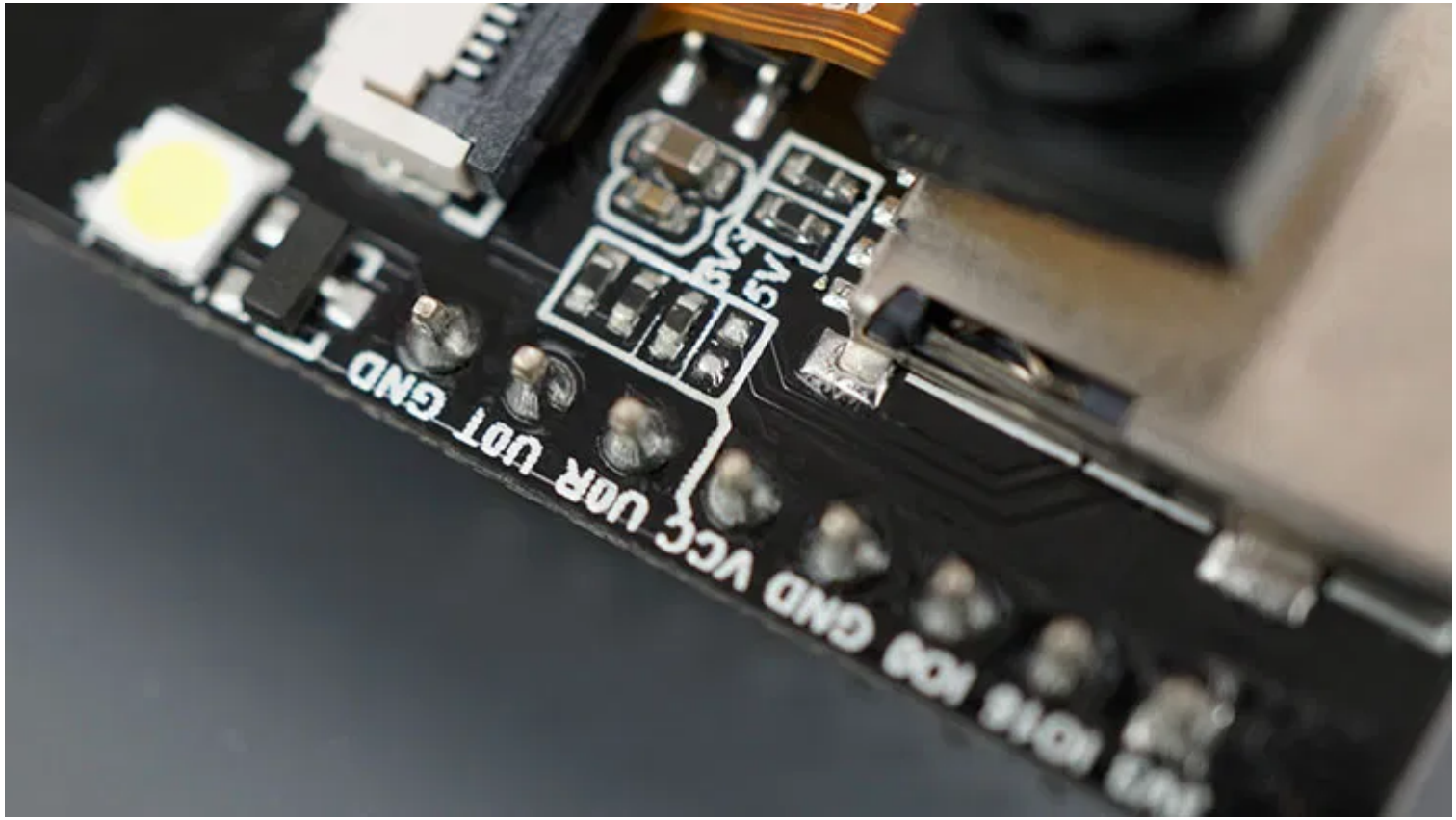
O ESP32-CAM vem com três GND pinos (coloridos na cor preta) e dois pinos de alimentação (coloridos na cor vermelha): 3,3 V e 5V .

Você pode alimentar o ESP32-CAM através do 3,3 V ou 5V alfinetes. Porém, muitas pessoas relataram erros ao alimentar o ESP32-CAM com 3.3V, então sempre aconselhamos **alimentar o ESP32-CAM pelo pino 5V** .

Pino de saída de energia

Há também o pino rotulado na serigrafia como **VCC** (colorido com um retângulo amarelo). Você não deve usar esse pino para alimentar o ESP32-CAM. Esse é um pino de alimentação de saída. Pode produzir 5V ou 3,3V.

No nosso caso, o ESP32-CAM produz 3,3V, seja alimentado com 5V ou 3,3V. Ao lado do pino VCC, existem dois pads. Um rotulado como 3.3V e outro como 5V.



Se você olhar de perto, você deve ter um jumper nos pads de 3,3V. Se você quiser ter uma saída de 5V no pino VCC, você precisa dessoldar essa conexão e soldar os pads de 5V.

Pinos de série

GPIO 1 e GPIO 3 são os pinos seriais (TX e RX, respectivamente). Como o ESP32-CAM não possui um programador embutido, você precisa usar esses pinos para se comunicar com a placa e fazer upload do código.

A melhor maneira de fazer upload de código para o ESP32-CAM é usando um programador FTDI .

Saiba como fazer upload de código para o ESP32-CAM AI-Thinker.

Você pode usar GPIO 1 e GPIO 3 para conectar outros periféricos como saídas ou sensores após o upload do código. No entanto, você não poderá abrir o Serial Monitor e ver se tudo está indo bem com sua configuração.

GPIO 0

GPIO 0 determina se o ESP32 está em modo intermitente ou não. Este GPIO é conectado internamente a um resistor pull-up de 10k Ohm.

Quando o GPIO 0 está conectado ao GND, o ESP32 entra em modo intermitente e você pode fazer o upload do código para a placa.

- GPIO 0 conectado a GND » ESP32-CAM em modo intermitente

Para fazer o ESP32 rodar “normalmente”, basta desconectar o GPIO 0 do GND.

Conexões de cartão microSD

Os seguintes pinos são usados para fazer a interface com o cartão microSD quando ele está em operação.

Cartão microSD	**ESP32**
CLK	GPIO 14

CMD	GPIO 15
DATA0	GPIO 2
DATA1 / lanterna	GPIO 4
DADOS2	GPIO 12
DADOS3	GPIO 13

Se você não estiver usando o cartão microSD, poderá usar esses pinos como entradas/saídas regulares. Você pode dar uma olhada no guia de pinagem do ESP32 para ver os recursos desses pinos.

Todos esses GPIOs são RTC e suportam ADC: GPIOs 2, 4, 12, 13, 14 e 15.

Lanterna (GPIO 4)

O ESP32-CAM possui um LED embutido muito brilhante que pode funcionar como flash ao tirar fotos. Esse LED está conectado internamente ao GPIO 4 .

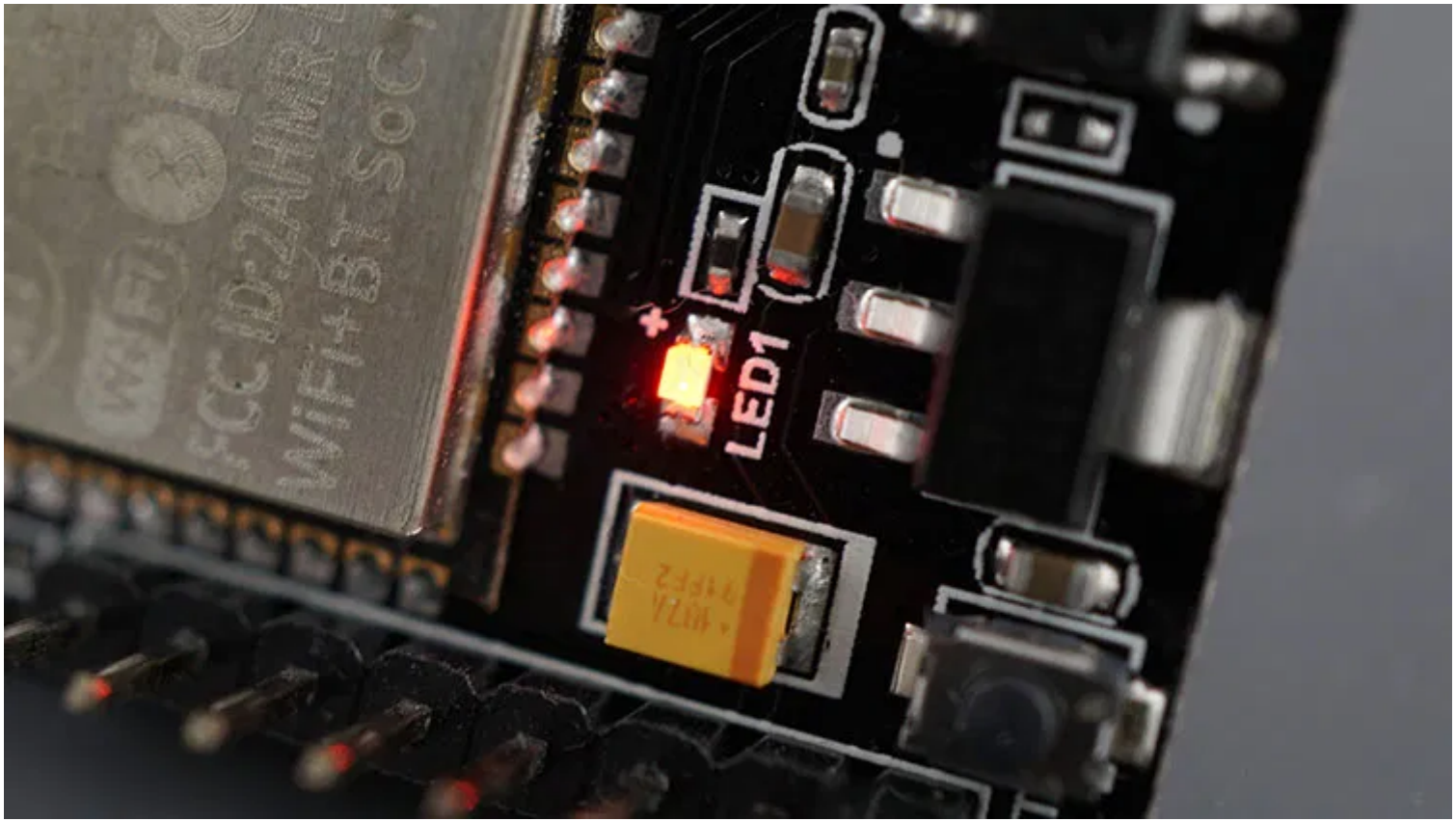
Esse GPIO também está conectado ao slot do cartão microSD, então você pode ter problemas ao tentar usar os dois ao mesmo tempo - a lanterna acenderá ao usar o cartão microSD.

Nota: um de nossos leitores compartilhou que se você inicializar o cartão microSD da seguinte forma, você não terá esse problema porque o cartão microSD não usará essa linha de dados.*

```
SD_MMC.begin("/sdcard", true)
```

* descobrimos que isso funciona e que o LED não fará esse efeito de flash. No entanto, o LED permanece aceso com baixo brilho – não temos certeza se está faltando alguma coisa.

GPIO 33 – LED vermelho embutido



Ao lado do botão RST, há um LED vermelho integrado. Esse LED está conectado internamente ao GPIO 33 . Você pode usar este LED para indicar que algo está acontecendo. Por exemplo, se o Wi-Fi estiver conectado, o LED fica vermelho ou vice-versa.

Esse LED funciona com lógica invertida, então você envia um `BAIXO` sinal para ligá-lo e um `ALTO` sinal para desligar.

Você pode experimentar o upload do trecho a seguir e ver se o LED acende.

```
void setup() {
  pinMode(33, OUTPUT);
}

void loop() {
  digitalWrite(33, LOW);
}
```

Conexões da câmera

As conexões entre a câmera e o ESP32-CAM AI-Thinker são mostradas na tabela a seguir.

CÂMERA OV2640	**ESP32**	**Nome da variável no código**
D0	GPIO 5	Y2_GPIO_NUM
D1	GPIO 18	Y3_GPIO_NUM

D2	GPIO 19	Y4_GPIO_NUM
D3	GPIO 21	Y5_GPIO_NUM
D4	GPIO 36	Y6_GPIO_NUM
D5	GPIO 39	Y7_GPIO_NUM
D6	GPIO 34	Y8_GPIO_NUM
D7	GPIO 35	Y9_GPIO_NUM
XCLK	GPIO 0	XCLK_GPIO_NUM
PCLK	GPIO 22	PCLK_GPIO_NUM
VSYNC	GPIO 25	VSYNC_GPIO_NUM
HREF	GPIO 23	HREF_GPIO_NUM
SDA	GPIO 26	SIOD_GPIO_NUM
SCL	GPIO 27	SIOC_GPIO_NUM
PIN DE ENERGIA	GPIO 32	PWDN_GPIO_NUM

Portanto, a definição de pinos para o ESP32-CAM AI-Thinker no Arduino IDE deve ser a seguinte:

```
#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
```