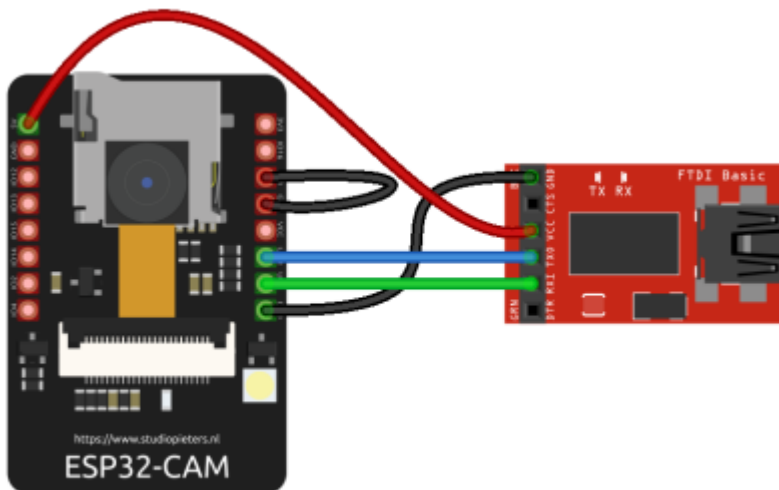


# Doorbell

[ESP32CAM](#)

[Esphome camera component](#)

[Esphome RTSPserver github](#)



il jumper tra IO0 e GND va inserito solo per la programmazione

Per la prima scrittura:

1. connettere uno dei pin di IO a GND, come mostrato in figura
2. attaccare l'USB
3. dare i permessi alla porta, esempio:

```
chmod 660 /dev/ttyUSB0
```

4. riavviare esphome
5. uploadare la configurazione

Per le scritture wireless non serve che GND sia connesso a un pin di IO

Configurazione base per camera e led

```
esphome:
  name: doorbell
  platform: ESP32
  board: esp32dev

# Enable logging
logger:

# Enable Home Assistant API
api:
```

```
wifi:
  ssid: "wifi ssid"
  password: "password"

esp32_camera:
  external_clock:
    pin: GPIO0
    frequency: 20MHz
  i2c_pins:
    sda: GPIO26
    scl: GPIO27
  data_pins: [GPIO5, GPIO18, GPIO19, GPIO21, GPIO36, GPIO39, GPIO34, GPIO35]
  vsync_pin: GPIO25
  href_pin: GPIO23
  pixel_clock_pin: GPIO22
  power_down_pin: GPIO32
  name: doorbell camera

# Flashlight CHECK PIN
output:
  - platform: gpio
    pin: GPIO4
    id: gpio_4

light:
  - platform: binary
    output: gpio_4
    name: doorbell light

sensor:
  - platform: wifi_signal
    name: doorbell wifi signal
    update_interval: 30s
  - platform: uptime
    name: doorbell uptime
```

## RTSP Server

Installazione del fork di esphome, che sostituirà esphome originale

```
mkdir esphome_rtsp
cd esphome_rtsp
git clone https://github.com/crossan007/esphome
cd esphome
checkout feature/rtsp-server
pip install .
esphome ~/config dashboard
```

Aggiungere alla configurazione di esp32cam:

```
esp32_camera:
  # id per rtsp server
  id: cam
  external_clock:
    pin: GPIO0
    ...
    ...

rtsp_server:
  port: 8675
  camera: cam
```

Visualizzare stream RTSP con VLC:

VLC menu -> Media -> Open network stream -> rtsp://indirizzo\_ip\_esp:8675

## HTTP Server

In alternativa si può usare anche questo componente http e convertire successivamente in rtsp quando lo stream passa per il raspberry. Nel mio caso stranamente ha dato risultati migliori.

[https://github.com/ayufan/esphome-components#25-esp32\\_camera\\_web\\_server](https://github.com/ayufan/esphome-components#25-esp32_camera_web_server)

## Conversione del video

La camera manda una sequenza di immagini JPEG con framerate variabile. Per mostrarle su Alexa è necessario convertirle in uno stream video con codec H264 (più leggero) o MPEG4. Dato che la potenza dell'esp è limitata è meglio affidare questo compito al raspberry.

- Lanciare un server rtsp sul raspberry, ad esempio <https://github.com/aler9/rtsp-simple-server> (togliere i servizi inutili nella configurazione)
- `sudo apt-get install ffmpeg`
- Lanciare uno tra questi comandi sul raspberry per prendere il flusso della camera, convertirlo in video e lanciare lo stream sul server rtsp

Formato H264 (più compresso)

```
ffmpeg -use_wallclock_as_timestamps 1 -i http://192.168.1.254:8080 -c:v libx264 -preset veryslow -tune zerolatency -movflags +faststart -pix_fmt yuv420p -disposition:v:0 default -r 10 -g 1 -f rtsp rtsp://192.168.1.176:8443/stream
```

Formato MPEG4

```
ffmpeg -use_wallclock_as_timestamps 1 -i http://192.168.1.254:8080 -c:v mpeg4 -tune zerolatency -movflags +faststart -pix_fmt yuv420p -disposition:v:0 default -r 10 -g 1 -f rtsp rtsp://192.168.1.176:8443/stream
```

Diminuire il framerate (-r) e aumentare il gop (-g) a piacere

Mentre ffmpeg è in esecuzione è importante guardare il numero di frame duplicate e droppate da ffmpeg nella barra di stato, e aggiustare il framerate di conseguenza (modificarlo anche nella configurazione esphome della camera)

Il preset veryslow comprime molto e tiene leggera la banda. Se la speed di ffmpeg va sotto a x1 si può usare un preset più veloce per alleggerire il lavoro al processore.

From:

<https://wiki.csgalileo.org/> - **Galileo Labs**

Permanent link:

<https://wiki.csgalileo.org/projects/iotaiuto/doorbell?rev=1633076458>

Last update: **2021/10/01 10:20**

