

ESP Cam

Hardware

hier siehst du die Hardware die ich verwende

[ESP Cam](#)

[FTDI Adapter](#)

[Jumpwire](#)

Software

 **Download**

Hier muss noch etwas vorbereitet werden:

image not found or type unknown



Gehe in die Einstellungen und füge:

`https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json`

in den additional board managern hinzu.

image not found or type unknown



Jetzt kannst du noch den Board Manager installieren:

image not found or type unknown



Image not found or type unknown



Sketch Code

```
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h" //disable brownout problems
#include "soc/rtc_cntl_reg.h" //disable brownout problems
#include "esp_http_server.h"

//Replace with your network credentials
const char* ssid = "WLAN NAME";
const char* password = "WLAN PASSWORT";

#define PART_BOUNDARY "1234567890000000000000987654321"

// This project was tested with the AI Thinker Model, M5STACK PSRAM Model and M5STACK WITHOUT PSRAM
#define CAMERA_MODEL_AI_THINKER
// #define CAMERA_MODEL_M5STACK_PSRAM
// #define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM

// Not tested with this model
// #define CAMERA_MODEL_WROVER_KIT

#if defined(CAMERA_MODEL_WROVER_KIT)
#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM  -1
#define XCLK_GPIO_NUM    21
#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    19
#define Y4_GPIO_NUM    18
#define Y3_GPIO_NUM     5
#define Y2_GPIO_NUM     4
#define VSYNC_GPIO_NUM 25
#define HREF_GPIO_NUM   23
#define PCLK_GPIO_NUM   22
```

```
#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
```

```
#define PWDN_GPIO_NUM  -1
#define RESET_GPIO_NUM 15
#define XCLK_GPIO_NUM   27
#define SIOD_GPIO_NUM   25
#define SIOC_GPIO_NUM   23
```

```
#define Y9_GPIO_NUM    19
#define Y8_GPIO_NUM    36
#define Y7_GPIO_NUM    18
#define Y6_GPIO_NUM    39
#define Y5_GPIO_NUM     5
#define Y4_GPIO_NUM    34
#define Y3_GPIO_NUM    35
#define Y2_GPIO_NUM    32
#define VSYNC_GPIO_NUM 22
#define HREF_GPIO_NUM   26
#define PCLK_GPIO_NUM   21
```

```
#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
```

```
#define PWDN_GPIO_NUM  -1
#define RESET_GPIO_NUM 15
#define XCLK_GPIO_NUM   27
#define SIOD_GPIO_NUM   25
#define SIOC_GPIO_NUM   23
```

```
#define Y9_GPIO_NUM    19
#define Y8_GPIO_NUM    36
#define Y7_GPIO_NUM    18
```

```

#define Y6_GPIO_NUM    39
#define Y5_GPIO_NUM    5
#define Y4_GPIO_NUM    34
#define Y3_GPIO_NUM    35
#define Y2_GPIO_NUM    17
#define VSYNC_GPIO_NUM 22
#define HREF_GPIO_NUM  26
#define PCLK_GPIO_NUM  21

#elif 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 "Camera model not selected"

#endif

static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary=" PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";

httpd_handle_t stream_httpd = NULL;

static esp_err_t stream_handler(httpd_req_t *req){
    camera_fb_t * fb = NULL;

```

```

esp_err_t res = ESP_OK;
size_t _jpg_buf_len = 0;
uint8_t * _jpg_buf = NULL;
char * part_buf[64];

res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
if(res != ESP_OK){
    return res;
}

while(true){
    fb = esp_camera_fb_get();
    if (!fb) {
        Serial.println("Camera capture failed");
        res = ESP_FAIL;
    } else {
        if(fb->width > 400){
            if(fb->format != PIXFORMAT_JPEG){
                bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
                esp_camera_fb_return(fb);
                fb = NULL;
                if(!jpeg_converted){
                    Serial.println("JPEG compression failed");
                    res = ESP_FAIL;
                }
            } else {
                _jpg_buf_len = fb->len;
                _jpg_buf = fb->buf;
            }
        }
    }
    if(res == ESP_OK){
        size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
        res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
        res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if(res == ESP_OK){

```

```

    res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY, strlen(_STREAM_BOUNDARY));
}
if(fb){
    esp_camera_fb_return(fb);
    fb = NULL;
    _jpg_buf = NULL;
} else if(_jpg_buf){
    free(_jpg_buf);
    _jpg_buf = NULL;
}
if(res != ESP_OK){
    break;
}

//Serial.printf("MJPEG: %uB\n", (uint32_t)(_jpg_buf_len));
}
return res;
}

```

```

void startCameraServer(){
    httpd_config_t config = HTTPD_DEFAULT_CONFIG();
    config.server_port = 80;

    httpd_uri_t index_uri = {
        .uri      = "/",
        .method    = HTTP_GET,
        .handler   = stream_handler,
        .user_ctx  = NULL
    };

    //Serial.printf("Starting web server on port: '%d'\n", config.server_port);
    if (httpd_start(&stream_httpd, &config) == ESP_OK) {
        httpd_register_uri_handler(stream_httpd, &index_uri);
    }
}

```

```

void setup() {
    WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector

    Serial.begin(115200);
}

```

```
Serial.setDebugOutput(false);

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;
}

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

```

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

Serial.print("Camera Stream Ready! Go to: http://");
Serial.print(WiFi.localIP());

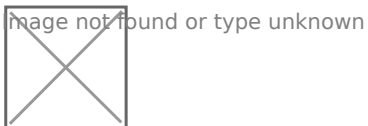
// Start streaming web server
startCameraServer();
}

void loop() {
  delay(1);
}

```

Verkabelung

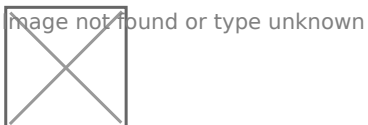
hier ist es nun wichtig alles richtig zwischen CAM und FTDI zu verbinden:



bitte vergess nicht die hier Grau eingezeichneten Brücke

Flashing

Jetzt verbindest du deinen FTDI mit einem USB deines PCs und startest den Flashvorgang wie folgt:



Board Typ auswählen (hier musste ich sehr weit runter scrollen)

image not found or type unknown



COM Port wählen

Starte nun den Flash Vorgang mit dem Pfeil

image not found or type unknown



Dein Ergebnis sollte wiefolgt aussehen:

image not found or type unknown



Öffne jetzt den Serial Monitor

image not found or type unknown



Jetzt musst du die Brücke entfernen und den RST Button auf der Cam drücken

Hier bekommst du dann folgende Ausgabe:

image not found or type unknown



IP Adresse notieren

Einbinden in Homeassistant

Hier können wir die MJPEG Camera als Integration verwenden.

Also gehen wir zu unseren Integrationen und fügen diese hinzu:

image not found or type unknown



Bitte beachtet hier den SSL Haken zu entfernen und vergesst das HTTP vor der URL nicht.

Alternative ESP Home

Als alternative kann man auch ESP Home verwenden hier ist die Kamera aber nicht ganz so flüssig:

```
esphome:
  name: cam-1
  platform: ESP32
  board: esp32dev

# Enable logging
logger:

# Enable Home Assistant API
api:
  reboot_timeout: 0s

ota:
  platform: esphome

web_server:
  port: 80

wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password

# Enable fallback hotspot (captive portal) in case wifi connection fails
ap:
  ssid: "Esp32-Cam Fallback Hotspot"
  password: "C3FOhAPFtouw"

captive_portal:

# Example configuration entry
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

Image settings

name: Cam 1

Flashlight

output:

- platform: gpio

pin: GPIO4

id: gpio_4

GPIO_4 is the flash light pin

light:

- platform: binary

output: gpio_4

name: cam-1-flashlight

<https://www.youtube.com/embed/q7Q3XWDxNa0>

Revision #2

Created 17 March 2023 11:02:08 by Cryd

Updated 29 June 2024 06:06:40 by Cryd