90+ ESP32-CAM projekt, oktató-anyag és útmutató Arduino IDE-vel

Fordította: Maczák András

andras@maczak.eu

Tartalomjegyzék

Az ESP32-CAM kártya egy megfizethető fejlesztőkártya, amely egy ESP32-S chipet, egy OV2640 kamerát, több GPIO-t a perifériák csatlakoztatásához és egy microSD kártyanyílást kombinál. Lehetővé teszi video streaming webszerver beállítását, térfigyelő kamera építését, fényképek készítését, arcfelismerést és felismerést és még sok mást.

Számos ESP32-CAM oktatóanyagunk és projektötletünk van. Beszerezheted a Build ESP32-CAM Projects eBookunkat is, amely több mint 90 projektet tartalmaz. A következő gyorshivatkozások segítségével megtalálhatod az összes ESP32-CAM útmutatónkat, amelyekben könnyen követhető, lépésről lépésre található utasítások, áramköri rajzok, forráskód, képek és videók.

ESP32-CAM Video Streaming és arcfelismerés Arduino IDE-vel

Ez a cikk egy gyors útmutató az ESP32-CAM kártya használatához. Megmutatjuk, hogyan állíthatsz be egy video streaming webszervert arcfelismeréssel és észleléssel kevesebb mint 5 perc alatt az Arduino IDE segítségével.

Megjegyzés: ebben az oktatóanyagban az arduino-esp32 könyvtár példáját használjuk. Ez az oktatóanyag nem foglalkozik a példa módosításával.

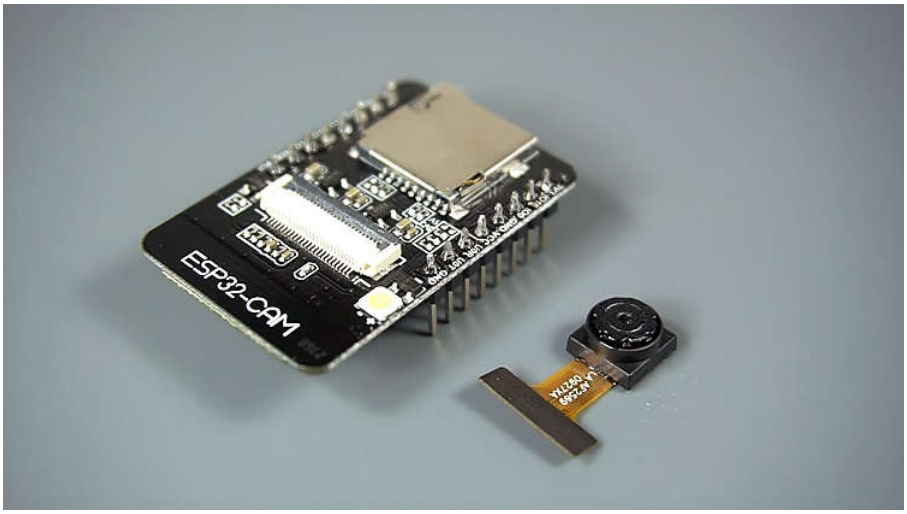
Kapcsolódó projekt: ESP32-CAM Video Streaming Web Server (Home Assistant és Node-Red funkcióval működik)

Szükséges alkatrészek

Az oktatóanyag követéséhez a következő alkatrészekre van szükséged:

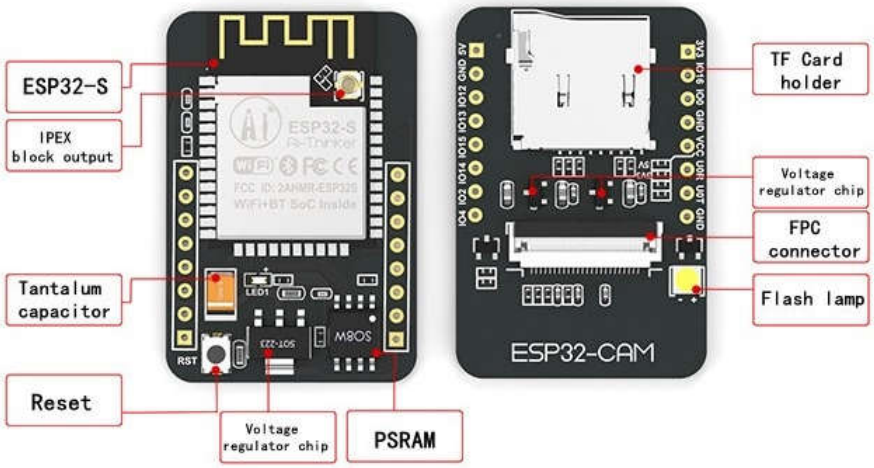
- ESP32-CAM OV2640-el
- FTDI programozó
- Hüvelyes áthidaló vezetékek

Bemutatkozik az ESP32-CAM

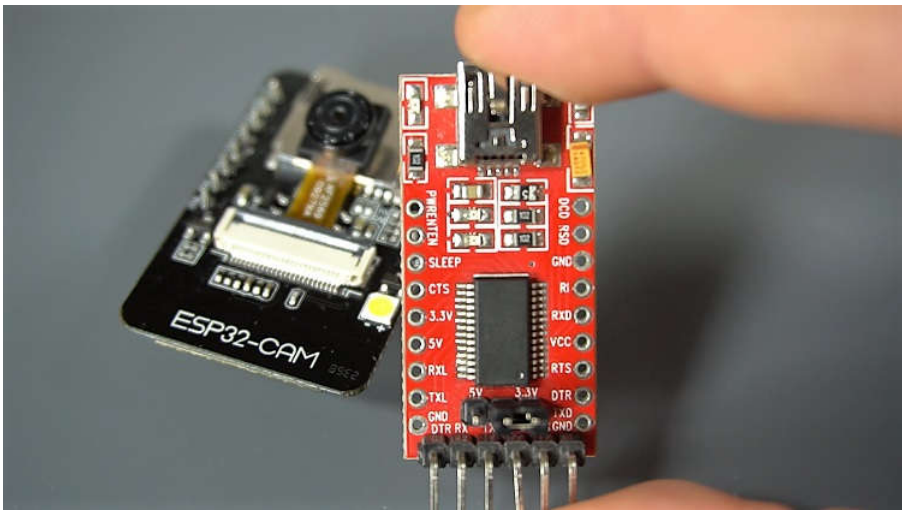


Az ESP32-CAM egy nagyon kicsi kameramodul ESP32-S chippel, amely körülbelül 10 dollárba kerül. Az OV2640 kamera és a perifériák csatlakoztatására szolgáló több GPIO mellett egy microSD kártyanyílás

is található, amely hasznos lehet a kamerával készített képek tárolására vagy a kliensek számára biztosított fájlok tárolására.



Az ESP32-CAM nem rendelkezik USB-csatlakozóval, ezért szükséged van egy FTDI-programozóra a kód feltöltéséhez az U0R és U0T érintkezőkön (soros érintkezőkön) keresztül.



Jellemzők

Itt van egy lista az ESP32-CAM jellemzőiről:

- A legkisebb 802.11b/g/n Wi-Fi BT SoC modul
- Alacsony fogyasztású 32 bites CPU, az alkalmazás processzort is ki tudja szolgálni
- Akár 160 MHz-es órajel, összesített számítási teljesítmény akár 600 DMIPS
- Beépített 520 KB SRAM, külső 4 MB PSRAM
- Támogatja az UART/SPI/I2C/PWM/ADC/DAC
- OV2640 és OV7670 kamerák támogatása, beépített vakulámpa
- Támogatja a WiFi képfeltöltést
- Támogatja a TF kártyát
- Több alvási módot támogat
- Beágyazott Lwip és FreeRTOS

- Támogatja az STA/AP/STA+AP üzemmódot
- Támogatja a Smart Config/AirKiss technológiát
- Soros port helyi és távoli firmware-frissítések (FOTA) támogatása

Az ESP32-CAM kivezetései

Három GND érintkező és két érintkező van a tápellátáshoz: 3,3 V vagy 5 V.

A GPIO 1 és GPIO 3 a soros érintkezők. Ezekre a tűkre van szükség a kód feltöltéséhez a kártyára. Ezenkívül a GPIO 0 is fontos szerepet játszik, mivel ez határozza meg, hogy az ESP32 flash üzemmódban van-e vagy sem. Ha a GPIO 0 a GND-hez van csatlakoztatva, az ESP32 flash üzemmódban van.

A következő érintkezők belsőleg csatlakoznak a microSD kártyaolvasóhoz:

- GPIO 14: CLK
- GPIO 15: CMD
- GPIO 2: Data 0
- GPIO 4: Data 1 (a fedélzeti LED-hez is csatlakoztatva)
- GPIO 12: Data 2
- GPIO 13: Data 3

Video Streaming szerver

Kövesd a következő lépéseket egy video streaming webszerver felépítéséhez az ESP32-CAM segítségével, amelyet a helyi hálózaton érhetsz el.

Fontos: Győződj meg arról, hogy frissítetted az Arduino IDE-jét, valamint az ESP32 kiegészítő legújabb verzióját.

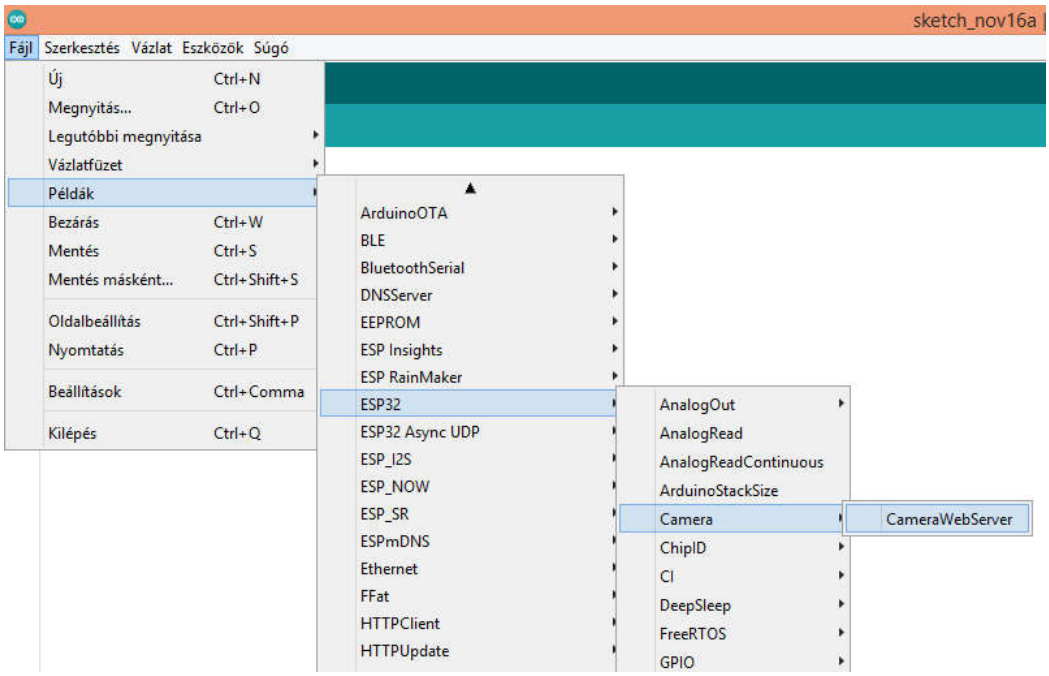
1. Az ESP32 kiegészítő telepítése

Ebben a példában az Arduino IDE-t használjuk az ESP32-CAM kártya programozásához. Tehát telepítened kell az Arduino IDE-t, valamint az ESP32 kiegészítőt. Kövesd a következő oktatóanyagot az ESP32 bővítmény telepítéséhez, haezt még nem tetted meg:

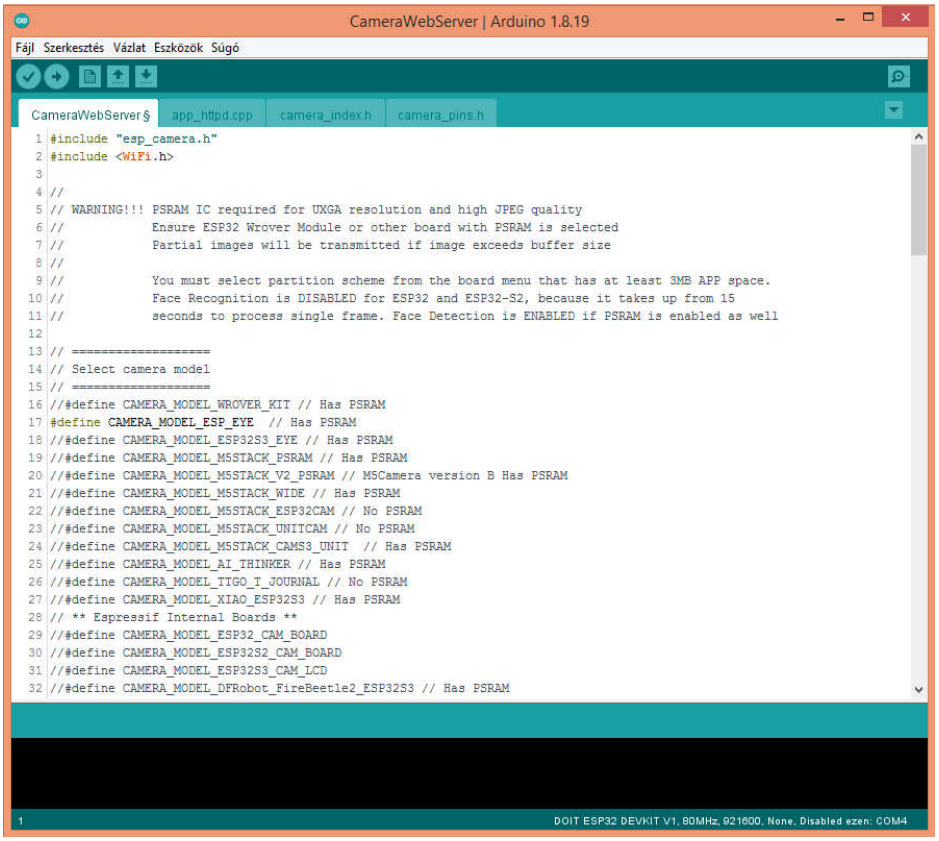
- Az ESP32 kártya telepítése Arduino IDE 2-ben (Windows, Mac OS X, Linux)

2. CameraWebServer példakód

Az Arduino IDE-ben lépj a **Fájl > Példák > ESP32 > Camera** menüpontra, és nyisd meg a **CameraWebServer** példát.



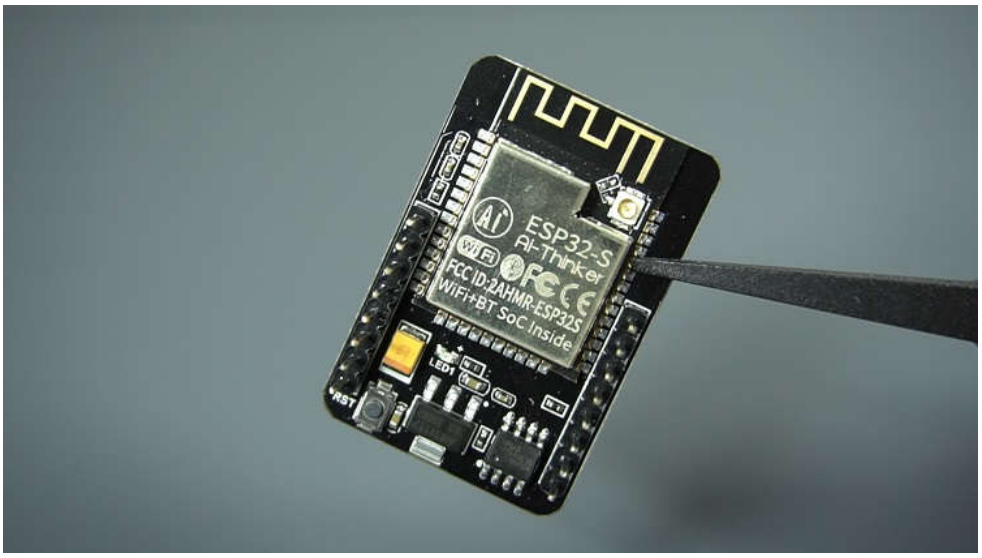
A következő kódot kell betölteni.



A kód feltöltése előtt be kell illesztenie hálózati hitelesítő adatait a következő változókba:

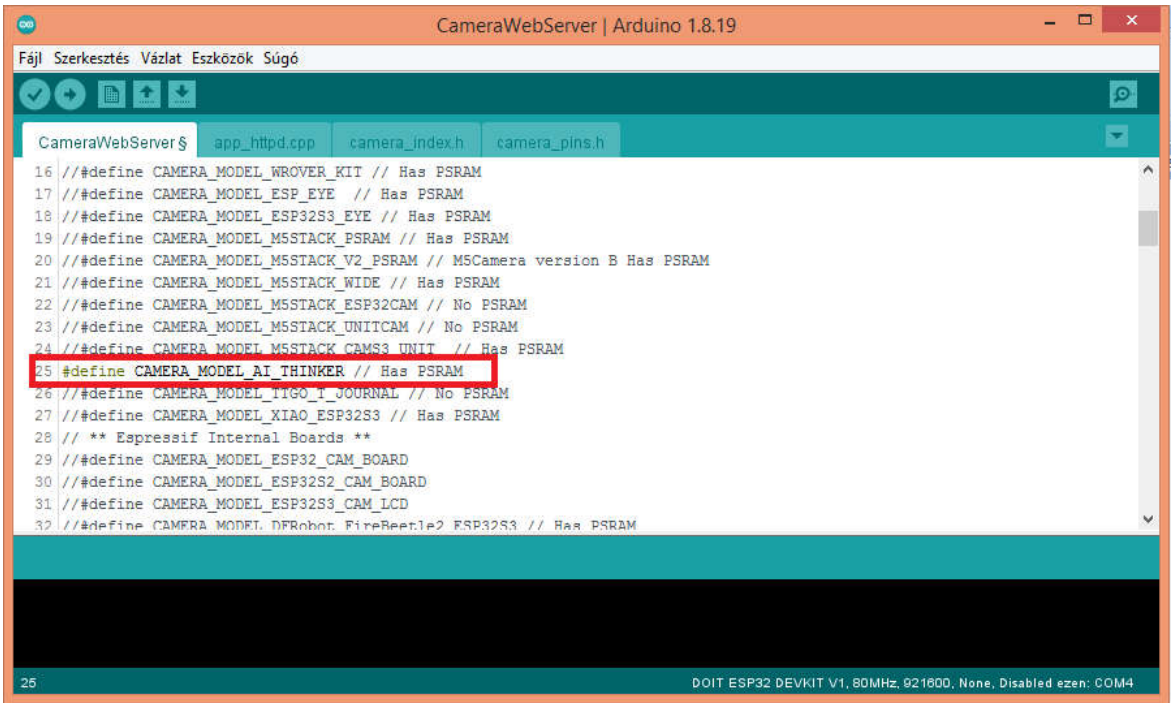
```
const char *ssid = "**********";
const char *password = "**********";
```

Ezután győződj meg arról, hogy a megfelelő kameramodult választottad. Ebben az esetben az AI-THINKER modellt használjuk.



Tehát kommentezd az összes többi modellt, és töröld a CAMERA_MODEL_AI_THINKER megjegyzését:

```
// Select camera model
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER
```

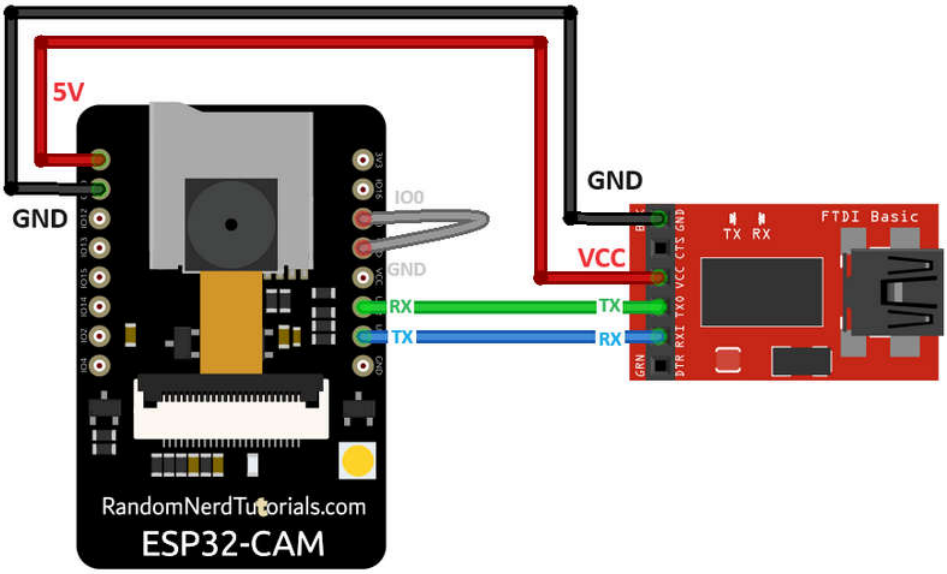


Ha a rendelkezésre álló lehetőségek egyike sem felel meg az általad használt kamerának, hozzá kell adnia az adott kártyához tartozó tű-hozzárendelést a ***camera_pins.h*** lapon.

Nos, a kód készen áll az ESP32-re való feltöltésre.

3. ESP32-CAM kód feltöltése

Csatlakoztass az ESP32-CAM kártyát a számítógépedhez egy FTDI programozó segítségével. Kövesd a következő sematikus vázlatot *(A fordító megjegyzése: a dokumentum végén alternatív megoldást mutatok be a feltöltés végrehajtásához.)*:



Sok FTDI programozó rendelkezik jumperrel, amely lehetővé teszi a 3,3 V vagy az 5 V kiválasztását. Győződj meg arról, hogy a jumper a megfelelő helyen van az 5V kiválasztásához.

Fontos: A GPIO 0-nak csatlakoznia kell a GND-hez, hogy fel tudd tölteni a kódot.

ESP32-CAM	FTDI Programmer
GND	GND
5V	VCC (5V)
U0R	TX
U0T	RX
GPIO 0	GND

A kód feltöltéséhez kövesd a következő lépéseket:

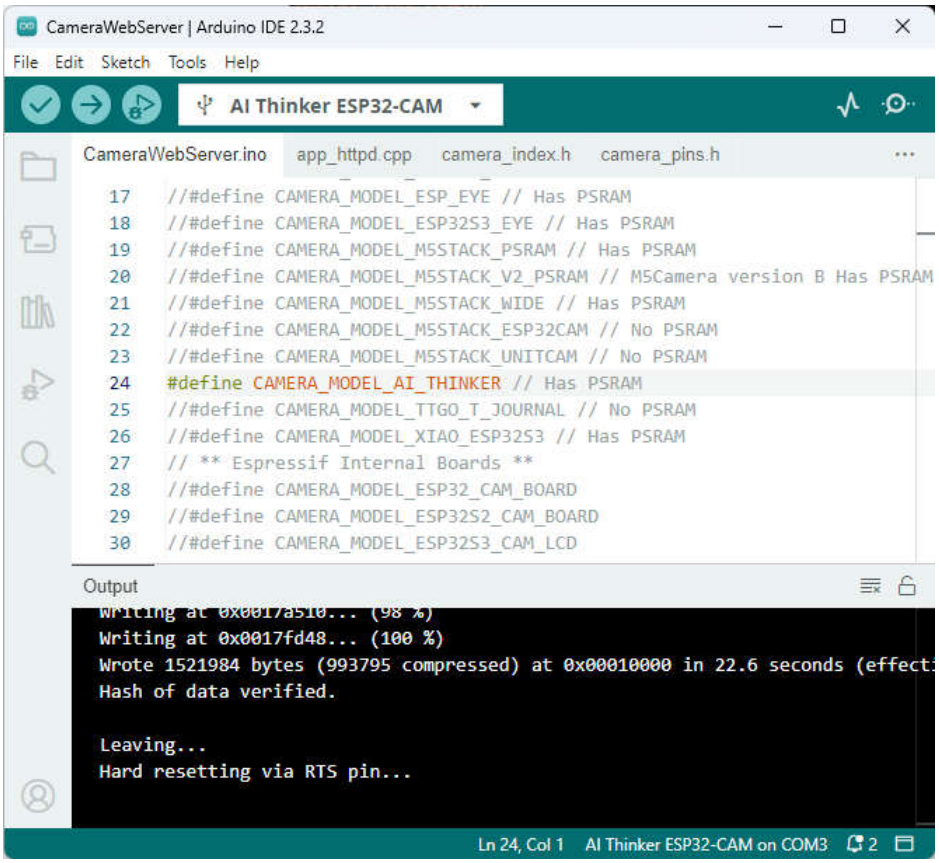
1) Lépjen az **Eszközök > Alaplap** elemre, és válassza az **AI-Thinker ESP32-CAM** lehetőséget.

2) Válassza az **Eszközök > Port** lehetőséget, és válassza ki azt a COM-portot, amelyhez az ESP32 csatlakozik.

3) Ezután kattintson a feltöltés gombra a kód feltöltéséhez: 

4) Ha a hibakereső ablakon néhány pont jelenik meg, előfordulhat, hogy meg kell nyomnod az ESP32-CAM fedélzeti RST gombját, ha az nem vált automatikusan flash módba.

Néhány másodperc múlva a kód sikeresen feltöltődik a táblára.

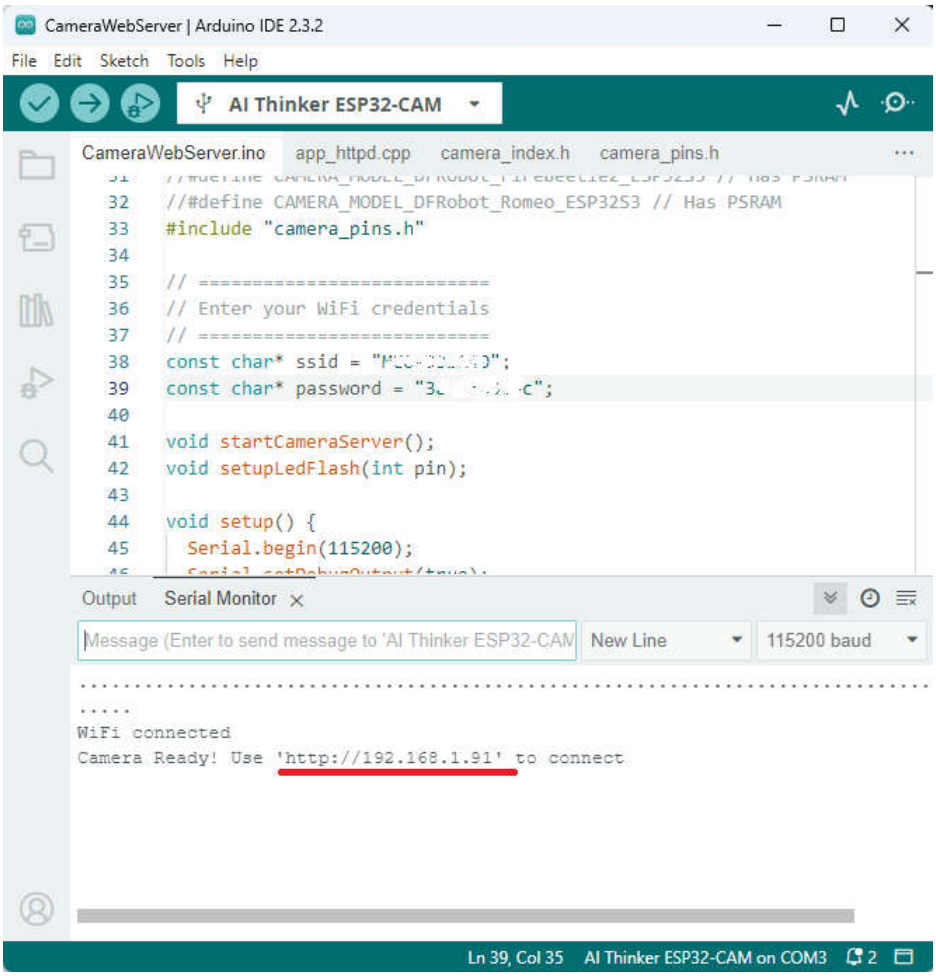


Az IP-cím megszerzése

A kód feltöltése után válaszd le a GPIO 0-t a GND-ről.

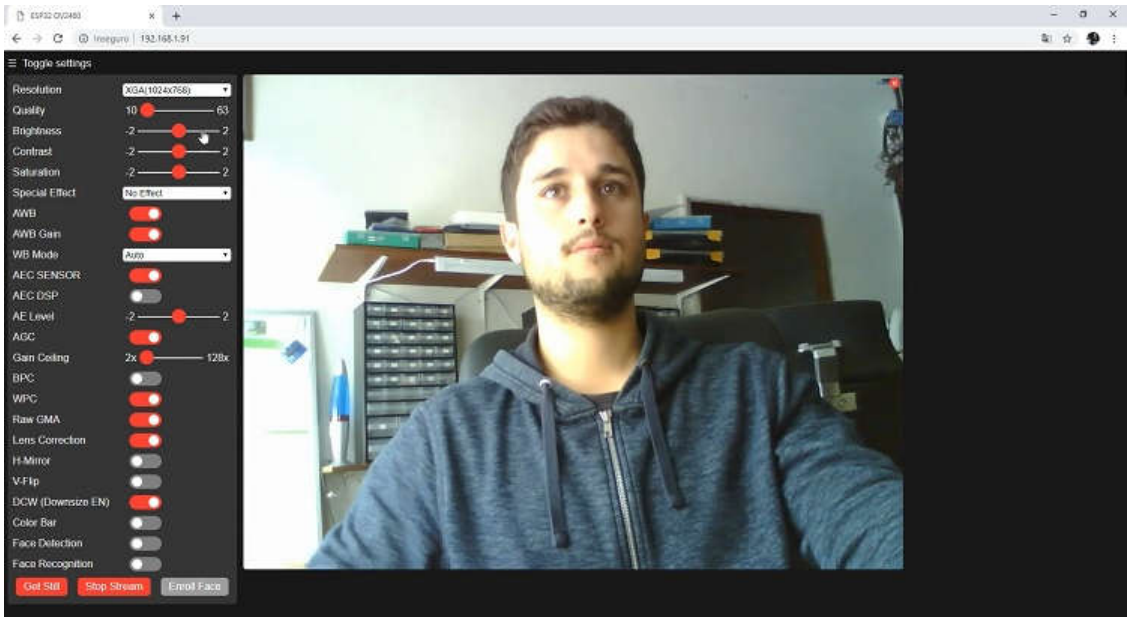
Nyisd meg a soros monitort 115200 adatátviteli sebességgel. Nyomd meg az ESP32-CAM alaplapi Reset gombot.

Az ESP32 IP-címe ki lesz nyomtatva a Soros monitorba.



A Video Streaming szerver elérése

Mostantól hozzáférhetsz a kamera streaming szerveréhez a helyi hálózaton. Nyiss meg egy böngészőt, és írd be az ESP32-CAM IP-címét. Nyomd meg a Streaming indítása gombot a videó streaming elindításához.

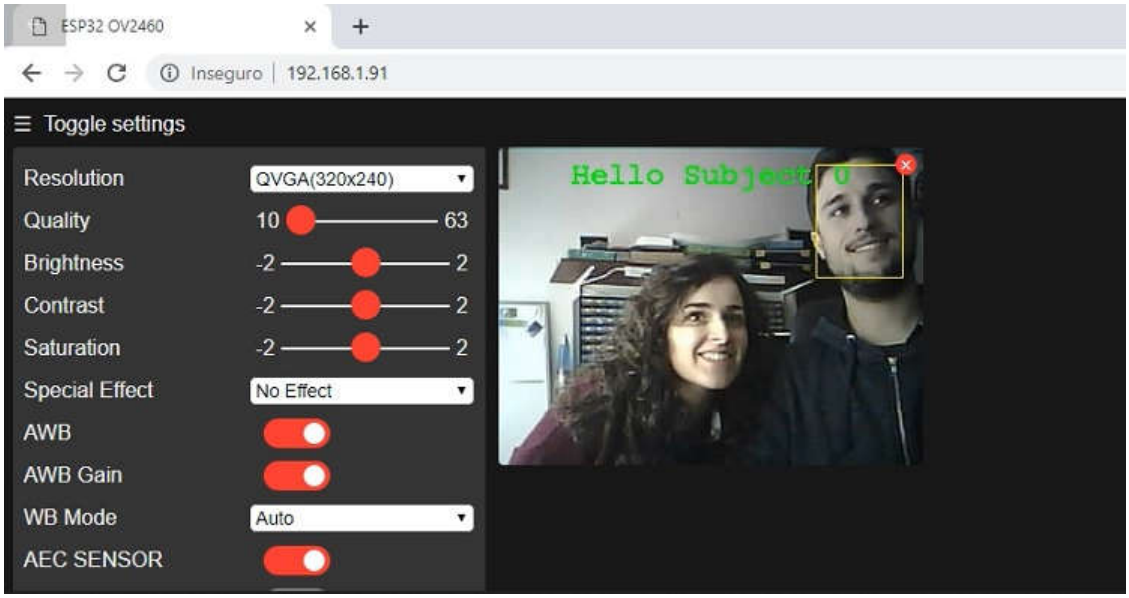


Lehetőséged van fényképeket is készíteni a Get Still gombra kattintva. Sajnos ez a példa nem menti a fényképeket, de módosíthatod úgy, hogy a beépített microSD kártyát használja a rögzített fényképek tárolására.

Számos kamerabeállítás is rendelkezésre áll, amelyekkel játszva módosíthatod a képbeállításokat.

Végül arc érzékelést és felismerést is végezhet.

Először is regisztrálnod kell egy új arcot. Többször megkísérli menteni az arcot. Új felhasználó regisztrálása után a későbbiekben észlelnie kell az arcot (0. alany).



És ennyi. Most már be van állítva a videó streaming webszervere, amely arc érzékeléssel és -felismeréssel rendelkezik a könyvtár példájával.

Hibaelhárítás

Ha a következő hibák bármelyikét észleled, olvasd el az ESP32-CAM hibaelhárítási útmutatónkat: A leggyakoribb problémák javítása.

- Nem sikerült csatlakozni az ESP32-hez: Időtúllépés a csomagfejlécre várakozva
- A kamera indítása meghiúsult 0x20001 vagy hasonló hiba miatt
- Brownout detektor vagy Guru meditációs hiba
- A vázlat túl nagy hiba – Rossz partíciós rendszer van kiválasztva
- A COMX-nél a tábla nem érhető el – a COM-port nincs kiválasztva
- Psram hiba: A GPIO isr szolgáltatás nincs telepítve
- Gyenge Wi-Fi jel
- Nincs IP-cím az Arduino IDE soros monitorban
- Nem lehet megnyitni a webszervert
- A kép késik/sok késést mutat

Becsomagolás

Az ESP32-CAM olcsó módot kínál fejlettebb otthoni automatizálási projektek készítésére, amelyek videót, fényképezést és arcfelismerést tartalmaznak.

Ebben az oktatóanyagban a CameraWebServer példát teszteltük, hogy teszteljük a kamera funkcióit. Most az az ötlet, hogy módosítsuk a példát, vagy írjunk egy teljesen új kódot más projektek felépítéséhez. Például készítsen fényképeket, és mentse el őket a microSD-kártyára, amikor a rendszer mozgást észlel, integrálja a videostreaminget otthoni automatizálási platformjába (például a Node-RED vagy a Home Assistant), és még sok más.

ESP32-CAM Video Streaming webszerver (az Otthon asszisztenssel működik)

Ebben a projektben egy IP megfigyelő kamerát építünk ESP32-CAM kártyával. Az ESP32 kamera egy video streaming webszerverhez fog csatlakozni, amelyet a hálózat bármely eszközével elérhetsz.

Ezt a video streaming webszervert integrálhatod olyan népszerű otthonautomatizálási platformokba, mint a Home Assistant vagy a Node-RED. Ebben az oktatóanyagban megmutatjuk, hogyan integrálhatod a Home Assistant és a Node-RED szolgáltatásba.

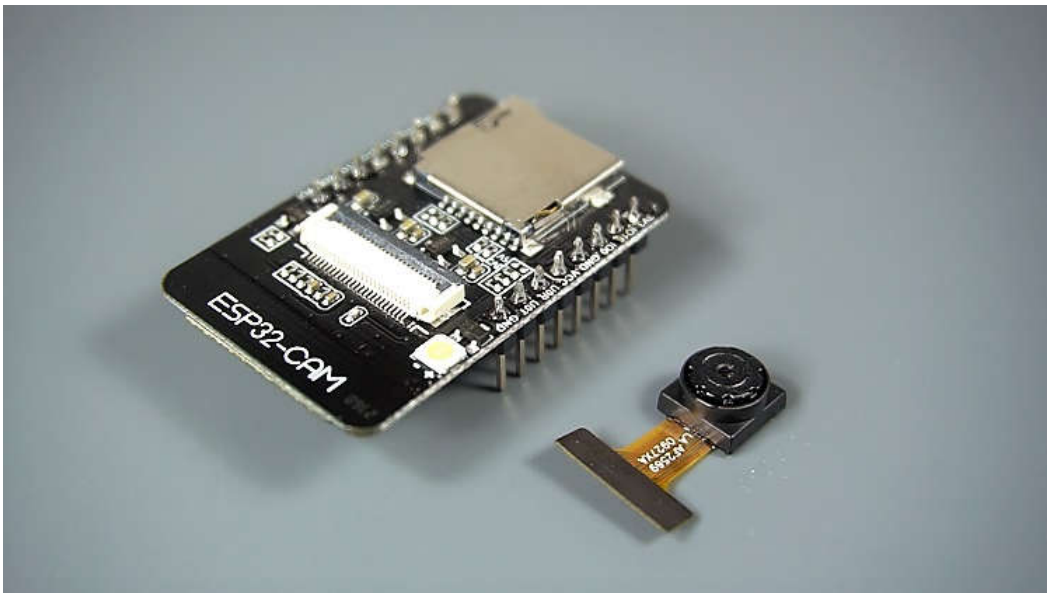
Szükséges alkatrészek

Az oktatóanyag követéséhez a következő alkatrészekre van szükséged:

- ESP32-CAM OV2640-el
- FTDI programozó
- Hüvelyes jumper vezetékek
- Hamis/ál gömb biztonsági kamera
- 5V-os tápegység az ESP32-CAM-hez
- Opcionális – Otthoni asszisztens Raspberry Pi-n:
    - Raspberry Pi alaplap
    - MicroSD kártya – 32 GB, Class10
    - Raspberry Pi tápegység (5V 2,5A)

Az ESP32-CAM bemutatása

Az ESP32-CAM egy nagyon kicsi kameramodul ESP32-S chippel, amely kevesebb, mint 10 dollárba kerül. Elolvashatod az ESP32-CAM kezdő lépései útmutatónkat, és megtanulhatod, hogyan kell használni a videostreamelést és az arcfelismerést.



Video Streaming szerver

Kövesd a következő lépéseket egy video streaming webszerver felépítéséhez az ESP32-CAM segítségével, amelyet a helyi hálózaton érhetsz el.

1. Telepítsd az ESP32 kiegészítőt

Ebben a példában az Arduino IDE-t használjuk az ESP32-CAM kártya programozásához. Tehát telepítened kell az Arduino IDE-t, valamint az ESP32 kiegészítőt. Az ESP32 bővítmény telepítéséhez kövesd a következő oktatóanyagok egyikét, ha még nem tetted meg:

- Az ESP32 kártya telepítése Arduino IDE-ben (Windows utasítások)
- Az ESP32 kártya telepítése Arduino IDE-ben (Mac és Linux utasítások)

2. Video streaming webszerver kódja

Ezt követően másold ki az alábbi kódot az Arduino IDE-be.

```
/*********
  Rui Santos
  A teljes projekt részletei: https://RandomNerdTutorials.com/esp32-cam-video-
streaming-web-server-camera-home-assistant/

  FONTOS!!!
   - Válaszd az "AI Thinker ESP32-CAM" alaplapot
   - A GPIO 0-t csatlakoztatni kell a GND-hez a vázlat feltöltése előtt
   - Miután csatlakoztattad a GPIO 0-t a GND-hez, nyomd meg az ESP32-CAM alaplapi
     RESET gombját, hogy a kártyát flash üzemmódba kapcsold.

  Ezúton ingyenes engedélyt adunk minden olyan személynek, aki másolatot
  szerez a szoftverről és a kapcsolódó dokumentációs fájlokról.

  A fenti szerzői jogi megjegyzést és ezt az engedélyt tartalmazó
  megjegyzést a Szoftver minden másolatának vagy jelentős részének
  tartalmaznia kell.
*********/
```

```
#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" // letiltja a barnulási problémákat
#include "soc/rtc_cntl_reg.h"  // letiltja a barnulási problémákat
#include "esp_http_server.h"

// Cseréld ki a hálózati hitelesítő adataival
const char* ssid = "CSERÉLD_KI_A_SAJÁT SSID-RE";
const char* password = "CSERÉLD_KI_A_SAJÁT_JELSZAVADRA ";

#define PART_BOUNDARY "123456789000000000000987654321"

//Ezt a projektet az AI Thinker modellel, az M5STACK PSRAM modellel és az M5STACK
WITHOUT PSRAM modellel tesztelték
#define CAMERA_MODEL_AI_THINKER
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM

// Ezzel a modellel nem tesztelték
//#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 "A kamera modellje nincs kiválasztva"
#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("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}

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

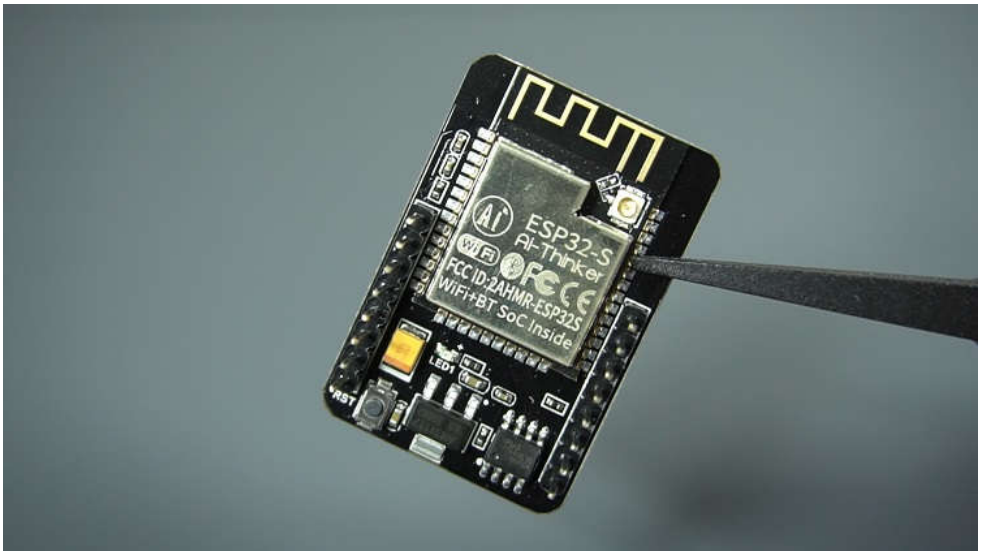
```
  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };

  //Serial.printf("Webszerver indítása a porton: '%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); // letiltja a barnulásérzékelőt

  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_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_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("A kamera init hiba miatt meghiúsult 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("Készen áll a kamerás adatfolyamra! Ugrás ide: http://");
  Serial.print(WiFi.localIP());

  // A webszerver adatfolyamának indítása
  startCameraServer();
}

void loop() {
  delay(1);
}
```

A kód feltöltése előtt be kell illesztenie hálózati hitelesítő adatait a következő változókba:

```
const char* ssid = "CSERÉLD_KI_A_SAJÁT SSID-RE";
const char* password = "CSERÉLD_KI_A_SAJÁT_JELSZAVADRA ";
```

Ezután győződj meg arról, hogy a megfelelő kameramodult választottad. Ebben az esetben az AI-THINKER modellt használjuk.



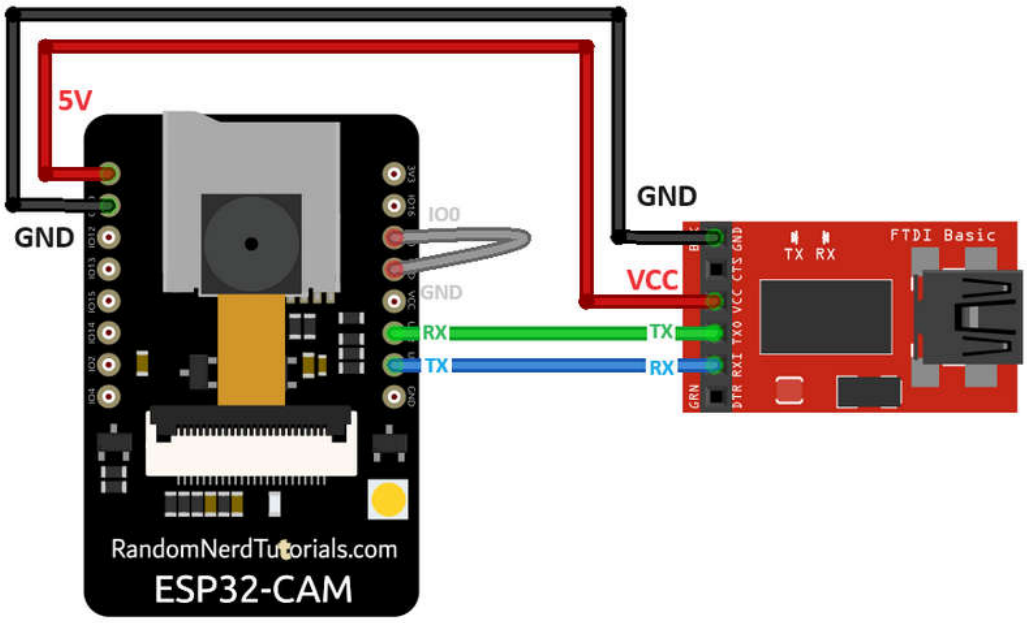
Ha ugyanazt a kameramodult használja, nem kell semmit módosítania a kódon.

```
#define CAMERA_MODEL_AI_THINKER
```

Most feltöltheted a kódot az ESP32-CAM kártyára.

3. A kód feltöltése

Csatlakoztasd az ESP32-CAM kártyát a számítógépéhez egy FTDI programozó segítségével. Kövesd a következő sematikus diagramot:



Sok FTDI programozó rendelkezik jumperrel, amely lehetővé teszi a 3,3 V vagy az 5 V kiválasztását. Győződj meg arról, hogy a jumper a megfelelő helyen van az 5V kiválasztásához.

Fontos: A GPIO 0-nak csatlakoznia kell a GND-hez, hogy fel tudd tölteni a kódot.

ESP32-CAM	FTDI Programmer
GND	GND
5V	VCC (5V)
U0R	TX
U0T	RX
GPIO 0	GND

A kód feltöltéséhez kövesd a következő lépéseket:

1) Lépjen az **Eszközök > Alaplap** elemre, és válassza az **AI-Thinker ESP32-CAM** lehetőséget.

2) Válassza az **Eszközök > Port** lehetőséget, és válassza ki azt a COM-portot, amelyhez az ESP32 csatlakozik.

3) Ezután kattintson a feltöltés gombra a kód feltöltéséhez: 

4) Ha a hibakereső ablakon néhány pont jelenik meg, előfordulhat, hogy meg kell nyomnod az ESP32-CAM fedélzeti RST gombját, ha az nem vált automatikusan flash módba.

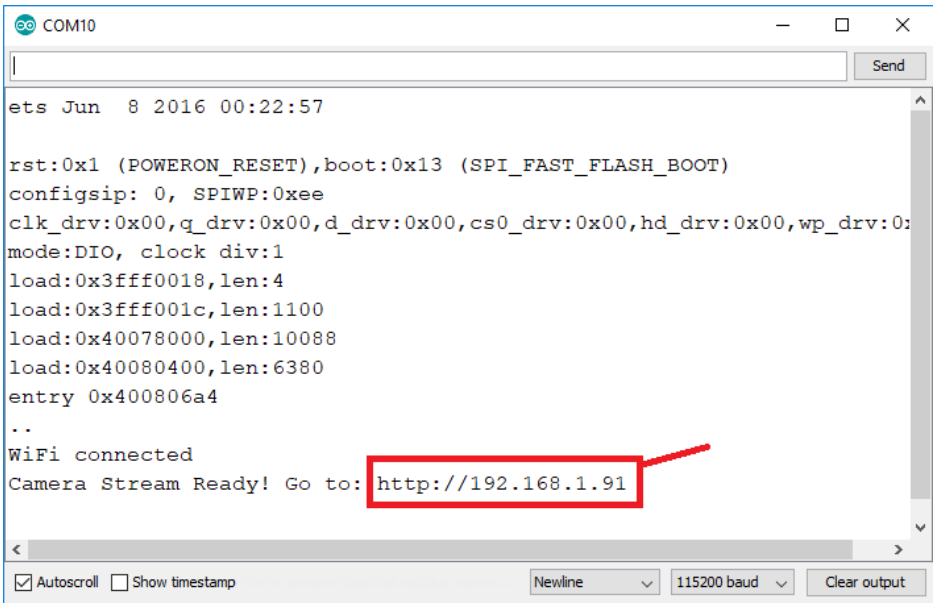
Néhány másodperc múlva a kód sikeresen feltöltődik a táblára.

Az IP-cím megszerzése

A kód feltöltése után válaszd le a GPIO 0-t a GND-ről.

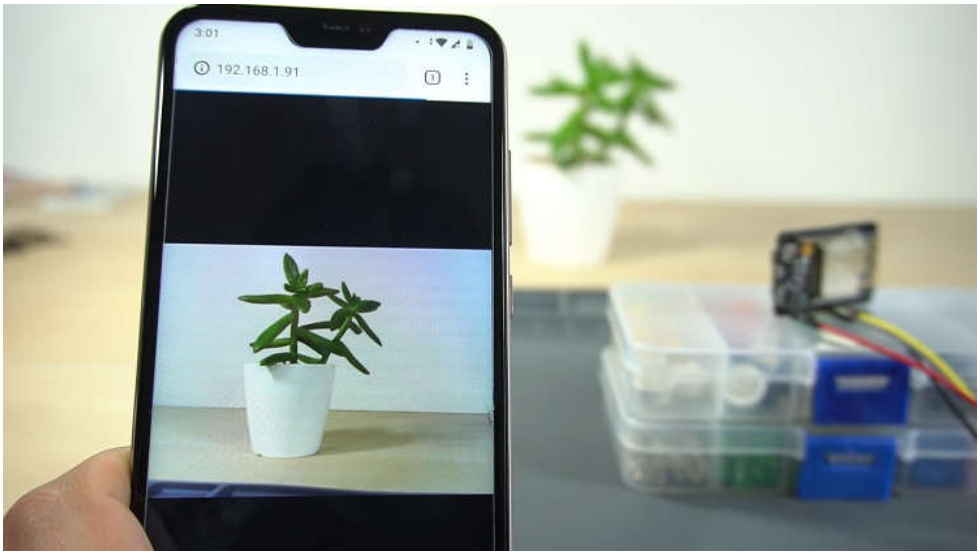
Nyisd meg a soros monitort 115200 adatátviteli sebességgel. Nyomd meg az ESP32-CAM alaplapi Reset gombot.

Az ESP32 IP-címe ki lesz nyomtatva a Soros monitorba.



A Video Streaming szerver elérése

Mostantól hozzáférhetsz a kamera streaming szerveréhez a helyi hálózaton. Nyiss meg egy böngészőt, és írd be az ESP32-CAM IP-címét. Egy oldalnak be kell töltenie az aktuális videofolyamot.



Otthon asszisztens integráció

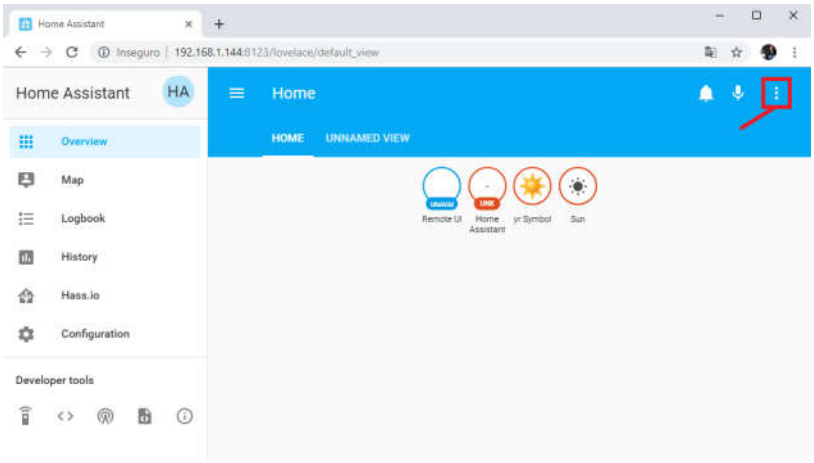
A legtöbb ember számára hasznos lehet, ha csak az ESP32-CAM működik IP-n keresztül, de ezt a projektet integrálhatja az Otthon asszisztenssel (vagy más otthoni automatizálási platformokkal). Folytasd az olvasást, hogy megtudd, hogyan integrálhatod az Otthon asszisztensbe.

Előfeltételek

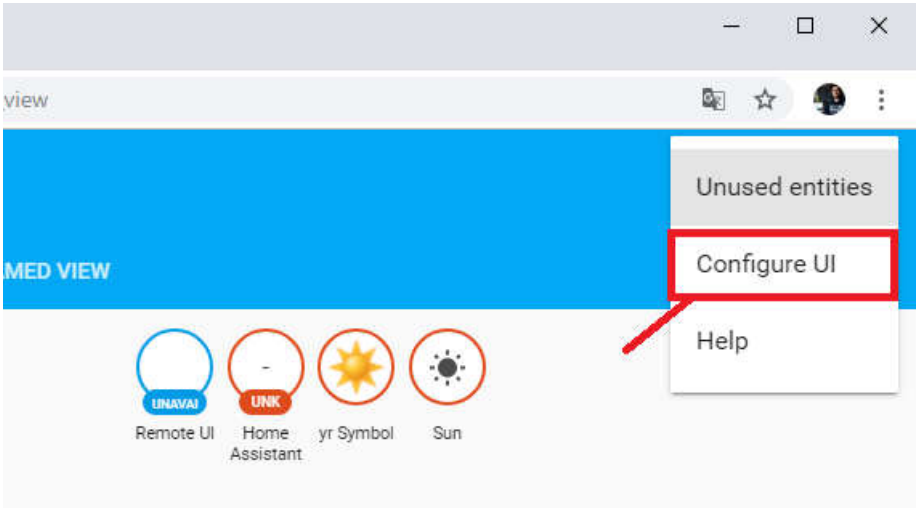
- Ismerned kell a Raspberry Pi-t – olvasd el a Raspberry Pi használatának megkezdését.
- Kezdő lépések a Raspberry Pi Otthon asszisztensével

ESP32-CAM hozzáadása az Otthon asszisztenshez

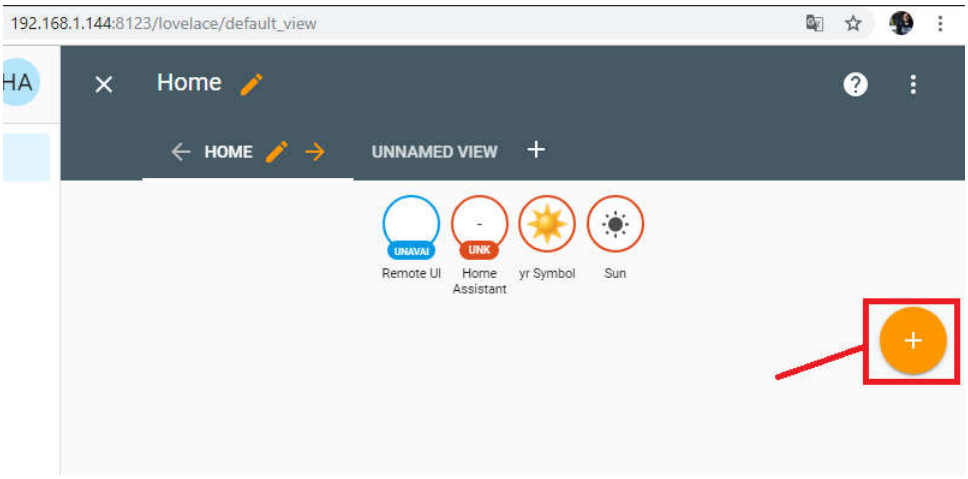
Nyisd meg a Home Assistant irányítópultját, és lépj a **Settings** menübe.



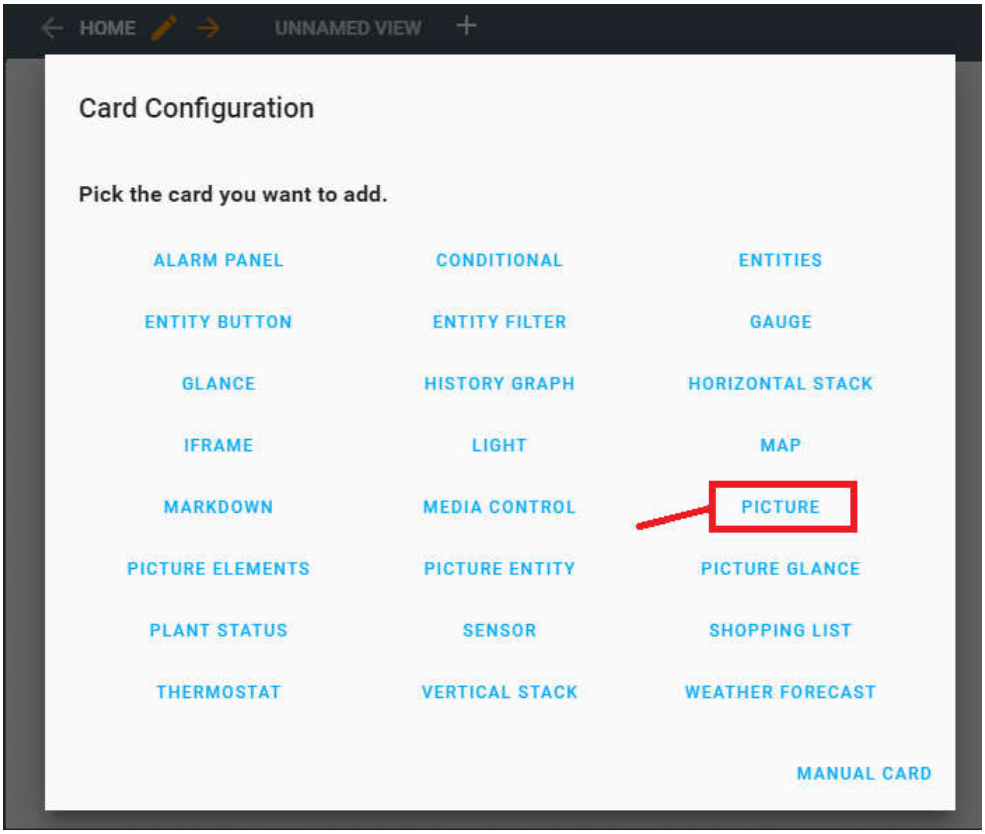
Nyisd meg a **Configure UI**-t



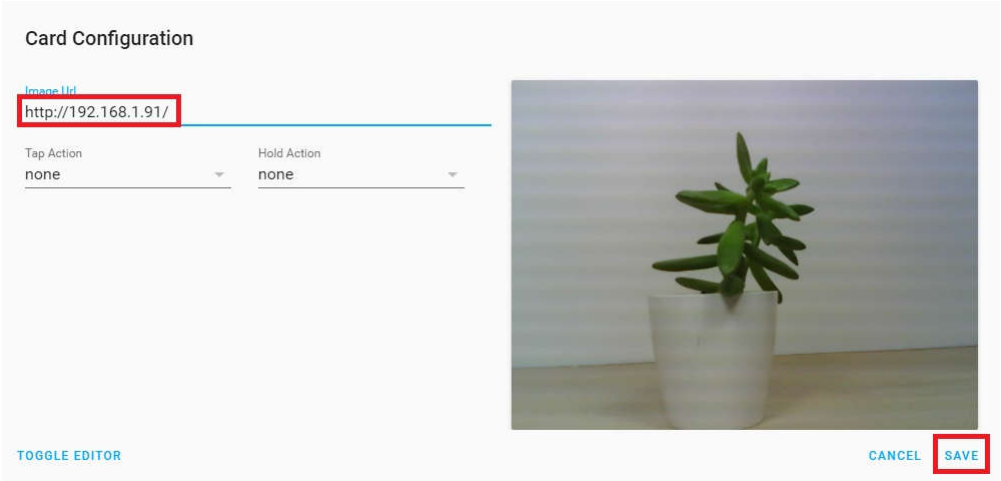
Új kártya hozzáadása az irányítópulthoz:



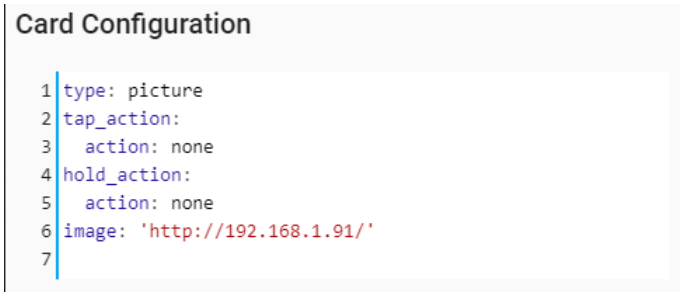
Válassz egy **Picture** típusú kártyát.



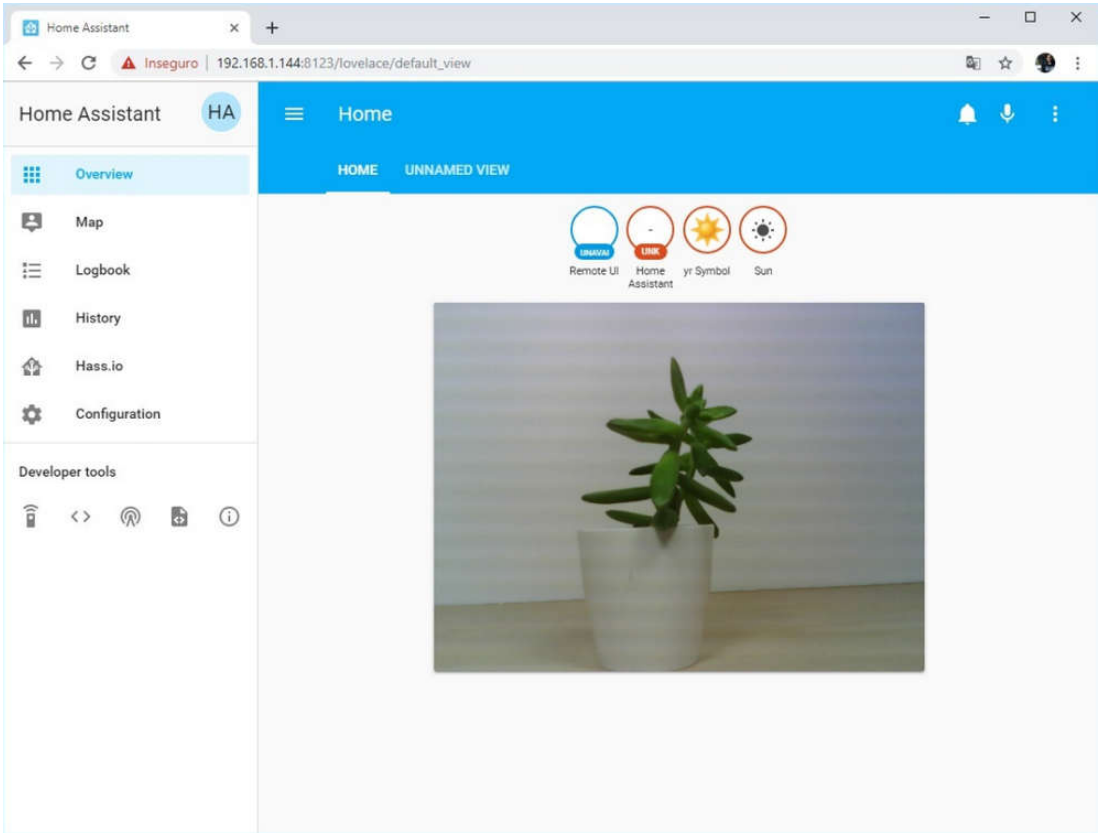
A Kép URL mezőbe írd be az ESP32-CAM IP-címét. Ezután kattints a „SAVE” gombra, és térj vissza a fő műszerfalra.



Ha a konfigurációs fájlt használod, ezt kell hozzáadnod.



Ezt követően a Home Assistant megjeleníti az ESP32-CAM video streaminget.

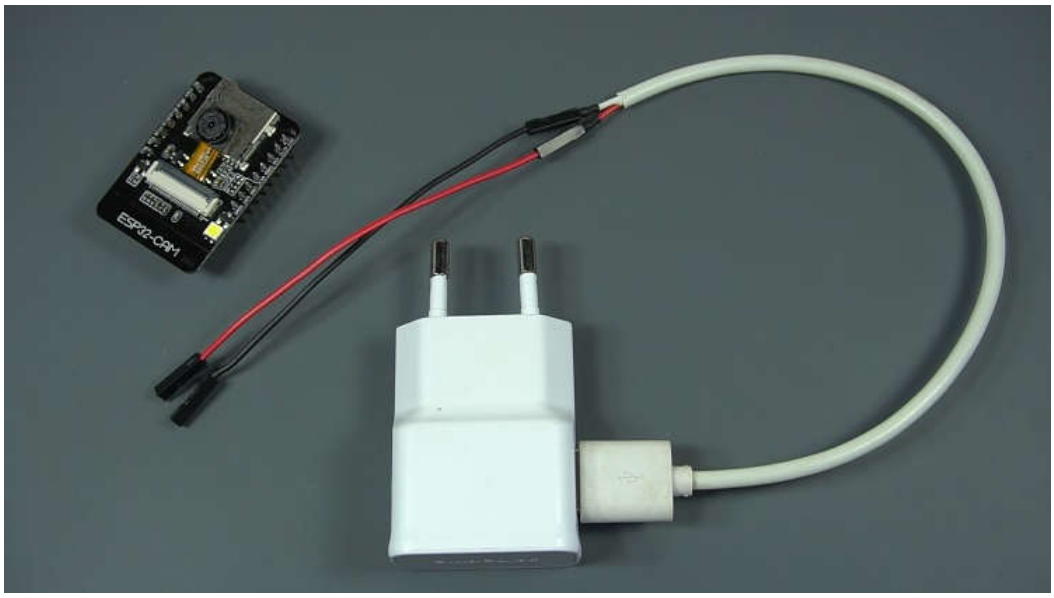


Tovább vive

A projekt továbbviteléhez használhatsz egy hamis álkamerát, és helyezd bele az ESP32-CAM-et.

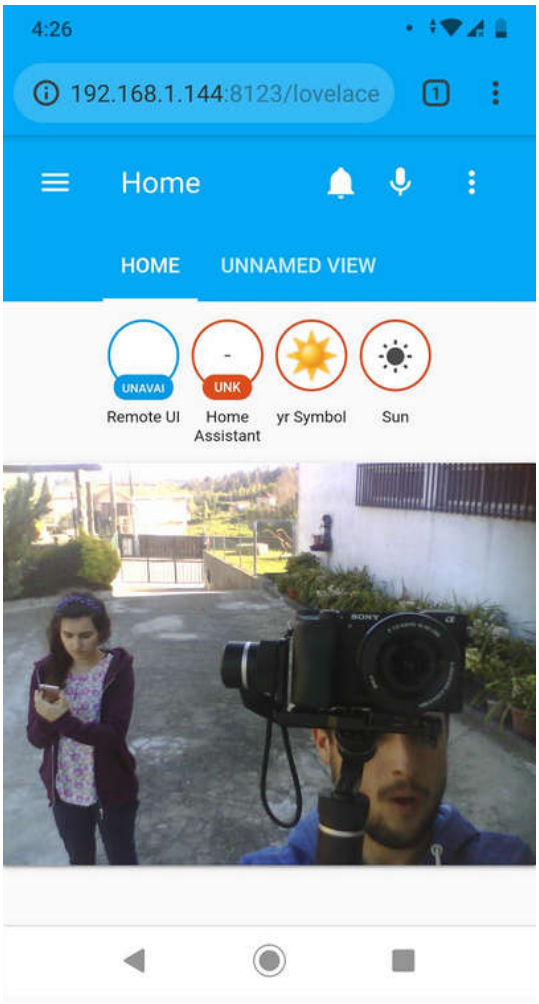
Az ESP32-CAM kártya tökéletesen illeszkedik a vakkamera házába.

5 V-os hálózati adapterrel táplálhatja az ESP32-CAM-ot GND és 5 V-os érintkezőkön keresztül.

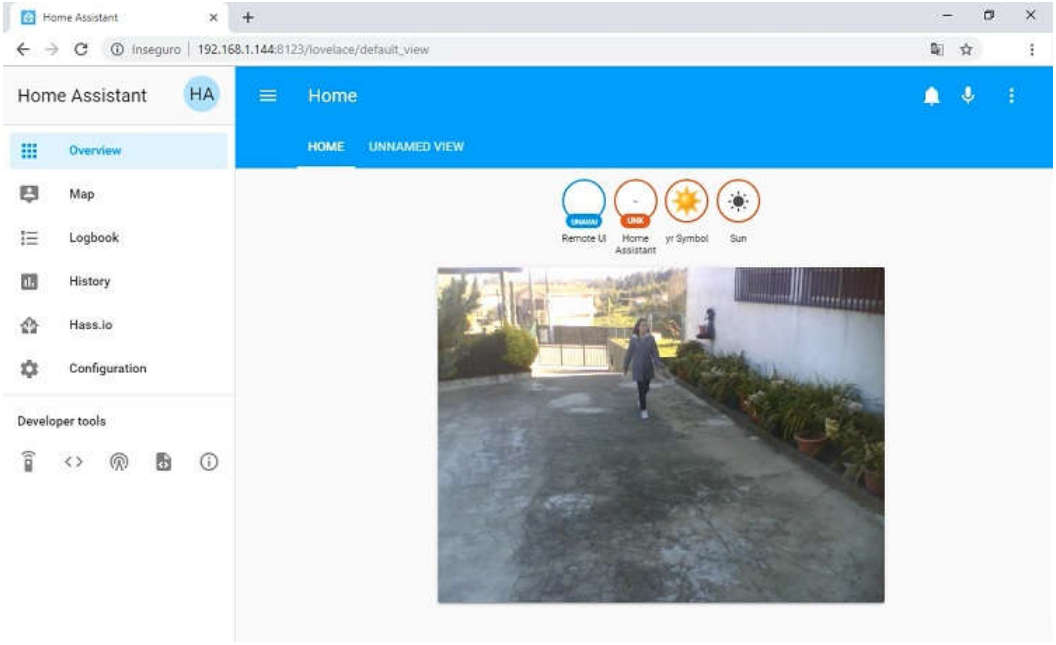


Helyezd a térfigyelő kamerát megfelelő helyre.

Ezután lépj a kamera IP-címére vagy a Home Assistant irányítópultjára, és valós időben nézze meg, mi történik. A következő képen a videó streaming kamera tesztelése látható. Sara képernyőképet készít, miközben én forgatom a kamerát.



Lenyűgöző, mire képes ez a kis 9 dolláros ESP32 kameramodul, és megbízhatóan működik. Mostantól a térfigyelő kamera segítségével valós időben láthatjuk, mi történik a bejáratomban.



Tipp: Node-RED integráció

A video streaming webszerver a Node-RED és a Node-RED Dashboard rendszerbe is integrálható. Csak létre kell hoznod egy sablon csomópontot, és hozzá kell adnod a következőket:

```
<div style="margin-bottom: 10px;">
<img src="https://YOUR-ESP32-CAM-IP-ADDRESS" width="650px">
</div>
```

Az *src* attribútumban meg kell adnod az ESP32-CAM IP-címét:

```
<div style="margin-bottom: 10px;">
<img src="https://192.168.1.91" width="650px">
</div>
```

Hibaelhárítás

Ha a következő hibák bármelyikét észleli, olvassa el az ESP32-CAM hibaelhárítási útmutatónkat: A leggyakoribb problémák javítása

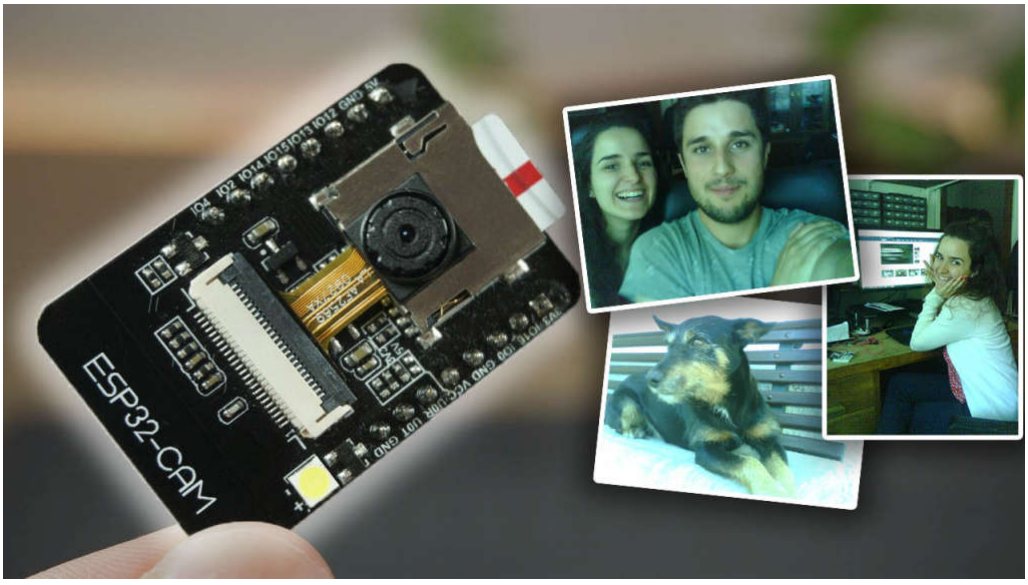
- Nem sikerült csatlakozni az ESP32-hez: Időtúllépés a csomagfejlécre várakozva
- A kamera indítása meghiúsult 0x20001 vagy hasonló hiba miatt
- Brownout detektor vagy Guru meditációs hiba
- A vázlat túl nagy hiba – Rossz partíciós rendszer van kiválasztva
- A COMX-nél a tábla nem érhető el – a COM-port nincs kiválasztva
- Psram hiba: A GPIO isr szolgáltatás nincs telepítve
- Gyenge Wi-Fi jel
- Nincs IP-cím az Arduino IDE soros monitorban
- Nem lehet megnyitni a webszervert
- A kép késik/sok késést mutat

Becsomagolás

Ebben az oktatóanyagban megmutattuk, hogyan építhetsz fel egy egyszerű video streaming webszervert az ESP32-CAM kártyával IP-kamera építéséhez. Az általunk épített webszerver könnyen integrálható otthoni automatizálási platformjával, mint például a Node-RED vagy a Home Assistanttal.

ESP32-CAM Fénykép készítése és mentése MicroSD kártyára

Ismerd meg, hogyan készíthetsz fényképeket az ESP32-CAM kártyával, és mentheted azokat microSD-kártyára az Arduino IDE segítségével. Ha megnyomod az ESP32-CAM RESET gombját, felébred, fényképet készít, és elmenti a microSD kártyára.



Az AI-Thinker modulként megjelölt ESP32-CAM kártyát fogjuk használni, de más moduloknak is működniük kell úgy, hogy a kódban a megfelelő pin-hozzárendelést megadják.

Az ESP32-CAM kártya egy 9 dolláros (vagy kevesebb) eszköz, amely egy ESP32-S chipet, egy OV2640 kamerát, egy microSD kártyanyílást és több GPIO érintkezőt kombinál.

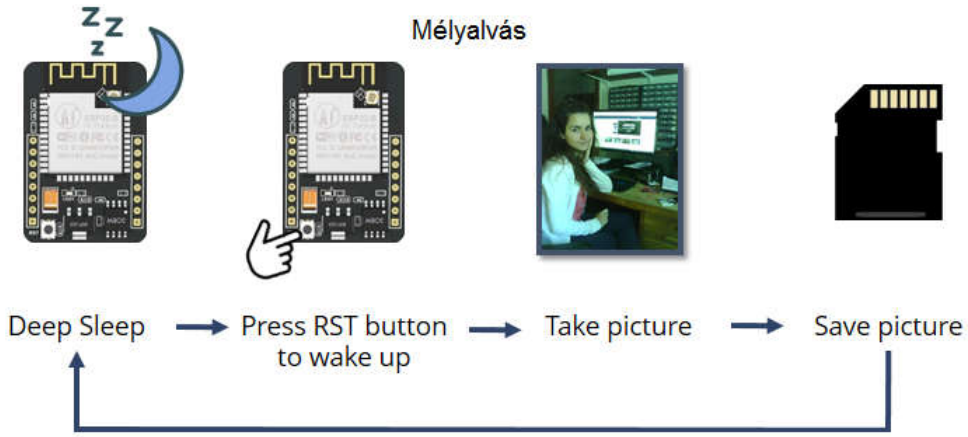
Szükséges alkatrészek

Az oktatóanyag követéséhez a következő alkatrészekre van szükséged:

- ESP32-CAM OV2640-el
- Mikro SD kártya
- FTDI programozó
- Hüvelyes jumper vezetékek
- 5V-os tápegység az ESP32-CAM-hez

A projekt áttekintése

Íme egy gyors áttekintés a projekt működéséről.



Az ESP32-CAM mély alvó módban van

Megnyomjuk a RESET gombot a tábla felébresztéséhez

A kamera fényképet készít

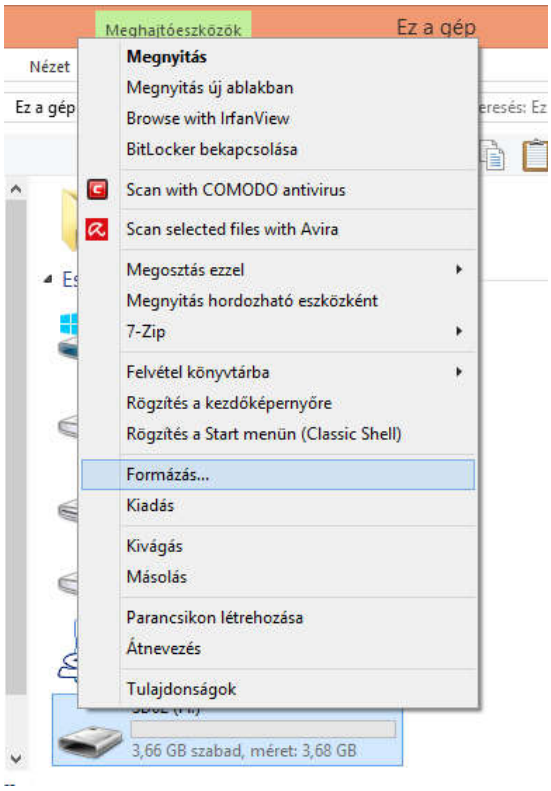
A fényképet a microSD kártyára menti a rendszer a következő néven: pictureX.jpg, ahol az X a képszámnak felel meg

A képszám az ESP32 flash memóriájába kerül, így a RESET során nem törlődik, és nyomon tudjuk követni a készített fényképek számát.

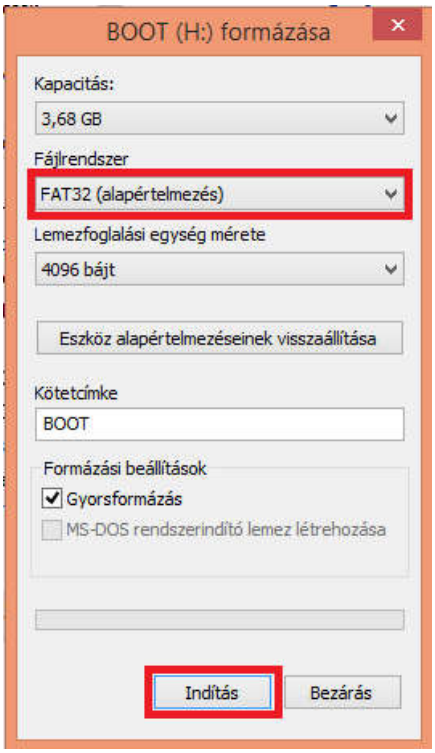
A MicroSD kártya formázása

Az első dolog, amit javasolunk, a microSD-kártya formázása. Használhatod a Windows formázó eszközt vagy bármilyen más microSD formázó szoftvert.

1. Helyezd be a microSD-kártyát a számítógépbe. Lépj az **Ez a gép** elemre, és kattints a jobb gombbal az SD-kártyára. Válaszd a **Formázás** lehetőséget az alábbi ábrán látható módon.



2. Egy új ablak jelenik meg. Válaszd ki a FAT32-t, nyomd meg az **Indítás** gombot a formázási folyamat inicializálásához, és kövesd a képernyőn megjelenő utasításokat.



Megjegyzés: a termék specifikációi szerint az ESP32-CAM csak 4 GB-os SD-kártyákat támogat. Azonban 16 GB-os SD-kártyával teszteltük, és jól működött.

Az ESP32 kiegészítő telepítése

Az ESP32 kártyát Arduino IDE segítségével programozzuk. Tehát telepítened kell az Arduino IDE-t, valamint az ESP32 kiegészítőt. Kövesd a következő oktatóanyagot az ESP32 bővítmény telepítéséhez, ha még nem tetted meg:

- Az ESP32 tábla telepítése Arduino IDE 2-ben (Windows, Mac OS X, Linux)

Fotó készítésének és mentésének vázlata

Másold a következő kódot az Arduino IDE-be.

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp32-cam-take-photo-save-microsd-card

  IMPORTANT!!!
   - Select Board "AI Thinker ESP32-CAM"
   - GPIO 0 must be connected to GND to upload a sketch
   - After connecting GPIO 0 to GND, press the ESP32-CAM on-board RESET button to put your board in flashing mode

  Permission is hereby granted, free of charge, to any person obtaining a copy
  of this software and associated documentation files.
  The above copyright notice and this permission notice shall be included in all
  copies or substantial portions of the Software.
*********/

#include "esp_camera.h"
#include "Arduino.h"
#include "FS.h"                // SD Card ESP32
#include "SD_MMC.h"            // SD Card ESP32
#include "soc/soc.h"           // Disable brownour problems
#include "soc/rtc_cntl_reg.h"  // Disable brownour problems
#include "driver/rtc_io.h"
#include <EEPROM.h>            // read and write from flash memory

// define the number of bytes you want to access
#define EEPROM_SIZE 1

// Pin definition for 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

int pictureNumber = 0;

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

  Serial.begin(115200);
  //Serial.setDebugOutput(true);
  //Serial.println();

  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_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_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; // FRAMESIZE_ +
QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

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

  //Serial.println("Starting SD Card");
  if(!SD_MMC.begin()){
    Serial.println("SD Card Mount Failed");
    return;
  }
```

```
  uint8_t cardType = SD_MMC.cardType();
  if(cardType == CARD_NONE){
    Serial.println("No SD Card attached");
    return;
  }

  camera_fb_t * fb = NULL;

  // Take Picture with Camera
  fb = esp_camera_fb_get();
  if(!fb) {
    Serial.println("Camera capture failed");
    return;
  }
  // initialize EEPROM with predefined size
  EEPROM.begin(EEPROM_SIZE);
  pictureNumber = EEPROM.read(0) + 1;

  // Path where new picture will be saved in SD Card
  String path = "/picture" + String(pictureNumber) +".jpg";

  fs::FS &fs = SD_MMC;
  Serial.printf("Picture file name: %s\n", path.c_str());

  File file = fs.open(path.c_str(), FILE_WRITE);
  if(!file){
    Serial.println("Failed to open file in writing mode");
  }
  else {
    file.write(fb->buf, fb->len); // payload (image), payload length
    Serial.printf("Saved file to path: %s\n", path.c_str());
    EEPROM.write(0, pictureNumber);
    EEPROM.commit();
  }
  file.close();
  esp_camera_fb_return(fb);

  // Turns off the ESP32-CAM white on-board LED (flash) connected to GPIO 4
  pinMode(4, OUTPUT);
  digitalWrite(4, LOW);
  rtc_gpio_hold_en(GPIO_NUM_4);

  delay(2000);
  Serial.println("Going to sleep now");
  delay(2000);
  esp_deep_sleep_start();
  Serial.println("This will never be printed");
}

void loop() {

}
```

A kód azzal kezdődik, hogy befoglaljuk a kamera használatához szükséges könyvtárakat. Befoglaljuk a microSD-kártyával való interakcióhoz szükséges könyvtárakat is:

```
#include "esp_camera.h"
#include "Arduino.h"
#include "FS.h"                // SD Card ESP32
```

```
#include "SD_MMC.h"            // SD Card ESP32
#include "soc/soc.h"           // Disable brownour problems
#include "soc/rtc_cntl_reg.h"  // Disable brownour problems
#include "driver/rtc_io.h"
#include <EEPROM.h>            // read and write from flash memory
```

És az EEPROM könyvtárat az állandó adatok flash memóriába való mentéséhez.

```
#include <EEPROM.h>
```

Ha többet szeretnél megtudni arról, hogyan kell adatokat olvasni és írni a flash memóriába, kövesd a következő oktatóanyagot:

- ESP32 Flash memória – Állandó adatok tárolása (írás és olvasás)

Határozzuk meg a flash memóriában elérni kívánt bájtok számát. Itt csak egy bájtot használunk, amely lehetővé teszi akár 256 képszám generálását.

```
#define EEPROM_SIZE 1
```

Ezután határozza meg az AI-THINKER kameramodul érintkezőit.

```
// Pin definition for 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
```

Megjegyzés: előfordulhat, hogy módosítanod kell a tű meghatározását a használt kártyától függően. A helytelen tű-kiosztás a kamera hibás beindítását eredményezi.

Inicializáljunk egy `pictureNumber` nevű int változót, amely létrehozza a fotó nevét: kép1.jpg, kép2.jpg és így tovább.

```
int pictureNumber = 0;
```

Minden kódunk a `setup()`-ban van. A kód csak egyszer fut le, amikor az ESP32 felébred (ebben az esetben a fedélzeti RESET gomb megnyomásakor).

Megadjuk a kamera beállításait:

```
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;
```

Használjuk a következő beállításokat egy PSRAM-mal rendelkező fényképezőgéphez (mint az ebben az oktatóanyagban használt).

```
if(psramFound()){
  config.frame_size = FRAMESIZE_UXGA; // FRAMESIZE_ +
QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA
  config.jpeg_quality = 10;
  config.fb_count = 2;
}
```

Ha az alaplapon nincs PSRAM, állítsuk be a következőket:

```
else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
}
```

A kamera inicializálása:

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

A microSD kártya inicializálása:

```
//Serial.println("Starting SD Card");
if(!SD_MMC.begin()){
  Serial.println("SD Card Mount Failed");
  return;
}

uint8_t cardType = SD_MMC.cardType();
```

```
if(cardType == CARD_NONE){
  Serial.println("No SD Card attached");
  return;
}
```

A microSD kártya használatáról további információkat a következő projektben találsz:

- ESP32 hőmérséklet adatnaplózása a MicroSD kártyára

A következő sorok készítenek fényképet a fényképezőgéppel:

```
camera_fb_t * fb = NULL;

// Take Picture with Camera
fb = esp_camera_fb_get();
if(!fb) {
  Serial.println("Camera capture failed");
  return;
}
```

Ezután inicializáljuk az EEPROM-ot a korábban meghatározott mérettel:

```
EEPROM.begin(EEPROM_SIZE);
```

A képszámot úgy állítjuk elő, hogy 1-et adunk a flash memóriában tárolt aktuális számhoz.

```
pictureNumber = EEPROM.read(0) + 1;
```

A fénykép microSD-kártyára mentéséhez hozzunk létre egy elérési utat a fájlhoz. A fényképet elmentjük a microSD kártya főkönyvtárába, és a fájl neve (kép1.jpg, kép2.jpg, kép3.jpg stb…) lesz.

```
String path = "/picture" + String(pictureNumber) +".jpg";
```

A következő sorok mentik a fényképet a microSD-kártyára:

```
fs::FS &fs = SD_MMC;
Serial.printf("Picture file name: %s\n", path.c_str());

File file = fs.open(path.c_str(), FILE_WRITE);
if(!file){
  Serial.println("Failed to open file in writing mode");
}
else {
  file.write(fb->buf, fb->len); // payload (image), payload length
  Serial.printf("Saved file to path: %s\n", path.c_str());
  EEPROM.write(0, pictureNumber);
  EEPROM.commit();
}
file.close();
```

A fénykép mentése után az aktuális képszámot elmentjük a flash memóriába, hogy nyomon követhessük a készített fényképek számát.

```
EEPROM.write(0, pictureNumber);
EEPROM.commit();
```

Amikor az ESP32-CAM fényképet készít, villog a beépített LED. A fénykép elkészítése után a LED égve marad, ezért küldünk utasítást a kikapcsolásra. A LED a GPIO 4-hez csatlakozik.

```
pinMode(4, OUTPUT);
digitalWrite(4, LOW);
rtc_gpio_hold_en(GPIO_NUM_4);
```

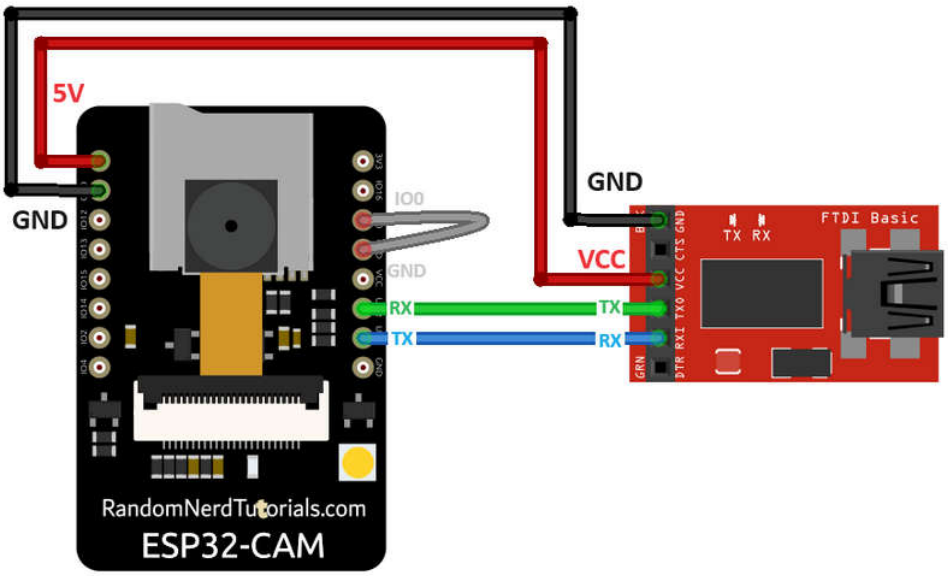
Végül mély álomba helyeztük az ESP32-t.

```
esp_deep_sleep_start();
```

Mivel nem adunk át argumentumot a mélyalvás funkciónak, az ESP32 kártya korlátlan ideig aludni fog a RESET-ig.

ESP32-CAM kód feltöltése

Ha kódot szeretnél feltölteni az ESP32-CAM kártyára, csatlakoztasd azt a számítógépéhez egy FTDI programozó segítségével. Kövesd a következő sematikus diagramot:



Sok FTDI programozó rendelkezik jumperrel, amely lehetővé teszi a 3,3 V vagy az 5 V kiválasztását. Győződj meg arról, hogy a jumper a megfelelő helyen van az 5V kiválasztásához.

Fontos: A GPIO 0-nak csatlakoznia kell a GND-hez, hogy fel tudd tölteni a kódot.

ESP32-CAM	FTDI Programmer
GND	GND
5V	VCC (5V)
U0R	TX
U0T	RX
GPIO 0	GND

A kód feltöltéséhez kövesd a következő lépéseket:

1) Lépjen az **Eszközök > Alaplap** elemre, és válassza az **AI-Thinker ESP32-CAM** lehetőséget.

2) Válassza az **Eszközök > Port** lehetőséget, és válassza ki azt a COM-portot, amelyhez az ESP32 csatlakozik.

3) Ezután kattintson a feltöltés gombra a kód feltöltéséhez: 

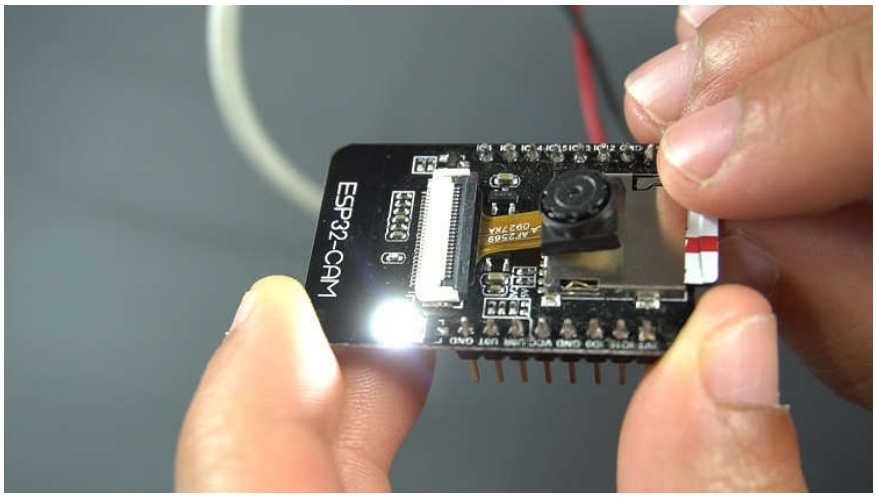
4) Ha a hibakereső ablakon néhány pont jelenik meg, előfordulhat, hogy meg kell nyomnod az ESP32-CAM fedélzeti RST gombját, ha az nem vált automatikusan flash módba.

Néhány másodperc múlva a kód sikeresen feltöltődik a táblára.

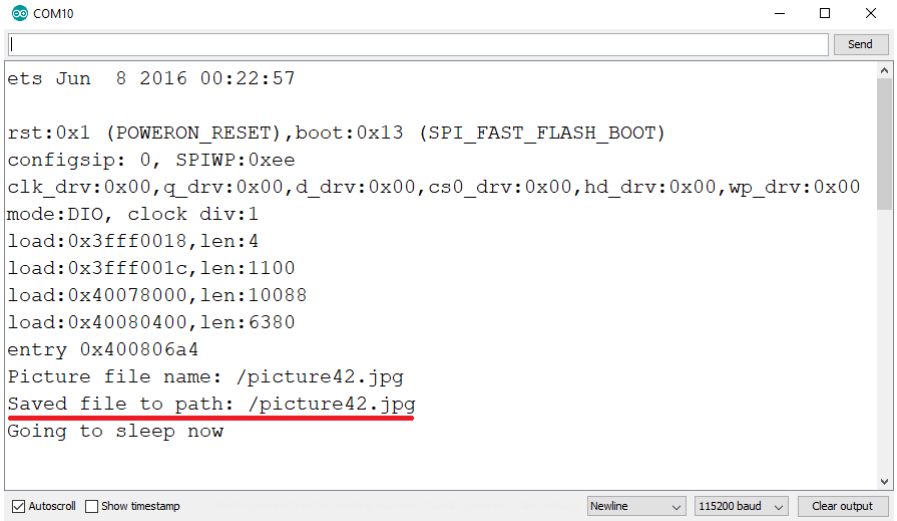
Demonstráció

A kód feltöltése után távolítsd el a GPIO 0-t a GND-ről összekötő jumpert.

Nyisd meg a soros monitort 115200 adatátviteli sebességgel. Nyomd meg az ESP32-CAM reset gombot. Inicializálódnia kell, és fényképet kell készítenie. Fényképezéskor bekapcsolja a vakut (GPIO 4).



Ellenőrizze az Arduino IDE Serial Monitor ablakában, hogy minden a várt módon működik-e. Mint látható, a kép sikeresen mentve a microSD kártyára.



Megjegyzés: ha problémái vannak az ESP32-CAM-mel, tekintse meg hibaelhárítási útmutatónkat, és nézze meg, segít-e: ESP32-CAM hibaelhárítási útmutató: A leggyakoribb problémák javítása

Miután megbizonyosodtál arról, hogy minden a várt módon működik, leválaszthatod az ESP32-CAM-ot az FTDI programozóról, és egy független tápegység segítségével táplálhatod.

Az elkészített fényképek megtekintéséhez vedd ki a microSD-kártyát a microSD-kártyanyílásból, és helyezd be a számítógépbe. Az összes fényképet el kell mentened.

A fénykép minősége a fényviszonyoktól függ. A túl sok fény tönkreteheti a fényképeket, a sötét környezet pedig sok fekete képpontot eredményez.

Hibaelhárítás

Ha a következő hibák bármelyikét észleli, olvassa el az ESP32-CAM hibaelhárítási útmutatónkat: A leggyakoribb problémák javítása

- Nem sikerült csatlakozni az ESP32-hez: Időtúllépés a csomagfejlécre várakozva
- A kamera indítása meghiúsult 0x20001 vagy hasonló hiba miatt

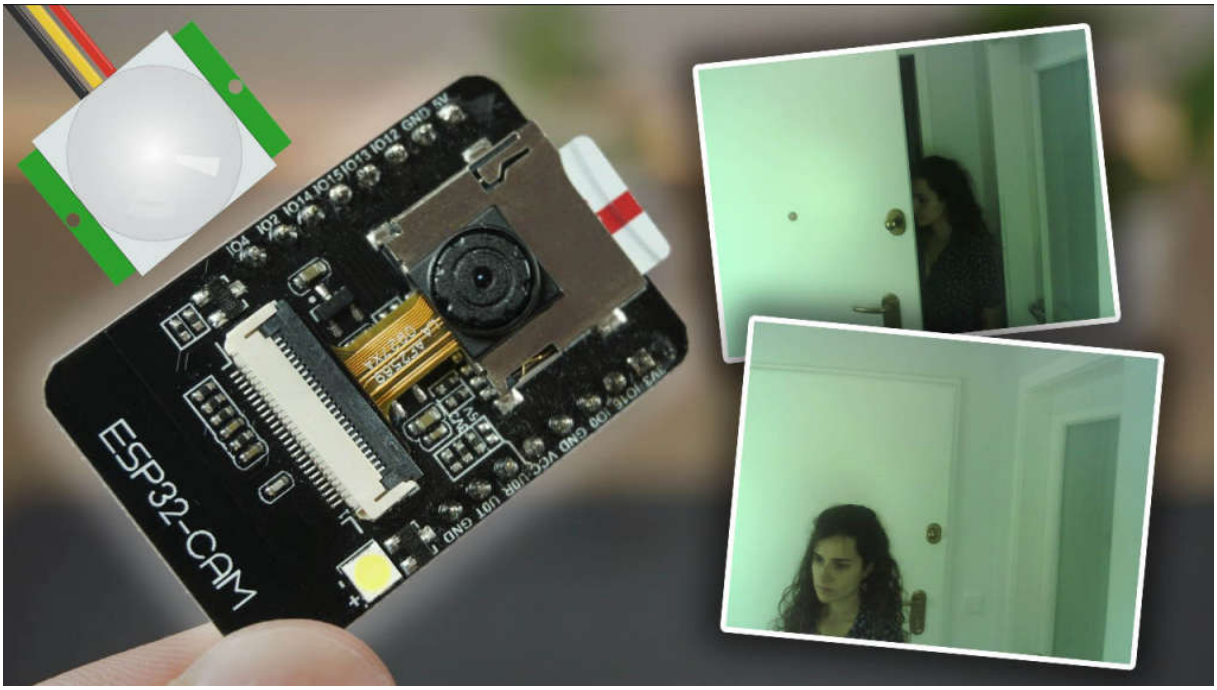
- Brownout detektor vagy Guru meditációs hiba
- A vázlat túl nagy hiba – Rossz partíciós rendszer van kiválasztva
- A COMX-nél a tábla nem érhető el – a COM-port nincs kiválasztva
- Psram hiba: A GPIO isr szolgáltatás nincs telepítve
- Gyenge Wi-Fi jel
- Nincs IP-cím az Arduino IDE soros monitorban
- Nem lehet megnyitni a webszervert
- A kép késik/sok késést mutat

Becsomagolás

Reméljük, hogy hasznosnak találtad ezt az oktatóanyagot, és fel tudod használni projektjeid során.

ESP32-CAM PIR mozgásérzékelő fotórögzítéssel (microSD kártyára ment)

Ebben a projektben egy mozgásérzékelős detektort fogunk készíteni fotórögzítéssel ESP32-CAM segítségével. Amikor a PIR-érzékelő mozgást észlel, felébred, fényképet készít, és elmenti a microSD-kártyára.



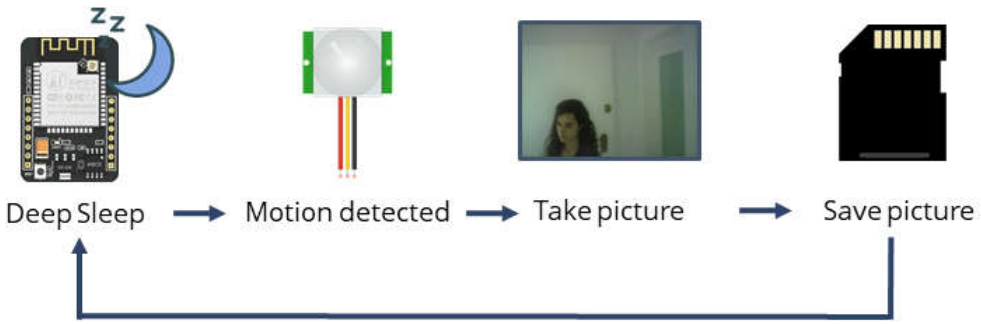
Ez a projekt nagyon hasonlít a korábbihoz, de annyit változtattunk, hogy PIR mozgásérzékelőt adtunk az áramkörhöz. Tehát, amikor a rendszer mozgást észlel, egy kép készül, és az a microSD-kártyára kerül.

Szükséges alkatrészek

Az oktatóanyag követéséhez a következő alkatrészekre van szükséged:

- ESP32-CAM OV2640-el
- Mikro SD kártya
- PIR mozgásérzékelő
- 2N3904 tranzisztor
- 1 kOhmos ellenállás
- 10 kOhmos ellenállás
- FTDI programozó
- Hüvelyes jumper vezetékek
- 5V-os tápegység az ESP32-CAM-hez

A projekt áttekintése



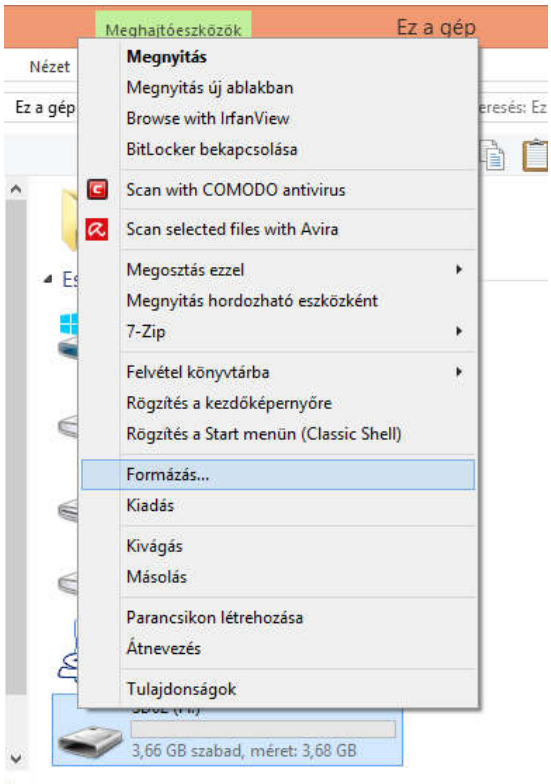
Íme egy gyors áttekintés a projekt működéséről.

- Az ESP32-CAM mély alvó módban van, a külső ébresztés engedélyezve van.
- Ha mozgást észlel, a PIR mozgásérzékelő jelet küld, hogy felébressze az ESP32-t.
- Az ESP32-CAM fényképet készít, és elmenti a microSD kártyára.
- Visszatér mély alvó üzemmódba, amíg új jelet nem kap a PIR mozgásérzékelőtől.

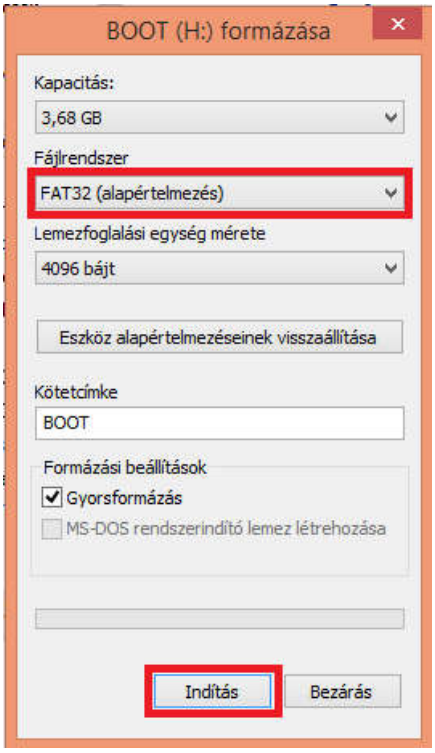
A MicroSD kártya formázása

Az első dolog, amit javasolunk, a microSD-kártya formázása. Használhatod a Windows formázó eszközt vagy bármilyen más microSD formázó szoftvert.

1. Helyezd be a microSD-kártyát a számítógépbe. Lépj az **Ez a gép** elemre, és kattints a jobb gombbal az SD-kártyára. Válaszd a **Formázás** lehetőséget az alábbi ábrán látható módon.



2. Egy új ablak jelenik meg. Válaszd ki a FAT32-t, nyomd meg az **Indítás** gombot a formázási folyamat inicializálásához, és kövesd a képernyőn megjelenő utasításokat.



Megjegyzés: a termék specifikációi szerint az ESP32-CAM csak 4 GB-os SD-kártyákat támogat. Azonban 16 GB-os SD-kártyával teszteltük, és jól működött.

Az ESP32 kiegészítő telepítése

Az ESP32 kártyát Arduino IDE segítségével programozzuk. Tehát telepítened kell az Arduino IDE-t, valamint az ESP32 kiegészítőt. Kövesd a következő oktatóanyagot az ESP32 bővítmény telepítéséhez, ha még nem tetted meg:

- Az ESP32 tábla telepítése Arduino IDE 2-ben (Windows, Mac OS X, Linux)

Fotó készítésének és mentésének vázlata

Másold a következő kódot az Arduino IDE-be.

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp32-cam-pir-
motion-detector-photo-capture/

  IMPORTANT!!!
   - Select Board "AI Thinker ESP32-CAM"
   - GPIO 0 must be connected to GND to upload a sketch
   - After connecting GPIO 0 to GND, press the ESP32-CAM on-board RESET button to
put your board in flashing mode

  Permission is hereby granted, free of charge, to any person obtaining a copy
```

```
  of this software and associated documentation files.
  The above copyright notice and this permission notice shall be included in all
  copies or substantial portions of the Software.
*********/

#include "esp_camera.h"
#include "Arduino.h"
#include "FS.h"                // SD Card ESP32
#include "SD_MMC.h"            // SD Card ESP32
#include "soc/soc.h"           // Disable brownour problems
#include "soc/rtc_cntl_reg.h"  // Disable brownour problems
#include "driver/rtc_io.h"
#include <EEPROM.h>            // read and write from flash memory
// define the number of bytes you want to access
#define EEPROM_SIZE 1

RTC_DATA_ATTR int bootCount = 0;

// Pin definition for 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

int pictureNumber = 0;

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector
  Serial.begin(115200);

  Serial.setDebugOutput(true);

  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_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_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;

  pinMode(4, INPUT);
  digitalWrite(4, LOW);
  rtc_gpio_hold_dis(GPIO_NUM_4);

  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA; // FRAMESIZE_ +
QVGA|CIF|VGA|SVGA|XGA|SXGA|UXGA
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

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

  Serial.println("Starting SD Card");

  delay(500);
  if(!SD_MMC.begin()){
    Serial.println("SD Card Mount Failed");
    //return;
  }

  uint8_t cardType = SD_MMC.cardType();
  if(cardType == CARD_NONE){
    Serial.println("No SD Card attached");
    return;
  }

  camera_fb_t * fb = NULL;

  // Take Picture with Camera
  fb = esp_camera_fb_get();
  if(!fb) {
    Serial.println("Camera capture failed");
    return;
  }
  // initialize EEPROM with predefined size
  EEPROM.begin(EEPROM_SIZE);
  pictureNumber = EEPROM.read(0) + 1;

  // Path where new picture will be saved in SD Card
  String path = "/picture" + String(pictureNumber) +".jpg";

  fs::FS &fs = SD_MMC;
```

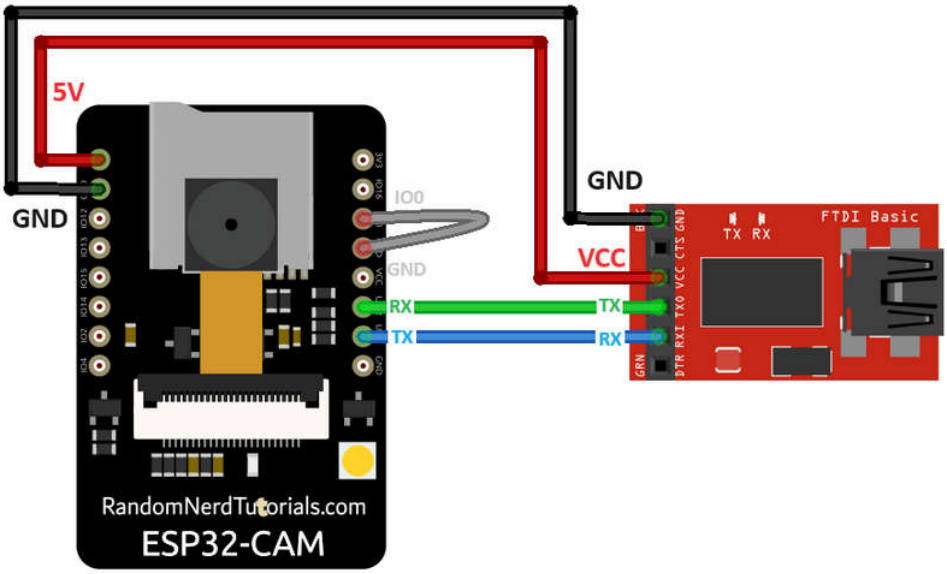
```
  Serial.printf("Picture file name: %s\n", path.c_str());

  File file = fs.open(path.c_str(), FILE_WRITE);
  if(!file){
    Serial.println("Failed to open file in writing mode");
  }
  else {
    file.write(fb->buf, fb->len); // payload (image), payload length
    Serial.printf("Saved file to path: %s\n", path.c_str());
    EEPROM.write(0, pictureNumber);
    EEPROM.commit();
  }
  file.close();
  esp_camera_fb_return(fb);

  delay(1000);

  // Turns off the ESP32-CAM white on-board LED (flash) connected to GPIO 4
  pinMode(4, OUTPUT);
  digitalWrite(4, LOW);
  rtc_gpio_hold_en(GPIO_NUM_4);

  esp_sleep_enable_ext0_wakeup(GPIO_NUM_13, 0);

  Serial.println("Going to sleep now");
  delay(1000);
  esp_deep_sleep_start();
  Serial.println("This will never be printed");
}

void loop() {

}
```

Ez a kód nagyon hasonlít az egyik korábbi ESP32-CAM projektünkhöz, de lehetővé teszi a külső felébresztést a GPIO 13-on.

```
esp_sleep_enable_ext0_wakeup(GPIO_NUM_13,0);
```

ESP32-CAM kód feltöltése

Ha kódot szeretnél feltölteni az ESP32-CAM kártyára, csatlakoztasd azt a számítógépéhez egy FTDI programozó segítségével. Kövesd a következő sematikus diagramot:



Sok FTDI programozó rendelkezik jumperrel, amely lehetővé teszi a 3,3 V vagy az 5 V kiválasztását. Győződj meg arról, hogy a jumper a megfelelő helyen van az 5V kiválasztásához.

Fontos: A GPIO 0-nak csatlakoznia kell a GND-hez, hogy fel tudd tölteni a kódot.

ESP32-CAM	FTDI Programmer
GND	GND
5V	VCC (5V)
U0R	TX
U0T	RX
GPIO 0	GND

A kód feltöltéséhez kövesd a következő lépéseket:

1) Lépjen az **Eszközök > Alaplap** elemre, és válassza az **AI-Thinker ESP32-CAM** lehetőséget.

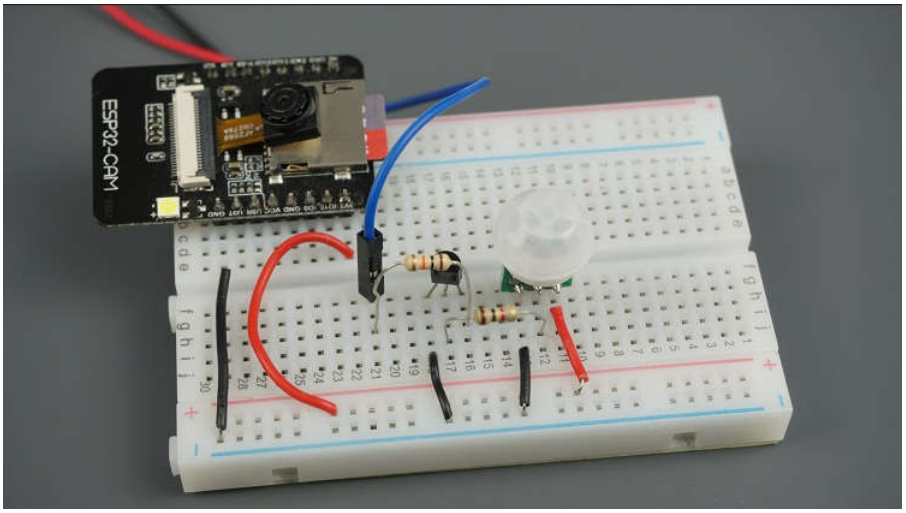
2) Válassza az **Eszközök > Port** lehetőséget, és válassza ki azt a COM-portot, amelyhez az ESP32 csatlakozik.

3) Ezután kattintson a feltöltés gombra a kód feltöltéséhez: 

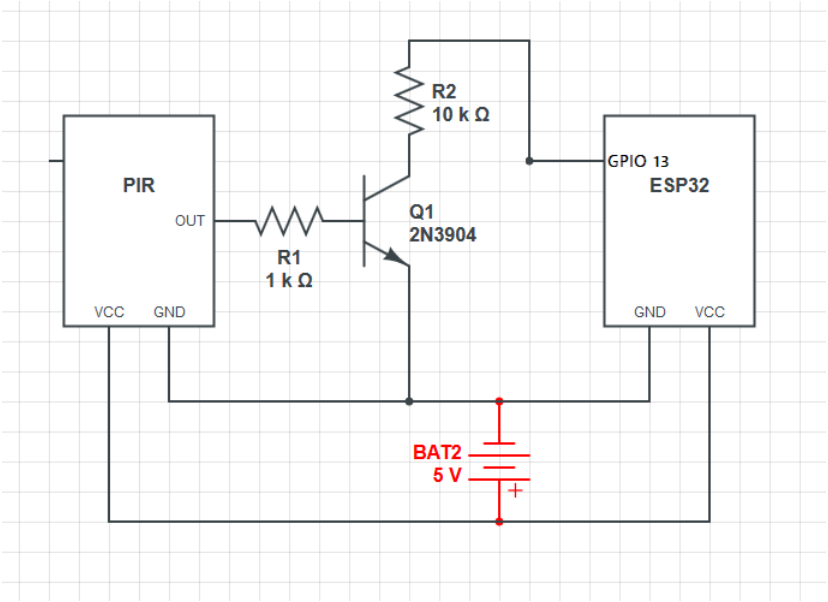
4) Ha a hibakereső ablakon néhány pont jelenik meg, előfordulhat, hogy meg kell nyomnod az ESP32-CAM fedélzeti RST gombját, ha az nem vált automatikusan flash módba.

Néhány másodperc múlva a kód sikeresen feltöltődik a táblára.

Sematikus diagram

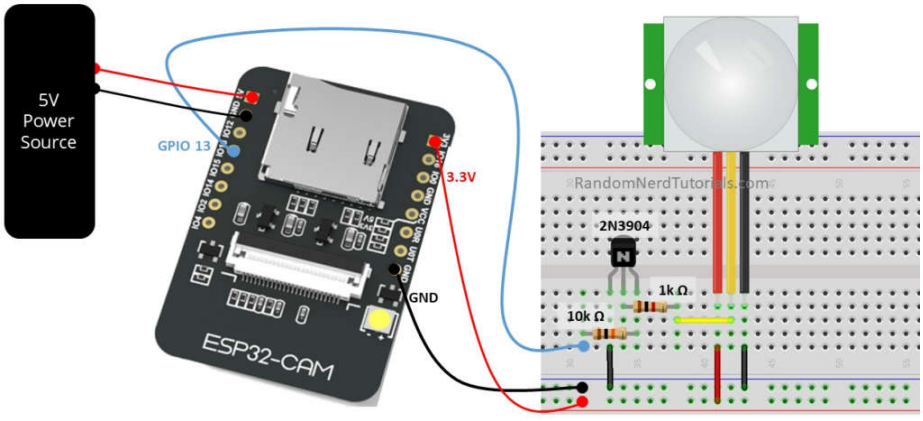


Szereld össze az összes alkatrészt az alábbi vázlatos diagram szerint.



Köszönjük David Graffnak, hogy megosztotta a projekt sematikus diagramját

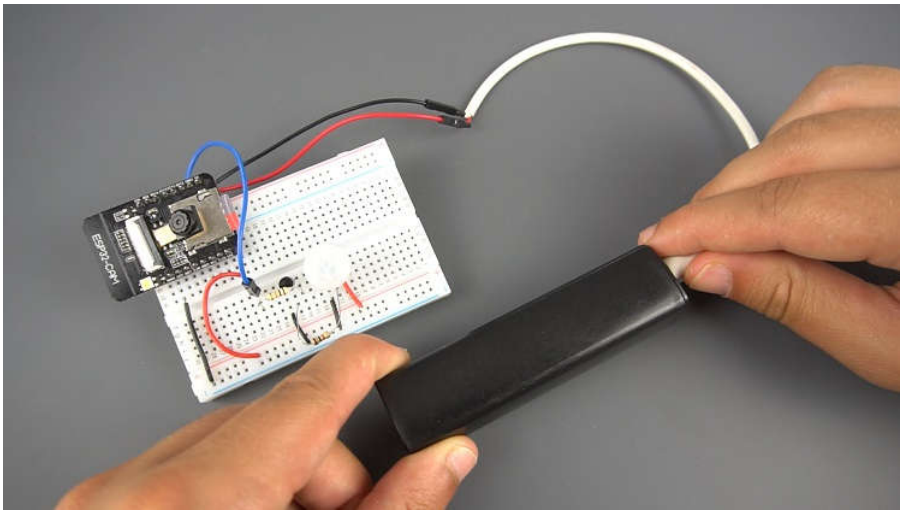
Ha úgy tetszik, kövesd a Fritzing diagramot.



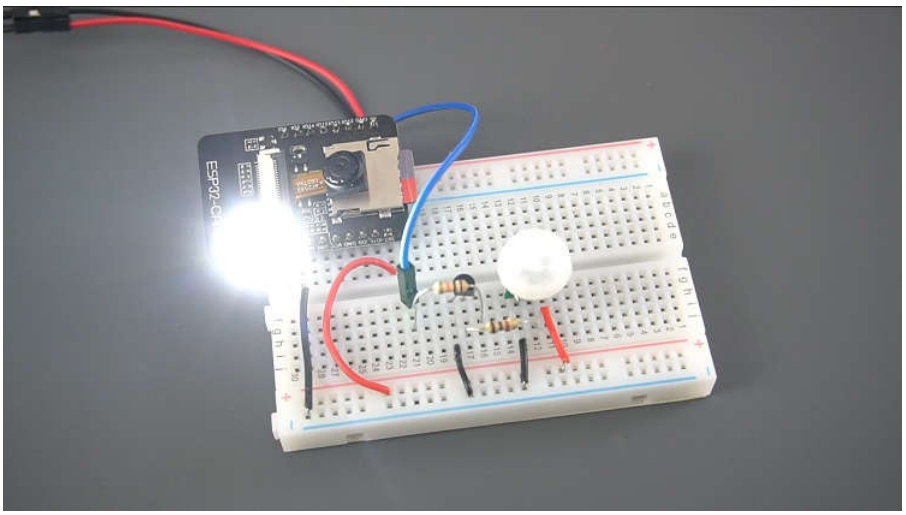
A feltöltés során felmerülő problémák elkerülése érdekében javasoljuk, hogy az áramkört csak a kód feltöltése után szereld össze.

Demonstráció

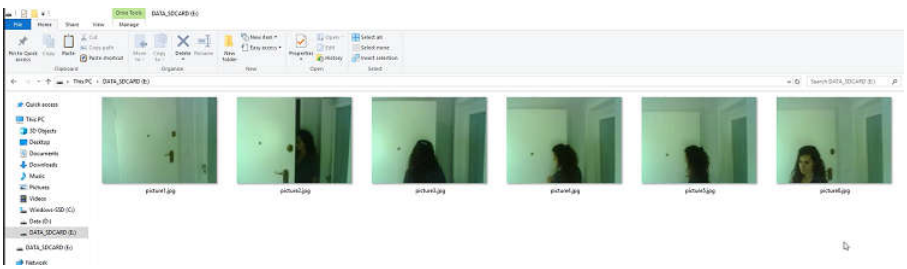
A kód feltöltése és az áramkör összeállítása után helyezz be egy formázott microSD-kártyát, és kapcsold be az áramkört – használhatsz például hordozható töltőt.



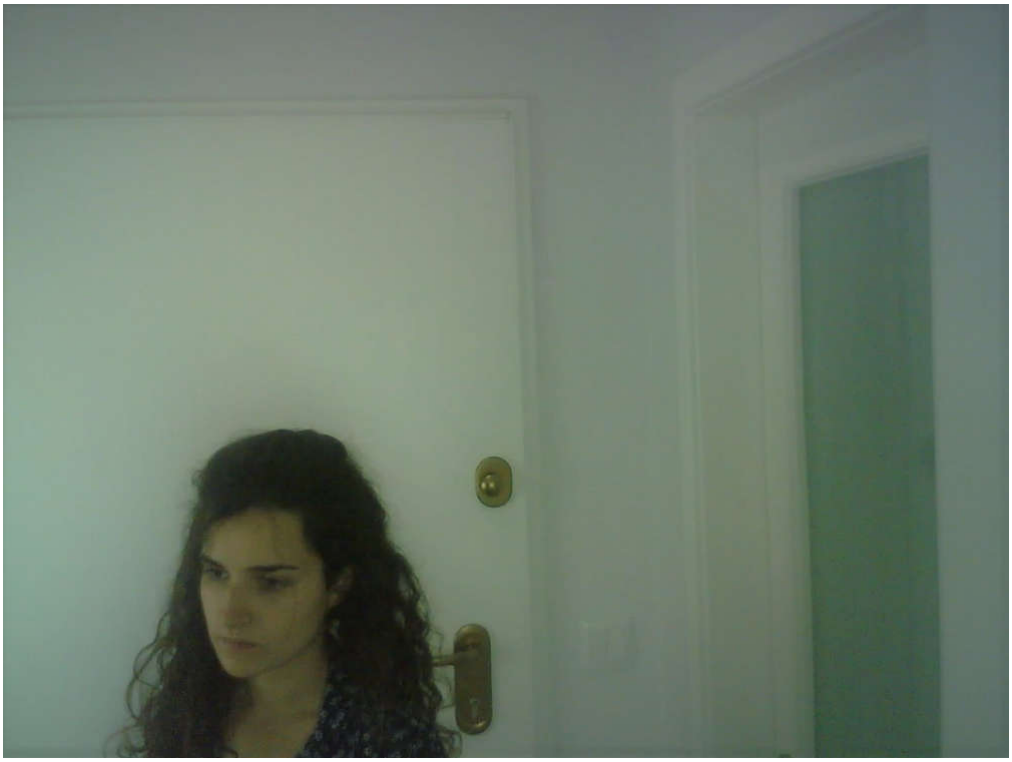
Ezután nyomd meg a reset (RST) gombot, és működnie kell. Ha mozgást észlel, bekapcsolja a vakut, fényképet készít, és elmenti a microSD kártyára.



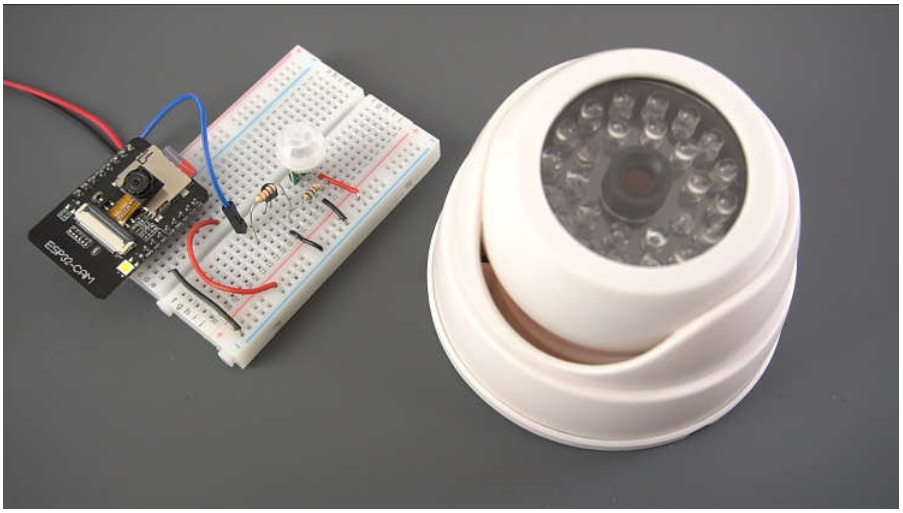
Kísérletezz többször ezzel az áramkörrel, hogy megbizonyosodj arról, hogy működik. Ezután helyezd be a microSD-kártyát a számítógépbe, hogy megtekinthesd a rögzített fényképeket.



Íme egy példa:



Most úgy fejezheted be ezt a projektet, ahogy akarod, vagy használj egy álkamerát, és helyezd be az ESP32-CAM-et a PIR mozgásérzékelővel, vagy megépítheted a saját házát.



Hibaelhárítás

Ha a következő hibák bármelyikét észleli, olvassa el az ESP32-CAM hibaelhárítási útmutatónkat: A leggyakoribb problémák javítása

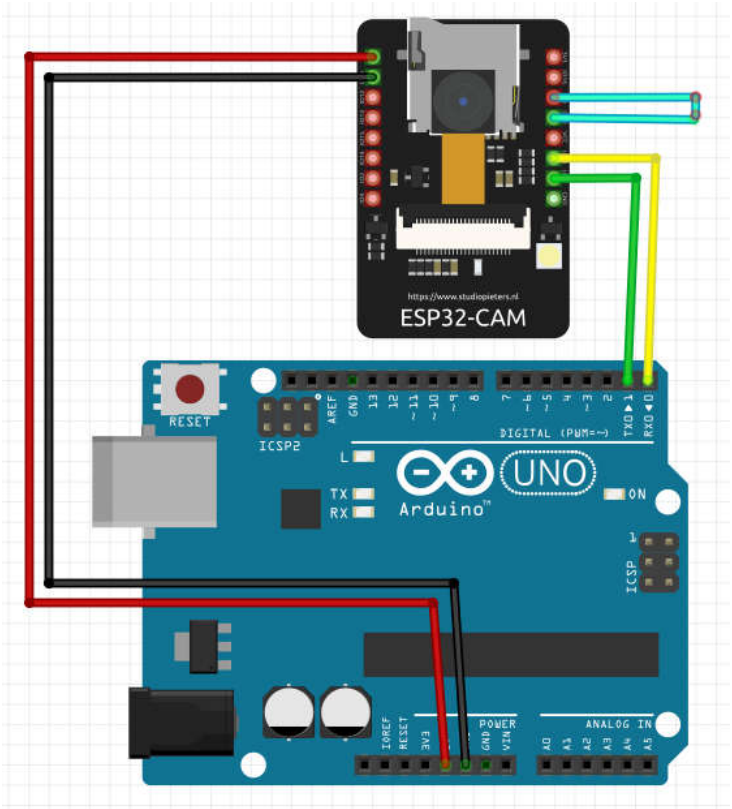
- Nem sikerült csatlakozni az ESP32-hez: Időtúllépés a csomagfejlécre várakozva
- A kamera indítása meghiúsult 0x20001 vagy hasonló hiba miatt
- Brownout detektor vagy Guru meditációs hiba
- A vázlat túl nagy hiba – Rossz partíciós rendszer van kiválasztva

- A COMX-nél a tábla nem érhető el – a COM-port nincs kiválasztva
- Psram hiba: A GPIO isr szolgáltatás nincs telepítve
- Gyenge Wi-Fi jel
- Nincs IP-cím az Arduino IDE soros monitorban
- Nem lehet megnyitni a webszervert
- A kép késik/sok késést mutat

Alternatív megoldások a kód feltöltéséhez

Nekem nem állt rendelkezésemre FTDI programozó, ezért más módszert kerestem a kód feltöltésének megoldására. Már korábban alkalmaztam ezeket a kapcsolásokat az ESP8266 alapú ESP-01-es Wi-Fi modul kódjának feltöltéséhez.

Megoldás Arduino Uno-val

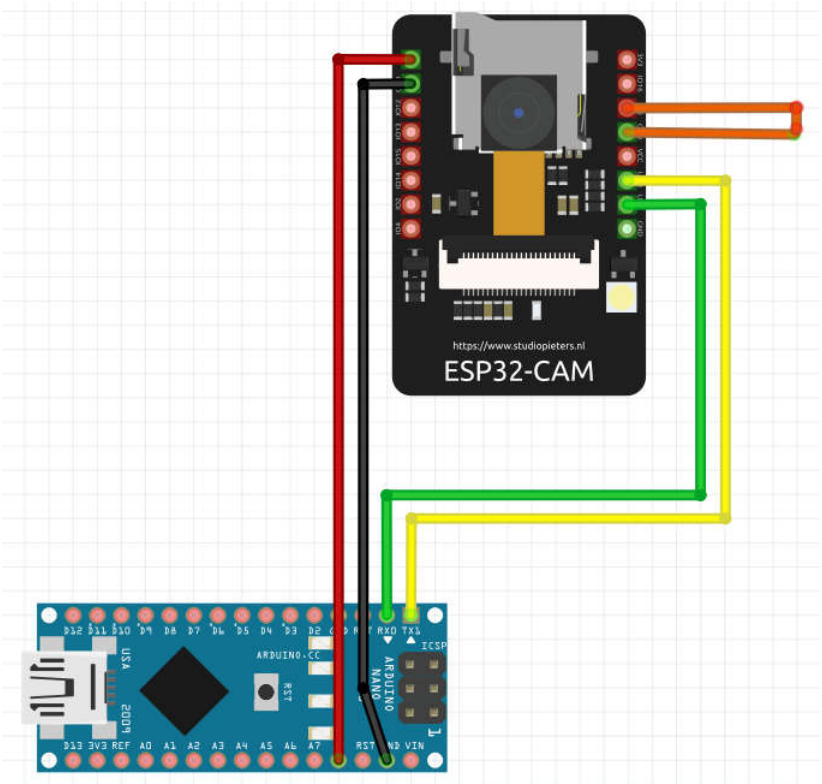


ESP32-CAM	Arduino Uno
GND	GND
5V	5V
U0R	TX
U0T	RX
GPIO 0	GND

Fontos: A GPIO 0-nak csatlakoznia kell a GND-hez, hogy fel tudd tölteni a kódot.

Maga a feltöltési folyamat lépései ugyanazok, mint az FTDI-vel való feltöltésnél.

Megoldás Arduino Nano-val



ESP32-CAM	Arduino Nano
GND	GND
5V	5V
U0R	TX
U0T	RX
GPIO 0	GND

Fontos: A GPIO 0-nak csatlakoznia kell a GND-hez, hogy fel tudd tölteni a kódot.

Maga a feltöltési folyamat lépései ugyanazok, mint az FTDI-vel való feltöltésnél.