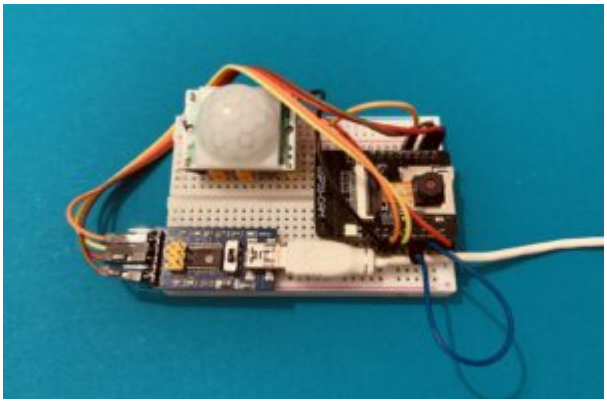
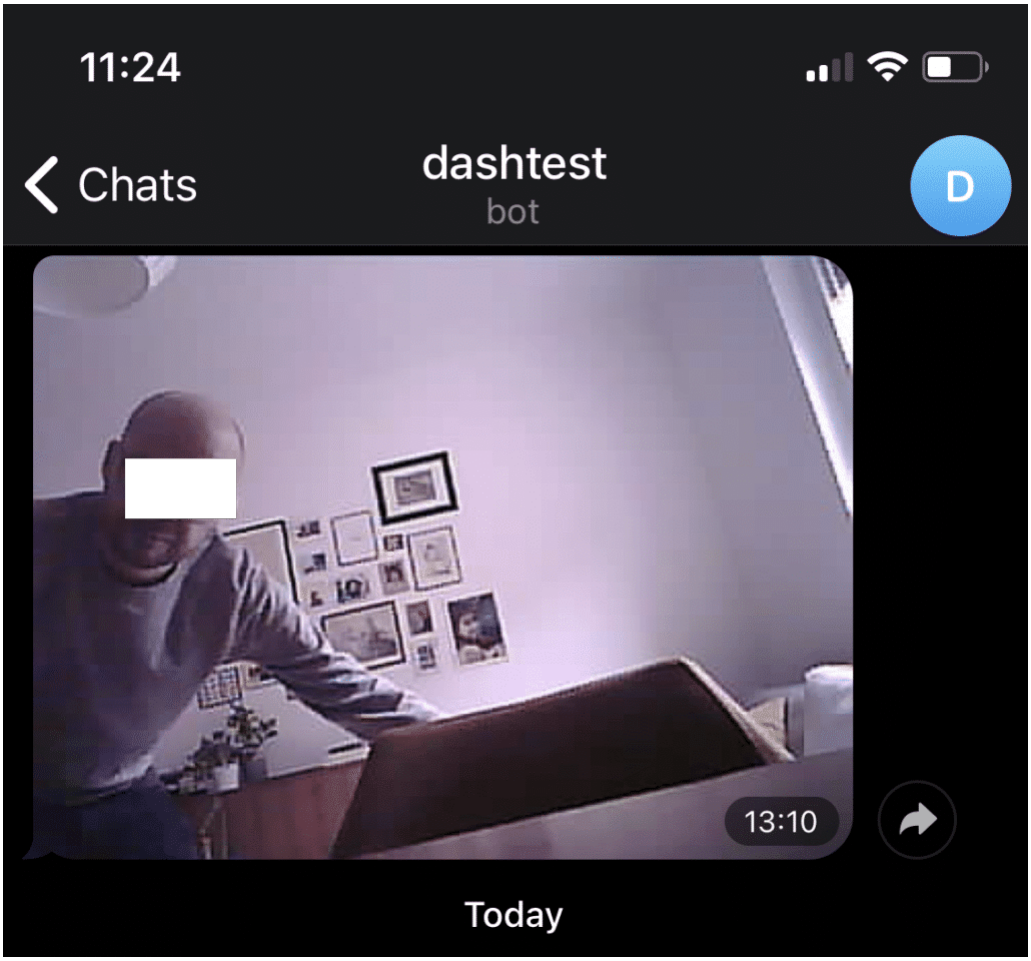
Eine Fotofalle mit der ESP32-CAM und Telegram



ESP32- und ESP8266-Boards sind zusammen mit Telegram eine tolle Kombination. Du kannst Daten blitzschnell auf dein Smartphone senden und von dort aus deinen Microcontroller steuern.

In diesem Projekt baust du eine Fotofalle mit einem speziellen ESP32-Board – der ESP32-CAM. Diese Falle schnappt zu, sobald sich jemand vor der Kamera bewegt. Dann nimmt sie ein Foto auf und sendet es umgehend an dein Smartphone.



Anfänger

1 Stunde

ca. 25 €

Für dieses Projekt benötigst du (Mengen s. Beschreibung):

[AZDelivery ESP32 Cam Modul + Bluetooth ESP32 5V Dual Core 32-bit CPU mit 2MP Kamera Entwicklungsboard kompatibel mit Arduino inklusive E-Book!]

14,99 €

Angebot

AZDelivery HC-SR501 PIR Bewegungssensor Bewegungsmelde Modul kompatibel mit Arduino und Raspberry Pi inklusive E-Book!

 PIR Modul zur Bewegungserkennung; Betriebsspannung 5V. Ideal für...

6,99 €

[AZDelivery Kompatibel mit FT232RL USB zu TTL Serial Adapter für 3,3V und 5V kompatibel mit Arduino inklusive E-Book!]
 AZDelivery Adapter kompatibel mit FT232RL USB zu TTL Serial für 3,3V und 5V.
6,99 €

Bevor du loslegen kannst, musst du noch ein paar Minuten in Vorbereitungen stecken.

Telegram vorbereiten

Zunächst benötigst du einen Account bei Telegram – und die dazugehörige App für dein Smartphone oder den Computer. Im Folgenden verwenden wir ein Smartphone. **Telegram ist kostenlos, werbefrei und funktioniert so ähnlich wie WhatsApp.** Allerdings hast du hier die Möglichkeit, Bots zu erstellen, mit denen du interagieren kannst.

In diesem Tutorial auf Pollux Labs lernst du, wie du deinen

eigenen Telegram-Bot erstellst.

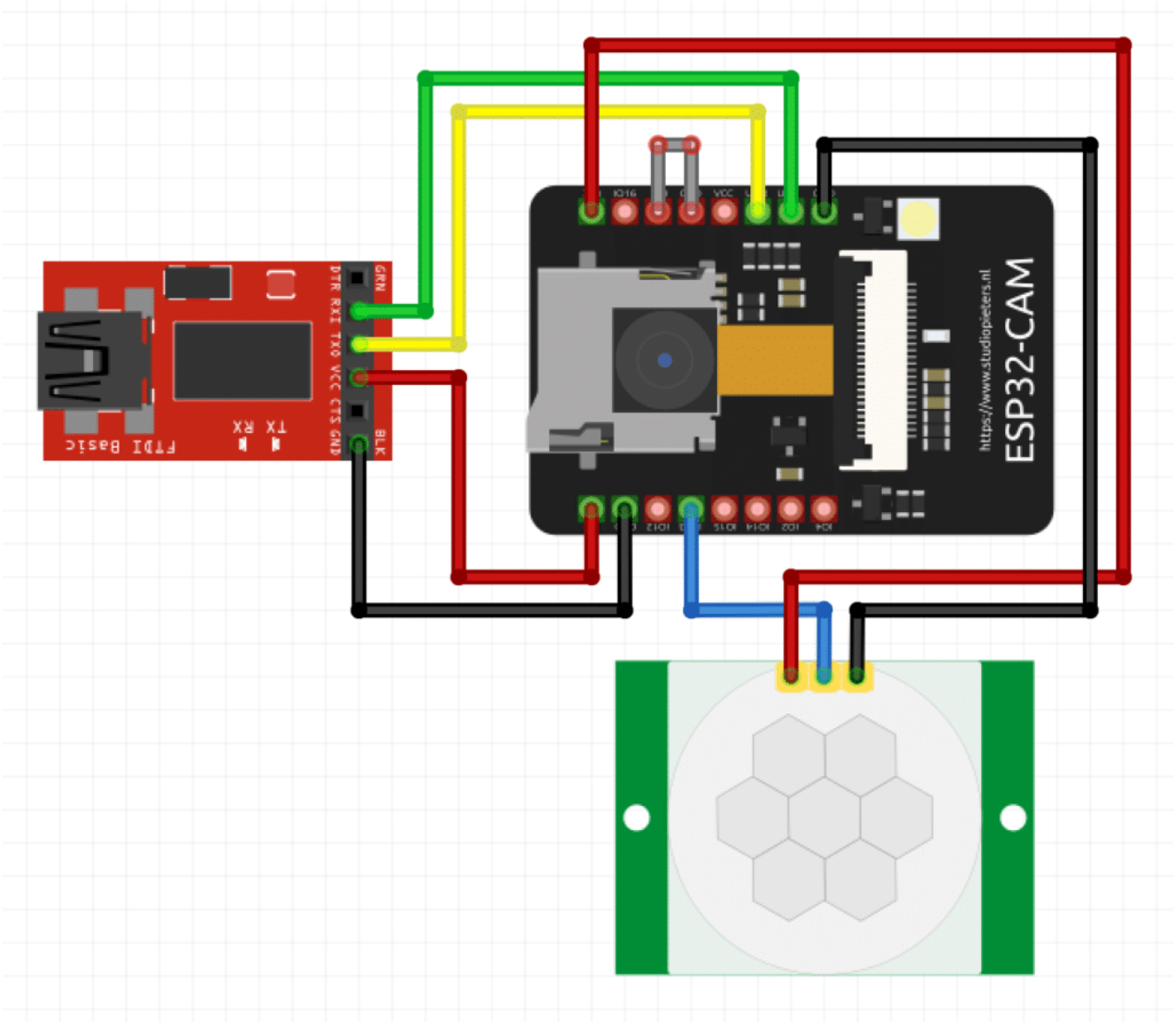
Die Fotofalle aufbauen

Die ESP32-CAM hat leider keinen USB-Port, weswegen du auf einen FTDI-Adapter zurückgreifen musst, um Sketches hochladen zu können. Wenn du hier noch keine Erfahrung sammeln konntest, informiere dich in diesem Tutorial darüber, wie du die ESP32-CAM programmieren kannst.

___STEADY_PAYWALL___

Wenn du den USB/Serial-Adapter verbunden hast, fehlt nur noch der Bewegungssensor. In diesem Projekt verwenden wir den PIR-Sensor HC-SR501, für den du nur einen Daten-Pin benötigst.

Orientiere dich beim Aufbau an diesem Schema:



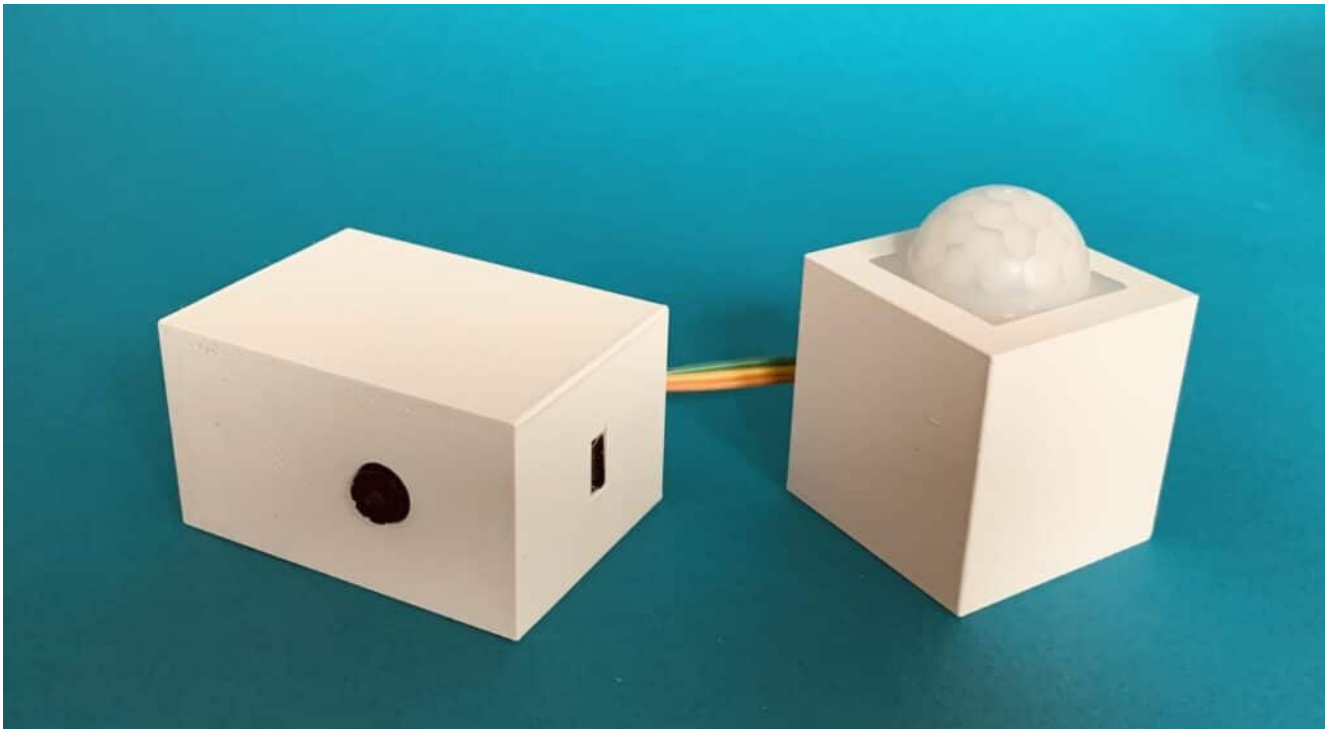
Hier noch einmal die Verbindungen übersichtlich dargestellt:

FTDI-Adapter	**ESP32-CAM**
GND	GND
VCC	5V
TXD	U0R
RXD	U0T
HC-SR501	**ESP32-CAM**
VCC	3v3
GND	GND
OUT	IO13

Hinweis: Im Schema oben findest du eine Brücke in Grau – denke immer daran, dass du diese benötigst, um einen Sketch hochzuladen. Sobald der Code auf deiner ESP32-CAM ist, **trenne die Brücke und drücke den RESET-Button** auf dem Board. Erst dann startet dein Sketch.

Das passende Gehäuse

Möchtest du die Fotofalle diskret aufbauen? Oder passt ein Breadboard nicht in das Design des Raums drumherum? **Dann sind unsere Gehäuse die richtige Wahl:**



Du kannst die .STL Dateien für die Gehäuse hier bei uns herunterladen.

Der Sketch

Für dieses Projekt haben wir einen Sketch von Random Nerd Tutorials als Ausgangspunkt genommen und diesen erweitert. Rui Santos verwendet hier die ESP32-CAM, um bei einer Bewegung ein

Foto auf eine SD-Karte zu speichern.

Wir haben den Aufbau und den Code so modifiziert, dass dein Board das Foto an deinen Telegram-Bot, also an dein Smartphone sendet, sobald eine Bewegung erkannt wurde.

Du findest den gesamten Sketch auf Github als .txt und am Ende dieses Artikels.

Wichtige Anpassungen im Code der Fotofalle

Damit der Sketch bei dir funktioniert, musst du ein paar Kleinigkeiten anpassen. Trage zunächst die Zugangsdaten deines WLAN-Netzwerks ein, damit sich die ESP32-CAM damit verbinden kann.

Anschließend benötigst du deinen Token und die UserID, die du beim Erstellen deines Telegram-Bots erhalten hast.

```
const char* ssid = "NETWORK";
const char* password = "PASSWORD";
String BOTtoken = "TOKEN";
String CHAT_ID = "USERID";
```

Eine Besonderheit, die du nach deinen Wünschen anpassen kannst, ist die Zeit zwischen den Fotos. Im Loop findest du einen **delay()** von 20 Sekunden. Das ist die Zeitspanne, in der der Bewegungssensor nach einem gesendeten Foto keine weitere Bewegung registriert.

Je kürzer du diese Zeitspanne einstellst, desto mehr Fotos erhältst du – sofern sich weiterhin jemand vor der Kamera bewegt.

Ebenso kannst du die Empfindlichkeit deines HC-SR501 einstellen. An ihm findest du zwei Potis: Wenn du die Platine nach oben drehst, kannst du am linken von ihnen die

Empfindlichkeit einstellen. Experimentiere hier ein wenig herum, um die passende Einstellung für dich zu finden.

Die Bibliothek UniversalTelegramBot

Die Bibliothek **UniversalTelegramBot.h** übernimmt die Kommunikation mit deinem Telegram-Bot. Du findest sie im Bibliotheksverwalter der Arduino IDE – diese kann jedoch veraltet sein. Deshalb empfehlen wir dir, die Bibliothek hier bei uns herunterzuladen.

Anschließend musst du diese Bibliothek in deinen Sketch einbinden, indem du im Menü der Arduino IDE **Sketch -> Bibliothek einbinden -> .ZIP-Bibliothek hinzufügen** wählst und die gerade heruntergeladene ZIP-Datei auswählst.

Noch ein Hinweis: Dieses Projekt dient dem Ausprobieren. Wenn du die Fotofalle produktiv einsetzen möchtest, mache dir vorab Gedanken über die IT-Sicherheit und achte darauf, keine Persönlichkeitsrechte zu verletzen!

Hier nun der gesamt Sketch zum Rauskopieren:

```
/*
  Rui Santos
            Complete project details at https://RandomNerdTutorials.com/telegram-esp32-cam-photo-arduino/

  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.

  Adapted by Pollux Labs – https://polluxlabs.net
*/
```

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_camera.h"
#include <UniversalTelegramBot.h>

const char* ssid = "NETWORK";
const char* password = "PASSWORD";

// Initialize Telegram BOT
String BOTtoken = "TOKEN";  // dein Bot-Token vom Botfather)

// Trage hier deine User-ID ein
String CHAT_ID = "CHAT-ID";

bool sendPhoto = false;

WiFiClientSecure clientTCP;
UniversalTelegramBot bot(BOTtoken, clientTCP);

//Pin of the motion sensor
#define PIR_PIN 13

//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


void configInitCamera() {
  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;

  //init with high specs to pre-allocate larger buffers
  if (psramFound()) {
    config.frame_size = FRAMESIZE_UXGA;
     config.jpeg_quality = 10;  //0-63 lower number means
higher quality
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
     config.jpeg_quality = 12;  //0-63 lower number means
higher quality
    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);
    delay(1000);
    ESP.restart();
  }

  // Drop down frame size for higher initial frame rate
  sensor_t * s = esp_camera_sensor_get();
      s->set_framesize(s,   FRAMESIZE_CIF);       //
UXGA|SXGA|XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA
}

String sendPhotoTelegram() {
  const char* myDomain = "api.telegram.org";
  String getAll = "";
  String getBody = "";

  camera_fb_t * fb = NULL;
  fb = esp_camera_fb_get();
  if(!fb) {
    Serial.println("Camera capture failed");
    delay(1000);
    ESP.restart();
    return "Camera capture failed";
  }
  Serial.println("Connect to " + String(myDomain));


  if (clientTCP.connect(myDomain, 443)) {
    Serial.println("Connection successful");
      String head = "--RandomNerdTutorials\r\nContent-
Disposition: form-data; name=\"chat_id\"; \r\n\r\n" + CHAT_ID
+ "\r\n--RandomNerdTutorials\r\nContent-Disposition: form-
data; name=\"photo\"; filename=\"esp32-cam.jpg\"\r\nContent-
Type: image/jpeg\r\n\r\n";
    String tail = "\r\n--RandomNerdTutorials--\r\n";

    uint16_t imageLen = fb->len;
    uint16_t extraLen = head.length() + tail.length();
    uint16_t totalLen = imageLen + extraLen;
```

```
      clientTCP.println("POST /bot"+BOTtoken+"/sendPhoto
HTTP/1.1");
    clientTCP.println("Host: " + String(myDomain));
    clientTCP.println("Content-Length: " + String(totalLen));
     clientTCP.println("Content-Type: multipart/form-data;
boundary=RandomNerdTutorials");
    clientTCP.println();
    clientTCP.print(head);
    uint8_t *fbBuf = fb->buf;
    size_t fbLen = fb->len;
    for (size_t n=0;n<fbLen;n=n+1024) {
      if (n+1024<fbLen) {
        clientTCP.write(fbBuf, 1024);
        fbBuf += 1024;
      }
      else if (fbLen%1024>0) {
        size_t remainder = fbLen%1024;
        clientTCP.write(fbBuf, remainder);
      }
    }
    clientTCP.print(tail);
    esp_camera_fb_return(fb);
    int waitTime = 10000;   // timeout 10 seconds
    long startTimer = millis();
    boolean state = false;
    while ((startTimer + waitTime) > millis()){
      Serial.print(".");
      delay(100);
      while (clientTCP.available()) {
        char c = clientTCP.read();
        if (state==true) getBody += String(c);
        if (c == '\n') {
          if (getAll.length()==0) state=true;
          getAll = "";
        }
        else if (c != '\r')
          getAll += String(c);
        startTimer = millis();
      }
      if (getBody.length()>0) break;
    }
```

```
    clientTCP.stop();
    Serial.println(getBody);
  }
  else {
    getBody="Connected to api.telegram.org failed.";
    Serial.println("Connected to api.telegram.org failed.");
  }
  return getBody;
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
  // Init Serial Monitor
  Serial.begin(115200);

  //Set PIR
  pinMode(PIR_PIN, INPUT);

  // Config and init the camera
  configInitCamera();

  // Connect to Wi-Fi
  WiFi.mode(WIFI_STA);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
   clientTCP.setCACert(TELEGRAM_CERTIFICATE_ROOT); // Add root
certificate for api.telegram.org
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
}

void loop() {
  Serial.println(digitalRead(PIR_PIN));
  if (digitalRead(PIR_PIN)) {
    Serial.println("Preparing photo");
    sendPhotoTelegram();
    delay(20000);
```

```
  }
}
```

Lade den obigen Sketch nun auf deine ESP32-CAM hoch. Achte darauf, dass du die Brücke für den Upload schließt, sie danach wieder öffnest und das Board neustartest mit dem Reset-Button. Im Seriellen Monitor solltest du nun sehen, dass sich das Board mit deinem WLAN verbindet und anschließend auf eine Bewegung wartet – hier sollten in schneller Folge Nullen durch den Seriellen Monitor jagen, bis dein PIR-Sensor eine Bewegung erkannt hat. Daraufhin sendet dir die ESP32-CAM ein Foto der aktuellen Situation an deinen Telegram-Bot.

Mögliche Fehler

Solltest du eine Fehlermeldung im Zusammenhang mit TELEGRAM_CERTIFICATE_ROOT bekommen, lade die aktuelle Version der Bibltiothek UniversalTelegramBot herunter. Möglicherweise ist die Version auf unserem Server bereits veraltet und noch nicht rechtzeitig von uns aktualisiert worden. Auf GitHub findest du die neueste Version – klicke dort rechts oben auf **Code** und anschließend auf **Download ZIP**.