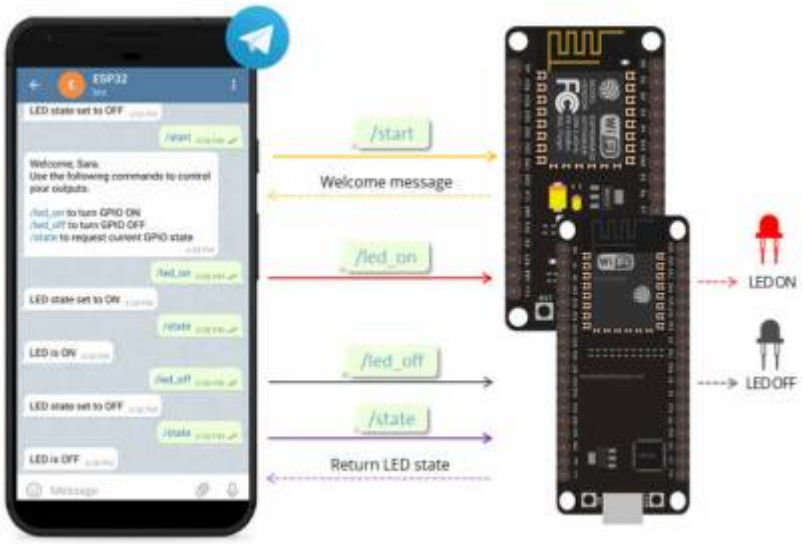
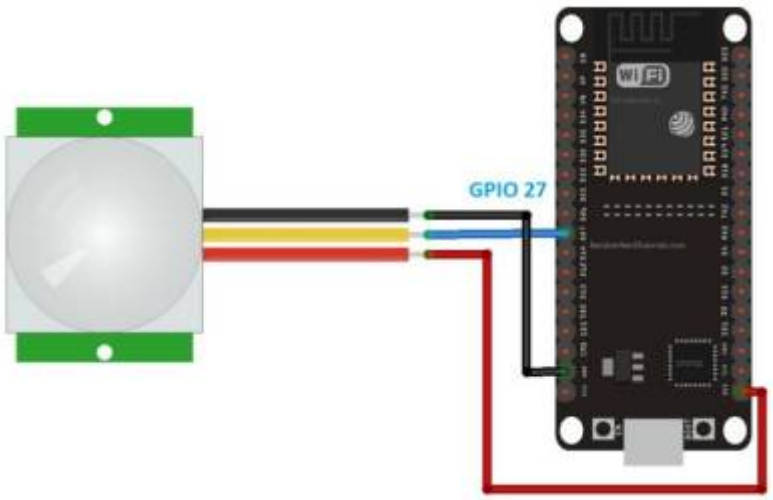
ESP 32 et Telegram

Esp32 et GPIO



ICI

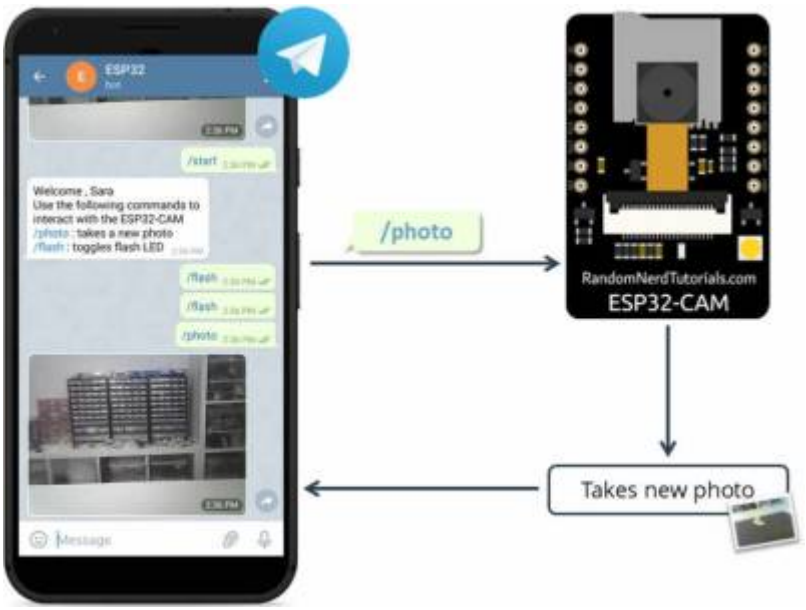
ESP32 Cam Telegram + Detection



ICI

CapteurMouvement ESP32 cam

ESP32 cam 001



Présentation du télégram

Telegram Messenger est un service de messagerie instantanée et de voix sur IP basé sur le cloud. Vous pouvez facilement l'installer sur votre smartphone (Android et iPhone) ou votre ordinateur (PC, Mac et Linux). C'est gratuit et sans publicité. Telegram vous permet de créer des robots avec lesquels vous pouvez interagir.

« Les bots sont des applications tierces qui s'exécutent dans Telegram. Les utilisateurs peuvent interagir avec les robots en leur envoyant des messages, des commandes et des requêtes en ligne. Vous contrôlez vos bots à l'aide de requêtes HTTPS vers l'API Telegram Bot ».

L'ESP32/ESP8266 interagira avec le bot Telegram pour recevoir et gérer les messages et envoyer des réponses. Dans ce didacticiel, vous apprendrez à utiliser Telegram pour envoyer des messages à votre bot afin de contrôler les sorties ESP de n'importe où (vous avez juste besoin de Telegram et d'un accès à Internet).

Télégram : ESP32-CAM Prendre et envoyer une photo (Arduino IDE)

ICI

Remarque : ce projet est compatible avec n'importe quelle carte de caméra ESP32 avec la caméra OV2640. Vous devez juste vous assurer que vous utilisez le bon brochage pour la carte que vous utilisez.

Voici un aperçu du projet que vous allez construire :

- Vous allez créer un bot Telegram pour votre ESP32-CAM ;
- Vous pouvez démarrer une conversation avec le bot ESP32-CAM ;

- Lorsque vous envoyez le message /la photo au bot ESP32-CAM, la carte ESP32-CAM reçoit le message, prend une nouvelle photo et répond avec cette photo ;
- Vous pouvez envoyer le message /flash pour basculer le flash LED de l'ESP32-CAM ;
- Vous pouvez envoyer le message /start pour recevoir un message de bienvenue avec les commandes pour contrôler la carte ;
- L'ESP32-CAM ne répondra qu'aux messages provenant de votre identifiant de compte Telegram.

Il s'agit d'un projet simple, mais montre comment vous pouvez utiliser Telegram dans vos projets IoT et domotique. L'idée est d'appliquer les concepts appris dans vos propres projets.

Création d'un bot de télégramme

Allez sur Google Play ou App Store, téléchargez et installez Telegram .

Ouvrez Telegram et suivez les étapes suivantes pour créer un Telegram Bot. Tout d'abord, recherchez « botfather » et cliquez sur BotFather comme indiqué ci-dessous. Ou ouvrez ce lien t.me/botfather dans votre smartphone.

La fenêtre suivante devrait s'ouvrir et vous serez invité à cliquer sur le bouton Démarrer .

Tapez /newbot et suivez les instructions pour créer votre bot. Donnez-lui un nom et un nom d'utilisateur.

Si votre bot est créé avec succès, vous recevrez un message avec un lien pour accéder au bot et au jeton de bot . Enregistrez le jeton du bot car vous en aurez besoin pour que l'ESP32 puisse interagir avec le bot

Remarque : le jeton du bot est une chaîne très longue. Pour vous assurer de bien faire les choses, vous pouvez accéder à l' interface Web de Telegram et copier votre jeton de bot à partir de là.

Obtenez votre identifiant d'utilisateur Telegram

Toute personne connaissant le nom d'utilisateur de votre bot peut interagir avec celui-ci. Pour vous assurer que nous ignorons les messages qui ne proviennent pas de notre compte Telegram (ou de tout utilisateur autorisé), vous pouvez obtenir votre identifiant d'utilisateur Telegram. Ensuite, lorsque votre bot télégramme reçoit un message, l'ESP peut vérifier si l'ID de l'expéditeur correspond à votre ID utilisateur et gérer le message ou l'ignorer.

Dans votre compte Telegram, recherchez « IDBot » ou ouvrez ce lien t.me/myidbot dans votre smartphone.

Démarrez une conversation avec ce bot et tapez /getid . Vous recevrez une réponse avec votre identifiant. Enregistrez cet ID utilisateur , car vous en aurez besoin plus tard dans ce didacticiel.

Préparation de l'IDE Arduino

Nous allons programmer la carte ESP32 à l' aide de l'IDE Arduino, alors assurez-vous qu'elle est installée dans votre IDE Arduino.

Installation de la carte ESP32 dans Arduino IDE (Windows, Mac OS X, Linux)

Bibliothèque universelle de robots Telegram

Pour interagir avec le bot Telegram, nous utiliserons la bibliothèque Universal Telegram Bot créée par Brian Lough qui fournit une interface simple pour l'API Telegram Bot.

Suivez les étapes suivantes pour installer la dernière version de la bibliothèque.

- Cliquez ici pour télécharger la bibliothèque Universal Arduino Telegram Bot .
- Allez dans Sketch > Inclure la bibliothèque > Ajouter une bibliothèque ZIP.. .
- Ajoutez la bibliothèque que vous venez de télécharger.

Important : n'installez pas la bibliothèque via Arduino Library Manager car cela pourrait installer une version obsolète.

Pour tous les détails sur la bibliothèque, consultez la page GitHub de la bibliothèque Universal Arduino Telegram Bot .

Bibliothèque ArduinoJson

Vous devez également installer la bibliothèque ArduinoJson . Suivez les étapes suivantes pour installer la bibliothèque.

- Accédez à Sketch > Inclure la bibliothèque > Gérer les bibliothèques .
- Recherchez “ArduinoJson”.
- Installez la bibliothèque.

Nous utilisons la bibliothèque ArduinoJson version 6.5.12.

Comment fonctionne le code

Cette section explique le fonctionnement du code. Continuez à lire pour savoir comment fonctionne le code ou passez à la section Démonstration.

Importation de bibliothèques

Commencez par importer les bibliothèques requises.

bibliotheques.ino

```
#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>
#include <ArduinoJson.h>
```

Identifiants du réseau

Insérez vos informations d'identification réseau dans les variables suivantes.

identifiantreseau.ino

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

ID utilisateur du télégramme

Insérez votre identifiant de chat. Celui que vous avez de l'IDBot.

identifiant_Ibot.ino

```
String CHAT_ID = "XXXXXXXXXX";
```

Jeton de robot de télégramme

Insérez votre jeton Telegram Bot que vous avez reçu de Botfather sur le BOTtoken variable.

jetonBot.ino

```
String BOTtoken = "XXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
```

Les envoyer une photoLa variable booléenne indique s'il est temps d'envoyer une nouvelle photo à votre compte télégramme. Par défaut, il est défini surfaux.

testenvoiePhoto.ino

```
bool sendPhoto = false;
```

Créez un nouveau client WiFi avec WiFiClientSécurisé.

```
WiFiClientSecure clientTCP;
```

Créez un bot avec le jeton et le client définis précédemment.

<codec> UniversalTelegramBot bot(BOTtoken, clientTCP); </code>

Créez une variable pour contenir la broche du flash LED (FLASH_LED_PIN). Dans l'ESP32-CAM AI Thinker, le flash est connecté àGPIO 4. Par défaut, définissez-le surMEUGLER.

```
define FLASH_LED_PIN 4
bool flashState = LOW;
```

Les botRequestDelay et lasTimeBotRanLes variables sont utilisées pour rechercher de nouveaux messages Telegram toutes les x secondes. Dans ce cas, le code vérifiera les nouveaux messages toutes les secondes (1000 millisecondes). Vous pouvez modifier ce délai dans lebotRequestDelay variable.

```
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;
```

Initialisation ESP32-CAM

Les lignes suivantes attribuent les broches ESP32-CAM AI-Thinker. Si vous utilisez un autre modèle de caméra ESP32, n'oubliez pas de modifier le brochage (lisez Cartes de caméra ESP32-CAM : Guide d'attribution des broches et des GPIO).

```
//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
```

Les configInitCamera() La fonction initialise la caméra ESP32.

```
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
}
```

handleNouveauxMessages()

Les handleNouveauxMessages() La fonction gère ce qui se passe lorsque de nouveaux messages arrivent.

<Code c> void handleNewMessages(int numNewMessages) {

```
Serial.print("Handle New Messages: ");
```

```
Serial.println(numNewMessages);
```

</code>

Il vérifie les messages disponibles :

```
for (int i = 0; i < numNewMessages; i++) {
```

Obtenez l'ID de chat pour un message particulier et stockez-le dans le chat_idvariable. L'ID de chat identifie qui a envoyé le message.

```
String chat_id = String(bot.messages[i].chat_id);
<:code>

Si la chat_id est différent de votre identifiant de chat (CHAT_ID), cela
signifie que quelqu'un (ce n'est pas vous) a envoyé un message à votre bot.
Si tel est le cas, ignorez le message et attendez le prochain message.

<code c>
if (chat_id != CHAT_ID){
  bot.sendMessage(chat_id, "Unauthorized user", "");
  continue;
}
```

Sinon, cela signifie que le message a été envoyé par un utilisateur valide, nous allons donc l'enregistrer dans le texte variable et vérifiez son contenu.

```
String text = bot.messages[i].text;
Serial.println(text);
```

Les de_nom variable enregistre le nom de l'expéditeur.

```
String from_name = bot.messages[i].from_name;
```

S'il reçoit le message /start , nous enverrons les commandes valides pour contrôler l'ESP. Ceci est utile s'il vous arrive d'oublier quelles sont les commandes pour contrôler votre carte.

```
if (text == "/start") {
  String welcome = "Welcome , " + from_name + "\n";
  welcome += "Use the following commands to interact with the ESP32-CAM \n";
  welcome += "/photo : takes a new photo\n";
  welcome += "/flash : toggles flash LED \n";
  bot.sendMessage(CHAT_ID, welcome, "");
}
```

Envoyer un message au bot est très simple. Il vous suffit d'utiliser leenvoyer le message() sur l'objet bot et passez en arguments l'ID de discussion du destinataire, le message et le mode d'analyse.

```
bool sendMessage(String chat_id, String text, String parse_mode = "");
```

Dans notre exemple, nous enverrons le message à l'ID stocké sur le CHAT_ID variable (qui correspond à votre identifiant de chat personnel) et envoyez le message enregistré sur la variable de bienvenue.

```
bot.sendMessage(CHAT_ID, welcome, "");
```

S'il reçoit le message /flash , inversez leflashStatevariable et mettre à jour l'état de la led flash. Si c'était auparavantMEUGLER, réglez-le sur HAUTE. Si c'était auparavantHAUTE, réglez-le sur MEUGLER.

```
if (text == "/flash") {
  flashState = !flashState;
  digitalWrite(FLASH_LED_PIN, flashState);
  Serial.println("Change flash LED state");
}
```

Enfin, s'il reçoit le message /photo , définissez leenvoyer une photovariable à vrai. Ensuite, dans leboucle(), vérifiez la valeur de envoyer une photo variable et procédez en conséquence.

```
if (text == "/photo") {
  sendPhoto = true;
  Serial.println("New photo request");
}
```

envoyerPhotoTelegram()

Les envoyerPhotoTelegram() La fonction prend une photo avec l'ESP32-CAM.

```
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";
}
```

Ensuite, il fait une requête HTTP POST pour envoyer la photo à votre bot de télégramme.

```
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);
```

mettre en place()

Dans le mettre en place(), initialisez le moniteur série.

```
Serial.begin(115200);
```

Réglez la LED flash comme sortie et réglez-la sur son état initial.

```
pinMode(FLASH_LED_PIN, OUTPUT);
digitalWrite(FLASH_LED_PIN, flashState);
```

Appeler le configInitCamera() fonction pour configurer et initialiser la caméra.

```
configInitCamera();
```

Connectez votre ESP32-CAM à votre réseau local.

```
WiFi.mode(WIFI_STA);
Serial.println();
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
  delay(500);
}
```

boucle()

Dans le boucle(), vérifiez l'état du envoyer une photovariable. Si c'estvrai, appeler le envoyerPhotoTelegram() fonction pour prendre et envoyer une photo à votre compte de télégramme. <codec> if (sendPhoto) {

```
Serial.println("Preparing photo");
sendPhotoTelegram();
```

</code> Quand c'est fait, réglez le envoyer une photo variable à faux.

```
sendPhoto = false;
```

Dans le boucle(), vous recherchez également de nouveaux messages chaque seconde.

```
if (millis() > lastTimeBotRan + botRequestDelay)  {
  int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
  while (numNewMessages) {
    Serial.println("got response");
    handleNewMessages(numNewMessages);
    numNewMessages = bot.getUpdates(bot.last_message_received + 1);
  }
  lastTimeBotRan = millis();
}
```

Lorsqu'un nouveau message arrive, appelez le handleNouveauxMessages() fonction.

```
while (numNewMessages) {
  Serial.println("got response");
  handleNewMessages(numNewMessages);
  numNewMessages = bot.getUpdates(bot.last_message_received + 1);
}
```

C'est à peu près comme ça que le code fonctionne.

Demonstration

Téléchargez le code sur votre carte ESP32-CAM. N'oubliez pas d'aller dans Outils > Tableau et sélectionnez le tableau que vous utilisez. Allez dans Outils > Port et sélectionnez le port COM auquel votre carte est connectée.

Après avoir téléchargé le code, appuyez sur le bouton RST intégré de l'ESP32-CAM pour qu'il commence à exécuter le code. Ensuite, vous pouvez ouvrir Serial Monitor pour vérifier ce qui se passe en arrière-plan.

Accédez à votre compte Telegram et ouvrez une conversation avec votre bot. Envoyez les commandes suivantes et voyez le bot répondre :

- /start affiche le message de bienvenue avec les commandes valides ;
- /flash inverse l'état du flash LED ;
- /photo prend une nouvelle photo et l'envoie à votre compte Telegram.

En même temps, sur le Serial Monitor, vous devriez voir que l'ESP32-CAM reçoit les messages.

Si vous essayez d'interagir avec votre bot à partir d'un autre compte, vous recevrez le message « Utilisateur non autorisé »

Code

Copiez le code suivant l'IDE Arduino. Pour que cette esquisse fonctionne pour vous, vous devez insérer vos informations d'identification réseau (SSID et mot de passe), le jeton Telegram Bot et votre identifiant utilisateur Telegram. De plus, vérifiez l'affectation des broches pour la carte de la caméra que vous utilisez.

ESP32Cam_Telegram.ino

```
/*
  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.
*/

#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>
#include <ArduinoJson.h>

const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

// Initialize Telegram BOT
String BOTtoken = "XXXXXXXXXX:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";  //
your Bot Token (Get from Botfather)

// Use @myidbot to find out the chat ID of an individual or a group
// Also note that you need to click "start" on a bot before it can
// message you
String CHAT_ID = "XXXXXXXXXX";

bool sendPhoto = false;

WiFiClientSecure clientTCP;
UniversalTelegramBot bot(BOTtoken, clientTCP);

#define FLASH_LED_PIN 4
bool flashState = LOW;

//Checks for new messages every 1 second.
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

//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
}

void handleNewMessages(int numNewMessages) {
  Serial.print("Handle New Messages: ");
  Serial.println(numNewMessages);

  for (int i = 0; i < numNewMessages; i++) {
    String chat_id = String(bot.messages[i].chat_id);
    if (chat_id != CHAT_ID){
      bot.sendMessage(chat_id, "Unauthorized user", "");
      continue;
    }

    // Print the received message
    String text = bot.messages[i].text;
    Serial.println(text);

    String from_name = bot.messages[i].from_name;
    if (text == "/start") {
      String welcome = "Welcome , " + from_name + "\n";
      welcome += "Use the following commands to interact with the
ESP32-CAM \n";
      welcome += "/photo : takes a new photo\n";
      welcome += "/flash : toggles flash LED \n";
      bot.sendMessage(CHAT_ID, welcome, "");
    }
    if (text == "/flash") {
      flashState = !flashState;
      digitalWrite(FLASH_LED_PIN, flashState);
      Serial.println("Change flash LED state");
    }
    if (text == "/photo") {
      sendPhoto = true;
      Serial.println("New photo request");
    }
  }
}

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 LED Flash as output
  pinMode(FLASH_LED_PIN, OUTPUT);
  digitalWrite(FLASH_LED_PIN, flashState);

  // 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);
  }
```

```
  Serial.println();
  Serial.print("ESP32-CAM IP Address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  if (sendPhoto) {
    Serial.println("Preparing photo");
    sendPhotoTelegram();
    sendPhoto = false;
  }
  if (millis() > lastTimeBotRan + botRequestDelay)  {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    while (numNewMessages) {
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }
}
```

ESP 32 Cam 002

ICI

Télégramme ESP32-CAM : envoyer l'image [Arduino IDE]

Comment créer un bot Telegram

La première étape consiste à créer un bot Telegram. Vous pouvez télécharger l'application depuis l'App Store ou Google Store ou utiliser une interface Web . Ouvrez l'application ou cliquez sur le lien et recherchez le Botfather :

Tapez ensuite la commande /newbot et donnez un nom à votre bot :

Intégration d'ESP32-CAM avec le bot Telegram

Une fois que nous avons créé notre bot Telegram, nous pouvons l'intégrer à ESP32-CAM. Pour ce faire, nous utiliserons une excellente bibliothèque nommée Universal Arduino Telegram bot . Il connecte l'ESP32-CAM à l'API du bot Telegram afin que nous puissions recevoir et envoyer des messages. De plus, nous pouvons envoyer des images capturées par l'ESP32-CAM. Installation du bot Universal Arduino Telegram dans l'IDE Arduino

Ouvrez l'IDE Arduino et dans le menu Sketch, sélectionnez Inclure l' élément de bibliothèque , puis Gérer la bibliothèque. Recherchez le télégramme universel Arduino et installez-le.

Assurez-vous que la version installée est la dernière. Sinon, vous pouvez cloner le dépôt github :

puis ajoutez la bibliothèque .zip.

Esquisse ESP32-CAM

Créez une nouvelle esquisse, puis ajoutez le code suivant :

esp32cam-telegramm002.ino

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include "esp_camera.h"
#include "camera_pins.h"
#include "UniversalTelegramBot.h"


#define BOT_TOKEN "your_bot_it"
#define _debug

const char* ssid = "wifi_ssid";
const char* password = "wifi_pwd";

WiFiClientSecure client;
UniversalTelegramBot bot(BOT_TOKEN, client);

long bot_last_check;
int bot_check_interval = 3000;

bool hasMoreData;
camera_fb_t * fb = NULL;

bool hasMoreDataAvailable();
byte* getNextBuffer();
int getBufferLen();

void setup() {
  Serial.begin(9600);
  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_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 PSRAM IC present, init with UXGA resolution and higher JPEG
quality
  // for larger pre-allocated frame buffer.
  if(psramFound()){
    config.frame_size = FRAMESIZE_QVGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_QVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // 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;
  }

  sensor_t * s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit
saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, 0); // lower the saturation
  }
  // drop down frame size for higher initial frame rate
```

```
  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  bot.longPoll = 60;
}


bool hasMoreDataAvailable() {
  Serial.println("Has more daa");
  if (hasMoreData) {
    hasMoreData = false;
    return true;
  }

  return false;
}


byte* getNextBuffer() {
  Serial.println("Next Buffer ");
  if (fb)
    return fb->buf;

  return nullptr;
}

int getBufferLen() {
Serial.println("Buffer len");
 if (fb)
   return fb->len;

  return 0;
}

void sendImage(String chat_id) {
  Serial.println("Sending Image");
```

```
  fb = NULL;
  fb = esp_camera_fb_get();
  hasMoreData = true;

  Serial.println(fb->len);

  bot.sendPhotoByBinary(chat_id, "image/jpeg", fb->len,
hasMoreDataAvailable, nullptr, getNextBuffer, getBufferLen);

  esp_camera_fb_return(fb);

}

void loop() {
  if (millis() > bot_last_check + bot_check_interval) {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    while (numNewMessages) {
        for (int i = 0; i < numNewMessages; i++) {
           String chat_id = bot.messages[i].chat_id;
           String msg = bot.messages[i].text;
           Serial.println("Chat id:" + chat_id);
           Serial.println("Msg: " + msg);
           sendImage(chat_id);
        }
        numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }

    bot_last_check = millis();
  }

 // delay(10);
}
```

Le noyau est la sendImagefonction. Tout d'abord, l'ESP32-CAM acquiert l'image et utilise le bot Universal Arduino Telegram pour envoyer l'image. Cette fonction utilise trois sous-fonctions différentes :

- hasMoreDataAvailable: pour savoir s'il y a plus de données à envoyer
- getNextBuffer: cela renvoie le tampon d'image acquis avant
- getBufferLen: cela renvoie la taille du tampon

Comme vous pouvez le remarquer, c'est très simple grâce à la merveilleuse bibliothèque Telegram Arduino.

Si vous souhaitez avoir plus d'informations sur ESP32-CAM, vous pouvez lire mon article expliquant comment capturer une image à l'aide de ESP32-CAM .

Testez le croquis : Envoi de l'image d'ESP32-CAM à Telegram

Nous pouvons maintenant tester le croquis Arduino. Nous utiliserons le bot Telegram pour déclencher l'acquisition d'images à partir de l'ESP32-CAM. Ensuite, l'ESP32-CAM renvoie l'image au client. Envoyer l'image de l'ESP32-CAM au télégramme

Comment améliorer l'ESP32-CAM et le télégramme

Le code ci-dessus envoie simplement une image à chaque fois que nous envoyons un message à l'ESP32-CAM. Nous pouvons l'améliorer et vérifier si le message contient une commande prédéfinie telle que /image ou /capture . Modifions le code comme indiqué ci-dessous :

modif_esp32cam_telegram002.ino

```
void loop() {
   if (millis() > bot_last_check + bot_check_interval) {
   int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
   while (numNewMessages) {
   for (int i = 0; i < numNewMessages; i++) {
   String chat_id = bot.messages[i].chat_id;
   String msg = bot.messages[i].text;
   Serial.println("Chat id:" + chat_id);
   Serial.println("Msg: " + msg);
   if (msg == "/capture") {
   sendImage(chat_id);
   }
   }
   numNewMessages = bot.getUpdates(bot.last_message_received + 1);
   }
   bot_last_check = millis();
   }
   // delay(10);
   }
```

Vérification de l'ID du télégramme

La prochaine amélioration consiste à vérifier l'ID du télégramme utilisé pour envoyer le message. De cette façon, nous ne pouvons accepter les messages que de notre client en évitant que d'autres puissent utiliser le bot.

Première recherche du Telegram ID Bot (nommé IDBot) :

puis tapez /getid. Vous obtiendrez votre pièce d'identité. Enfin, modifions le code :

Modif_identite_Esp32_cam_Telegram002.ino

```
void loop() {
    if (millis() > bot_last_check + bot_check_interval) {
```

```
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    while (numNewMessages) {
    for (int i = 0; i < numNewMessages; i++) {
    String chat_id = bot.messages[i].chat_id;
    String msg = bot.messages[i].text;
    Serial.println("Chat id:" + chat_id);
    Serial.println("Msg: " + msg);
    if (chat_id != "your_chat_id") {
    bot.sendMessage(chat_id, "You are not authorize to use this bot",
"");
    continue;
    }
    if (msg == "/capture") {
    sendImage(chat_id);  }
    }
    numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    bot_last_check = millis();
    }
    // delay(10);
    }
```