#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 = "Sathyavathi";

const char\* password = "sathya1234";

// Initialize Telegram BOT

String BOTtoken = "6745272067:AAHMrQhg8Or-f4-PiGLbrIjE7g8w05svgIQ";

// your Bot Token (Get from Botfather)

String CHAT\_ID = "989282823";

bool sendPhoto = false;

bool obstacleDetected = false;

// Flag to track obstacle detection WiFiClientSecure clientTCP;

UniversalTelegramBot bot(BOTtoken, clientTCP);

#define FLASH\_LED\_PIN 4

#define IR\_SENSOR\_PIN 14

// GPIO 14 for IR sensor input

#define OBSTACLE\_DELAY 5000

// Delay in milliseconds between sending photos when an obstacle is detected

bool flashState = LOW;

//Checks for new messages every 1 second. 23

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;

config.grab\_mode = CAMERA\_GRAB\_LATEST;

//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 = 1;

} else {

config.frame\_size = FRAMESIZE\_SVGA;

config.jpeg\_quality = 12;

//0-63 lower number means higher quality

config.fb\_count = 1;

}

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();

}

}

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 LockerSecuritySystem \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 = "";

//Dispose first picture because of bad quality

camera\_fb\_t \* fb = NULL;

fb = esp\_camera\_fb\_get();

esp\_camera\_fb\_return(fb);

// dispose the buffered image

// Take a new photo

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";

size\_t imageLen = fb->len;

size\_t extraLen = head.length() + tail.length();

size\_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);

pinMode(IR\_SENSOR\_PIN, INPUT);

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());

// Set up the IR sensor

attachInterrupt(digitalPinToInterrupt(IR\_SENSOR\_PIN), detectObstacle, CHANGE);

}

void loop() {

if (obstacleDetected) {

Serial.println("Obstacle detected");

captureAndSendPhotos();

obstacleDetected = false; // Reset the obstacle detection flag

}

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();

}

}

void detectObstacle() {

if (digitalRead(IR\_SENSOR\_PIN) == HIGH) {

obstacleDetected = true;

}

}

void captureAndSendPhotos() {

for (int i = 0; i < 5; i++) {

// Capture and send photos for 5 seconds

takeAndSendPhoto();

delay(1000);

}

}

void takeAndSendPhoto() {

// Take a new photo

camera\_fb\_t \* fb = NULL;

fb = esp\_camera\_fb\_get();

if(!fb) {

Serial.println("Camera capture failed");

delay(1000);

ESP.restart();

return;

}

// Send the photo via Telegram sendPPhotoTelegram(fb);

// Dispose the buffered image esp\_camera\_fb\_return(fb);

}

void sendPPhotoTelegram(camera\_fb\_t \*fb) { const char\* myDomain = "api.telegram.org"; String getAll = "";

String getBody = "";

// Connect to Telegram API

if (clientTCP.connect(myDomain, 443)) {

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"; 33

String tail = "\r\n--RandomNerdTutorials--\r\n";

size\_t imageLen = fb->len;

size\_t extraLen = head.length() + tail.length(); size\_t totalLen = imageLen + extraLen;

// Send HTTP POST request with photo data

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);

// Wait for response from Telegram API

while (clientTCP.available()) {

char c = clientTCP.read();

getBody += String(c);

}

// Close the TCP connection clientTCP.stop();

Serial.println("Response from Telegram API:"); Serial.println(getBody);

} else {

Serial.println("Failed to connect to Telegram API");

}

}