Making a new tomorrow in cyber stalking and e worms

Henrik Agerholm Ferrari, Þorvaldur Máni Danivalsson, Fei Gu

March 20, 2023

Contents

1 Problem statement

We want to make a product that can help with the outrageous prices that the security companies charge every month for small businesses or private individuals. We would be able to make something with Internet of Things,that can trigger sensors and give us some feedback to see, if there are any suspicious behaviors going on a certain parameter.

To fulfill this, we need to:

- Get the modules we need that can be used for the user to easily get access to.
- Connect the modules to an ESP32 device, so we can control it over the internet.
- Make a program that involves modules that can receive and send feedback.
- Set up a database for images to show to the user.

2 Illustration of network architecture

2.1 Modules

We decided to use the following modules:

- ESP32-Firebettle * 2
- ESP32-Cam
- Sound Sensor
- PIR Sensor
- Buzzer

2.2 net work architecture

Following the problem statement, we have to design the architecture base on three different terminal using three ESP32 MCU.

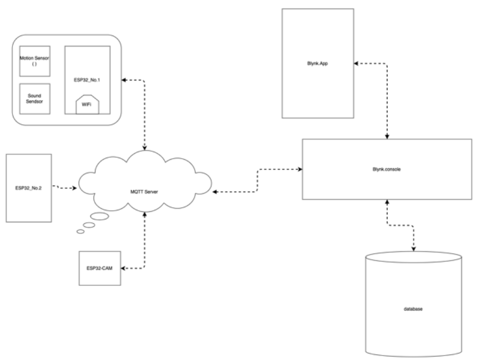


Figure 1: net work architecture

The first terminal which connect to the all sensor component will try to get the data. And then when the data value reach to a limit then send a message to MQTT server by Wi-Fi connection, at the same time send the data to “Blynk.console´´ to show the data.

The MQTT server will subscribe the topic
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

3 Illustration of the hardware setup

This part will explain the circuit and wire connection.

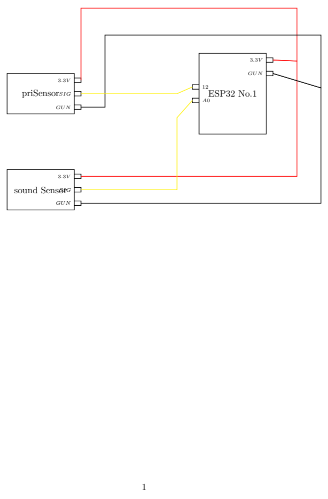
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

3.1 esp32No1 Sensor

This is the part ESP32 to connect to the sensors.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Figure 2: ESP32 no.1 connect to sensors

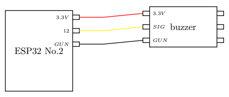


3.2 esp32No2 Buzzer

This part is talking about the ESP32 connect to the buzzer. this is a quite simple circuit we connect the 3.3 to power and gun to gun. and then connect the sig pin to D12 to set the data trans.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Figure 3: ESP32 no.2 connect to buzzer and LED-display



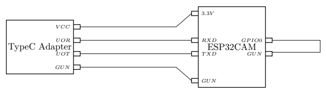
1

3.3 esp32Cam

This part is the connection about the Esp32 Camera can be flash under the develop mode. And when we leave the development mode then just unplug all the wires except the power and ground.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Figure 4: ESP32 CAM



1

4 Conclusion

We ended up with a rough prototype that can be used for a security system. We really wanted to show live feed from the ESP32-Camera, but all services that could help us with this costs money to use. That is why we used Blynk for some feedback from some modules because we thought that the cam service was free, but unfortunately it was not. It was just a nice and easy way for the user to get notified if something was in motion on their mobile phone.

A ESP32-Sensor Code

```
#include <Arduino.h>
#include "WiFi.h"
#include "../.pio/libdeps/esp32dev/Blynk/src/Blynk/BlynkTimer.h"
#include "../.pio/libdeps/esp32dev/Blynk/src/BlynkSimpleEsp32.h"
#include "../.pio/libdeps/esp32dev/Blynk/src/Blynk/BlynkHandlers.h"
#include "../.pio/libdeps/esp32dev/PubSubClient/src/PubSubClient.h"


#define BLYNK_TEMPLATE_ID "TMPLkv9OkR1o"
#define BLYNK_TEMPLATE_NAME "Quickstart Template"
#define BLYNK_AUTH_TOKEN "8MHFg1pmVwHBL8_RzEspb7Nl_pCF_Oml"

#define BLYNK_PRINT SERIAL
#define LEDPIN 2
#define PIRPIN 12
#define SOUNDPIN A0

int pirValue = LOW;
uint16_t soundValue = 0;

// Wi-Fi ssid and password
const char* ssid = "Evensnachi";
const char* pass = "12345678";
const char* mqttService = "mqtt.flespi.io";

WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);

BlynkTimer timer;
int timerID;

int cameraTrigger = 0;
int buzzerTrigger = 0;

void pubMqttBuzzerTriggerMsg(){
    String topicSensorValue = "buzzerTrigger";
    char publishTopic[topicSensorValue.length() + 1];
    strcpy(publishTopic, topicSensorValue.c_str());

    String messageSensorValue = String(buzzerTrigger);
    char publishMsg[messageSensorValue.length() + 1];
    strcpy(publishMsg, messageSensorValue.c_str());

    if(mqttClient.publish(publishTopic,publishMsg)){
        Serial.println("Topic: " + String(publishTopic));
        Serial.println("Message: " + String(publishMsg));
    }else{
        Serial.println("Publish Failed!");
    }
}

void pubMqttCameraTriggerMsg(){
    String topicSensorValue = "cameraTrigger";
    char publishTopic[topicSensorValue.length() + 1];
    strcpy(publishTopic, topicSensorValue.c_str());

    String messageSensorValue = String(cameraTrigger);
    char publishMsg[messageSensorValue.length() + 1];
    strcpy(publishMsg, messageSensorValue.c_str());

    if(mqttClient.publish(publishTopic,publishMsg)){
        Serial.println("Topic: " + String(publishTopic));
        Serial.println("Message: " + String(publishMsg));
    }else{
        Serial.println("Publish Failed!");
    }
}

void pubMqttSensorValueMsg() {

    String topicSensorValue = "sensorValue";
    char publishTopic[topicSensorValue.length() + 1];
    strcpy(publishTopic, topicSensorValue.c_str());

    String messageSensorValue = "sound: " + String(soundValue
   ) + "; pir: " + String(pirValue);
    char publishMsg[messageSensorValue.length() + 1];
    strcpy(publishMsg, messageSensorValue.c_str());

    if(mqttClient.publish(publishTopic,publishMsg)){
        Serial.println("Topic: " + String(publishTopic));
        Serial.println("Message: " + String(publishMsg));
    }else{
        Serial.println("Publish Failed!");
    }
}

void sendSensor(){
    pirValue = digitalRead(PIRPIN);
    soundValue = analogRead(SOUNDPIN);
    Serial.println(soundValue);
    Serial.println(pirValue);

    Blynk.virtualWrite(V3,soundValue);
    Blynk.virtualWrite(V4,pirValue);

    if(mqttClient.connected()){
        pubMqttSensorValueMsg();
        if(soundValue >= 2000 || pirValue == 1){
            buzzerTrigger = 1;
            cameraTrigger = 1;
            pubMqttCameraTriggerMsg();
            pubMqttBuzzerTriggerMsg();
        }
    }
}

void connectMQTTServer() {
//    String clientId = "esp32-Sensor-" + WiFi.macAddress();
    if(mqttClient.connect("ggg", "
   uszYF0QvKzAJ5kSCZByNuCbKukAMVf4fxu12kIoS7Mq1U8tHxPkRhksAsQcdV4gg
   ","")){
        Serial.println("MQTT Service connected!");
        Serial.println("Server address: ");
        Serial.println(mqttService);
    }else{
        Serial.println("MQTT server connect fail.. ");
        Serial.println("Client state: ");
        Serial.println(mqttClient.state());
        delay(3000);
    }
}

void wifiConnect(){
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, pass);
    WiFi.setSleep(false);
    Serial.println("Start to connect to wifi ..");

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

BLYNK_WRITE(V0){
    int value = param.asInt();
    Serial.println(value);

    if(value == 1){
        digitalWrite(LEDPIN,HIGH);
        Serial.println("led on! ");
        timerID = timer.setInterval(200L, sendSensor);
        timer.enable(timerID);
    } else {
        digitalWrite(LEDPIN, LOW);
        timer.disable(timerID);
    }
};


void setup() {
    Serial.begin(115200);

    pinMode(LEDPIN,OUTPUT);
    digitalWrite(LEDPIN,LOW);

    wifiConnect();

    mqttClient.setServer(mqttService,1883);
    connectMQTTServer();

    Blynk.begin(BLYNK_AUTH_TOKEN,ssid,pass);

}


void loop() {
    if(mqttClient.connected()){
        mqttClient.loop();
    }else{
        connectMQTTServer();
    }

    Blynk.run();
    timer.run();
}
```

B ESP32-Buzzer Code

```
#include <Arduino.h>
#include "WiFi.h"

#include "../.pio/libdeps/esp32dev/Blynk/src/Blynk/BlynkTimer.h"
#include "../.pio/libdeps/esp32dev/Blynk/src/BlynkSimpleEsp32.h"
#include "../.pio/libdeps/esp32dev/Blynk/src/Blynk/BlynkHandlers.h"
#include "../.pio/libdeps/esp32dev/PubSubClient/src/PubSubClient.h"

#define BLYNK_TEMPLATE_ID "TMPLkv9OkR1o"
#define BLYNK_TEMPLATE_NAME "Quickstart Template"
#define BLYNK_AUTH_TOKEN "8MHFg1pmVwHBL8_RzEspb7Nl_pCF_Oml"

#define BLYNK_PRINT SERIAL

#define BUZZERPIN 12

int trigger = 0;

// Wi-Fi ssid and password
const char* ssid = "Evensnachi";
const char* pass = "12345678";
const char* mqttService = "mqtt.flespi.io";

WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);

BlynkTimer timer;
int timerID;

void buzzerTone(){
    for (int i = 200; i <= 800; i++) {
        tone(BUZZERPIN,i);
        delay(5);
    }
//    delay(4000);
    for (int i = 800; i >=200; i--) {
        tone(BUZZERPIN,i);
        delay(10);
    }
```

```
}

void pubMqttBuzzerStatueMsg(){

    String topicSensorValue = "buzzerStatus"; // here insert
   the topic which you want to publish.
    char publishTopic[topicSensorValue.length() + 1];
    strcpy(publishTopic, topicSensorValue.c_str());

    String messageSensorValue = String(trigger); // here
   insert the message which you want to publish
    char publishMsg[messageSensorValue.length() + 1];
    strcpy(publishMsg, messageSensorValue.c_str());

    if(mqttClient.publish(publishTopic,publishMsg)){
        Serial.println("Topic: " + String(publishTopic));
        Serial.println("Message: " + String(publishMsg));
    }else{
        Serial.println("Publish Failed!");
    }
}

void receiveCallback(char* topic, byte* payload, unsigned int
    length) {
    Serial.print("Message Received [");
    Serial.print(topic);
    Serial.print("]");

    for (int i = 0; i < length; i++) {
        Serial.println((char) payload[i]);
    }

    Serial.println("");
    Serial.print("Message length(Bytes): ");
    Serial.println(length);

    if((char) payload[0] == 1){
        buzzerTone();
        trigger = 1;
        pubMqttBuzzerStatueMsg();
        Blynk.virtualWrite(V1,1);
    } else {
        tone(BUZZERPIN,0);
        trigger = 0;
        pubMqttBuzzerStatueMsg();
    }
}

void subscribeTopic() {
    String topicString = "buzzerTrigger"; // here insert the
   topic which you want to subscribed.
    char subTopic[topicString.length() + 1];
    strcpy(subTopic,topicString.c_str());

    if(mqttClient.subscribe(subTopic)){
        Serial.println("Subscrib Topic: ");
        Serial.println(subTopic);
    }else{
        Serial.println("Sbuscribe Fail..");
    }
}

void connectMQTTServer() {
//    String clientId = "esp32-Sensor-" + WiFi.macAddress();
    if(mqttClient.connect("ggg", "
   uszYF0QvKzAJ5kSCZByNuCbKukAMVf4fxu12kIoS7Mq1U8tHxPkRhksAsQcdV4gg
   ","")){
        Serial.println("MQTT Service connected!");
        Serial.println("Server address: ");
        Serial.println(mqttService);
        subscribeTopic(); // subscribe the topic which this
   method have.
    }else{
        Serial.println("MQTT server connect fail.. ");
        Serial.println("Client state: ");
        Serial.println(mqttClient.state());
        delay(3000);
    }
}

void wifiConnect(){
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, pass);
    WiFi.setSleep(false);
    Serial.println("Start to connect to wifi ..");

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

BLYNK_WRITE(V1){
    int value = param.asInt();
    Serial.println(value);

    if(value == 0){
        tune(BUZZERPIN,0);
        trigger = 0;
        pubMqttBuzzerStatueMsg();
    }
}

void setup() {
    Serial.begin(115200);
    Serial.setDebugOutput(true);

    pinMode(BUZZERPIN, OUTPUT);

    wifiConnect();

    if (WiFi.status() == WL_CONNECTED) {
        mqttClient.setServer(mqttService,1883);
        mqttClient.setCallback(receiveCallback);
        connectMQTTServer();
    } else {
        Serial.println("Waiting for WiFi .. ");
    }

    Blynk.begin(BLYNK_AUTH_TOKEN,ssid,pass);

}

void loop() {
    if(mqttClient.connected()){
        mqttClient.loop();
    }else{
        connectMQTTServer();
    }

    Blynk.run();
    timer.run();
```

```
}
```

C ESP32-Cam Code

```
#include "Arduino.h"
#include "esp_camera.h"
#include "WiFi.h"
#include "WiFiClient.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"  // Disable brownout problems
#include "driver/rtc_io.h"
#include <SPIFFS.h>
#include <FS.h>
#include "../.pio/libdeps/esp32cam/Firebase Arduino Client Library for ESP8266 and ESP32/src/Firebase_ESP_Client.h"
#include "../.pio/libdeps/esp32cam/Firebase Arduino Client Library for ESP8266 and ESP32/src/addons/TokenHelper.h" // Provide the token generation process info.
#include "../.pio/libdeps/esp32cam/Blynk/src/Blynk/BlynkTimer.h"
#include "../.pio/libdeps/esp32cam/Blynk/src/BlynkSimpleEsp32.h"
#include "../.pio/libdeps/esp32cam/Blynk/src/Blynk/BlynkHandlers.h"
#include "../.pio/libdeps/esp32cam/PubSubClient/src/PubSubClient.h"

#define CAMERA_MODEL_AI_THINKER // Has PSRAM
#include "camera_pins.h"


#define BLYNK_TEMPLATE_ID "TMPLkv9OkR1o"
#define BLYNK_TEMPLATE_NAME "Quickstart Template"
#define BLYNK_AUTH_TOKEN "U-QhTHFEP2aOvhuPmLMQrR2IrBRZoV24"
#define BLYNK_PRINT Serial


// OV2640 camera module pins (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

// Insert Firebase project API Key
#define API_KEY "AIzaSyAV4fMHmipIzuH4o3etOOTfp8xCpgXHxo4"

// Insert Authorized Email and Corresponding Password
#define USER_EMAIL "feix0033@easv365.dk"
#define USER_PASSWORD "12345678"

// Insert Firebase storage bucket ID e.g bucket-name.appspot.
   com
#define STORAGE_BUCKET_ID "esp32-smartpants.appspot.com"

// Photo File Name to save in SPIFFS
#define FILE_PHOTO "/data/photo.jpg"

// Wi-Fi ssid and password
const char* ssid = "Evensnachi";
const char* pass = "12345678";
const char* mqttService = "mqtt.flespi.io";

WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);

BlynkTimer timer;
int v3Value;
int takeNewPhoto = 0;
bool taskCompleted = false;

//Define Firebase Data objects
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig configF;

// Check if photo capture was successful
bool checkPhoto(fs::FS&fs){
    File f_pic = fs.open(FILE_PHOTO);
    unsigned int pic_sz = f_pic.size();
    return ( pic_sz > 100 );
```

```
}

// Capture Photo and Save it to SPIFFS
void capturePhotoSaveSpiffs( void ) {
    camera_fb_t * fb = NULL; // pointer
    bool ok = 0; // Boolean indicating if the picture has
   been taken correctly
    do {
        // Take a photo with the camera
        Serial.println("Taking a photo...");

        fb = esp_camera_fb_get();
        if (!fb) {
            Serial.println("Camera capture failed");
            return;
        }
        // Photo file name
        Serial.printf("Picture file name: %s\n", FILE_PHOTO);
        File file = SPIFFS.open(FILE_PHOTO, FILE_WRITE);
        // Insert the data in the photo file
        if (!file) {
            Serial.println("Failed to open file in writing
   mode");
        }
        else {
            file.write(fb->buf, fb->len); // payload (image),
    payload length
            Serial.print("The picture has been saved in ");
            Serial.print(FILE_PHOTO);
            Serial.print(" - Size: ");
            Serial.print(file.size());
            Serial.println(" bytes");
        }
        // Close the file
        file.close();
        esp_camera_fb_return(fb);

        // check if file has been correctly saved in SPIFFS
        ok = checkPhoto(SPIFFS);
    } while ( !ok );
}

void pubMqttCamStatus(){

    String topicSensorValue = "cameraStatus";
    char publishTopic[topicSensorValue.length() + 1];
    strcpy(publishTopic, topicSensorValue.c_str());

    String messageSensorValue = String(1);
    char publishMsg[messageSensorValue.length() + 1];
    strcpy(publishMsg, messageSensorValue.c_str());

    if(mqttClient.publish(publishTopic,publishMsg)){
        Serial.println("Topic: " + String(publishTopic));
        Serial.println("Message: " + String(publishMsg));
    }else{
        Serial.println("Publish Failed!");
    }
}

void pubMqttCamIpMsg() {

    String topicSensorValue = "cameraLink";
    char publishTopic[topicSensorValue.length() + 1];
    strcpy(publishTopic, topicSensorValue.c_str());

    String messageSensorValue = WiFi.localIP().toString();
    char publishMsg[messageSensorValue.length() + 1];
    strcpy(publishMsg, messageSensorValue.c_str());

    if(mqttClient.publish(publishTopic,publishMsg)){
        Serial.println("Topic: " + String(publishTopic));
        Serial.println("Message: " + String(publishMsg));
    }else{
        Serial.println("Publish Failed!");
    }
}

void startCameraServer();

void receiveCallback(char* topic, byte* payload, unsigned int
     length) {
    Serial.print("Message Received [");
    Serial.print(topic);
    Serial.print("]");

    for (int i = 0; i < length; i++) {
        Serial.println((char) payload[i]);
    }

    Serial.println("");
    Serial.print("Message length(Bytes): ");
    Serial.println(length);

    if ((char) payload[0] == 1) {
        Serial.println("Start camera server");
        startCameraServer();

        Serial.print("Camera Ready! Use 'http://");
        Serial.print(WiFi.localIP());
        Serial.println("' to connect");

        Blynk.virtualWrite(V2,1);
        Blynk.virtualWrite(V5,WiFi.localIP().toString());

        pubMqttCamIpMsg();
        pubMqttCamStatus();

        if (takeNewPhoto) {
            capturePhotoSaveSpiffs();
            takeNewPhoto = 0;
            Blynk.virtualWrite(V6,takeNewPhoto);
        }

        delay(1);

        if (Firebase.ready() && !taskCompleted){
            taskCompleted = true;
            Serial.print("Uploading picture... ");

            //MIME type should be valid to avoid the download problem.
            //The file systems for flash and SD/SDMMC can be changed in FirebaseFS.h.
            if (Firebase.Storage.upload(&fbdo, STORAGE_BUCKET_ID /* Firebase Storage bucket id */, FILE_PHOTO /* path to local file */, mem_storage_type_flash /* memory storage type, mem_storage_type_flash and mem_storage_type_sd */, FILE_PHOTO /* path of remote file stored in the bucket */, "image/jpeg" /* mime type */)){
                Serial.printf("\nDownload URL: %s\n", fbdo.downloadURL().c_str());
            } else {
                Serial.println(fbdo.errorReason());
```

```
              }
          }
      }


}

void subscribeTopic() {
    String topicString = "camerTrigger"; // topic name
    char subTopic[topicString.length() + 1];
    strcpy(subTopic,topicString.c_str());

    if(mqttClient.subscribe(subTopic)){  // subscribe the
   topic
        Serial.println("Subscrib Topic: ");
        Serial.println(subTopic);
    }else{
        Serial.println("Sbuscribe Fail..");
    }
}

void connectMQTTServer() {

    if(mqttClient.connect("ggg", "
   uszYF0QvKzAJ5kSCZByNuCbKukAMVf4fxu12kIoS7Mq1U8tHxPkRhksAsQcdV4gg
   ","")){
        Serial.println("MQTT Service connected!");
        Serial.println("Server address: ");
        Serial.println(mqttService);
        subscribeTopic(); // subscribe the topic which this
   method have.
    }else{
        Serial.println("MQTT server connect fail.. ");
        Serial.println("Client state: ");
        Serial.println(mqttClient.state());
        delay(3000);
    }
}

void wifiConnect(){
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, pass);
    WiFi.setSleep(false);
    Serial.println("Start to connect to wifi ..");

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

void initSPIFFS(){
    if (!SPIFFS.begin(true)) {
        Serial.println("An Error has occurred while mounting
   SPIFFS");
        ESP.restart();
    }
    else {
        delay(500);
        Serial.println("SPIFFS mounted successfully");
    }
}

void cameraInitProcess() {
    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.frame_size = FRAMESIZE_UXGA;
    config.pixel_format = PIXFORMAT_JPEG; // for streaming
    //config.pixel_format = PIXFORMAT_RGB565; // for face
    detection/recognition
//    config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
//    config.fb_location = CAMERA_FB_IN_PSRAM;
//    config.jpeg_quality = 12;
//    config.fb_count = 1;

//    // if PSRAM IC present, init with UXGA resolution and
    higher JPEG quality
//    //                      for larger pre-allocated frame
    buffer.
//    if(config.pixel_format == PIXFORMAT_JPEG){
//        if(psramFound()){
//            config.jpeg_quality = 10;
//            config.fb_count = 2;
//            config.grab_mode = CAMERA_GRAB_LATEST;
//        } else {
//            // Limit the frame size when PSRAM is not
    available
//            config.frame_size = FRAMESIZE_SVGA;
//            config.fb_location = CAMERA_FB_IN_DRAM;
//        }
//    } else {
//        // Best option for face detection/recognition
//        config.frame_size = FRAMESIZE_240X240;
//#if CONFIG_IDF_TARGET_ESP32S3
//        config.fb_count = 2;
//#endif
//    }
//
//#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, -2); // lower the saturation
//    }
//    // drop down frame size for higher initial frame rate
//    if(config.pixel_format == PIXFORMAT_JPEG){
//        s->set_framesize(s, FRAMESIZE_QVGA);
//    }
//
//#if defined(CAMERA_MODEL_M5STACK_WIDE) || defined(
    CAMERA_MODEL_M5STACK_ESP32CAM)
//    s->set_vflip(s, 1);
//  s->set_hmirror(s, 1);
//#endif
//
//#if defined(CAMERA_MODEL_ESP32S3_EYE)
//    s->set_vflip(s, 1);
//#endif
    if (psramFound()) {
        config.frame_size = FRAMESIZE_UXGA;
        config.jpeg_quality = 10;
        config.fb_count = 2;
    } else {
        config.frame_size = FRAMESIZE_SVGA;
        config.jpeg_quality = 12;
        config.fb_count = 1;
    }
    // Camera init
    esp_err_t err = esp_camera_init(&config);
    if (err != ESP_OK) {
        Serial.printf("Camera init failed with error 0x%x",
    err);
        ESP.restart();
    }

}

BLYNK_WRITE(V6){
    takeNewPhoto = param.asInt();
}

void setup() {
    Serial.begin(115200);
    Serial.setDebugOutput(true);

    initSPIFFS();
    WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
    cameraInitProcess();
    wifiConnect();

    if (WiFi.status() == WL_CONNECTED) {
        Serial.println("Start camera service .. ");

        mqttClient.setServer(mqttService, 1883); //set the mqtt service to connect
        mqttClient.setCallback(receiveCallback); //set the callback method to keep running the method witch can receive mqtt message.
        connectMQTTServer(); //connect to the mqtt server

        //Firebase
        // Assign the api key
        configF.api_key = API_KEY;
        //Assign the user sign in credentials
        auth.user.email = USER_EMAIL;
        auth.user.password = USER_PASSWORD;
        //Assign the callback function for the long running token generation task
        configF.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h

        Firebase.begin(&configF, &auth);
        Firebase.reconnectWiFi(true);

    }else {
        Serial.println("Waiting for WiFi .. ");
    }

    Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
}

void loop() {
    if(mqttClient.connected()){
        mqttClient.loop();
    }else{
        connectMQTTServer();
    }

    Blynk.run();
    timer.run();
//    delay(10000);
}
```