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 **2**

2 Illustration of network architecture **2**
2.1 Modules 2
2.2 Net work architecture 3

3 Illustration of the hardware setup **4**
3.1 esp32No1 Sensor 5
3.2 esp32No2 Buzzer 6
3.3 esp32Cam 7

4 Conclusion **8**

A ESP32-Sensor Code **9**

B ESP32-Buzzer Code **13**

C ESP32-Cam Code **17**

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

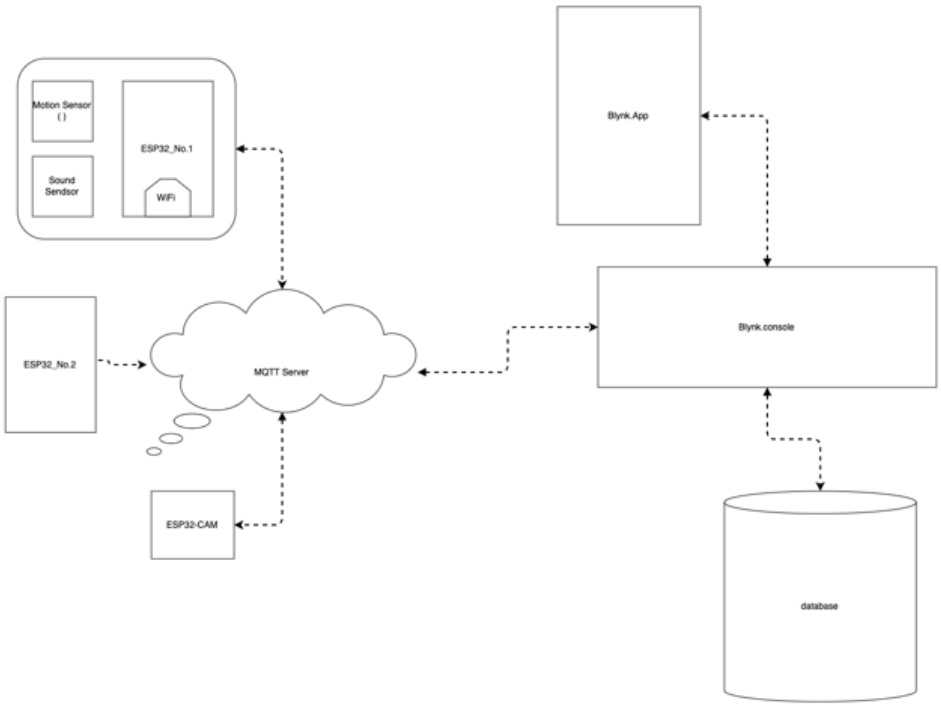


Figure 1: net work architecture

2.2 Net work architecture

Following the problem statement, we have to design the architecture base on three different device using three ESP32 MCU. And make sure all those device can communicate to each other to transport the data. And we should have a front-end mobile application to control the device and get to view the data. Afterward, we need a storage to contain the footage of the camera shoot.

Because the real situation which is the most app on the mobile device to be use fully function should be paid. And this is an academic porject that we should not to pay. So we decide to use the free version of Blynk as the mobile app UI. And in case we need use firebase web app to show the footage.

The MQTT server will subscribe the topic

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.

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.

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.

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);
```

```
92
93      Blynk.virtualWrite(V3,soundValue);
94      Blynk.virtualWrite(V4,pirValue);
95
96      if(mqttClient.connected()){
97          pubMqttSensorValueMsg();
98          if(soundValue >= 2000 || pirValue == 1){
99              buzzerTrigger = 1;
100             cameraTrigger = 1;
101             pubMqttCameraTriggerMsg();
102             pubMqttBuzzerTriggerMsg();
103         }
104     }
105 }
106
107 void connectMQTTServer() {
108 //    String clientId = "esp32-Sensor-" + WiFi.macAddress();
109     if(mqttClient.connect("ggg", "
    uszYF0QvKzAJ5kSCZByNuCbKukAMVf4fxu12kIoS7Mq1U8tHxPkRhksAsQcdV4gg
    ","")){
110         Serial.println("MQTT Service connected!");
111         Serial.println("Server address: ");
112         Serial.println(mqttService);
113     }else{
114         Serial.println("MQTT server connect fail.. ");
115         Serial.println("Client state: ");
116         Serial.println(mqttClient.state());
117         delay(3000);
118     }
119 }
120
121 void wifiConnect(){
122     WiFi.mode(WIFI_STA);
123     WiFi.begin(ssid, pass);
124     WiFi.setSleep(false);
125     Serial.println("Start to connect to wifi ..");
126
127     while (WiFi.status() != WL_CONNECTED) {
128         delay(500);
129         Serial.print(".");
130     }
131     Serial.println("");
132     Serial.println("WiFi connected");
133     Serial.println(WiFi.localIP());
134 }
135
136 BLYNK_WRITE(V0){
137     int value = param.asInt();
138     Serial.println(value);
```

```
139
140     if(value == 1){
141         digitalWrite(LEDPIN,HIGH);
142         Serial.println("led on! ");
143         timerID = timer.setInterval(200L, sendSensor);
144         timer.enable(timerID);
145     } else {
146         digitalWrite(LEDPIN, LOW);
147         timer.disable(timerID);
148     }
149 };
150
151
152 void setup() {
153     Serial.begin(115200);
154
155     pinMode(LEDPIN,OUTPUT);
156     digitalWrite(LEDPIN,LOW);
157
158     wifiConnect();
159
160     mqttClient.setServer(mqttService,1883);
161     connectMQTTServer();
162
163     Blynk.begin(BLYNK_AUTH_TOKEN,ssid,pass);
164
165 }
166
167
168 void loop() {
169     if(mqttClient.connected()){
170         mqttClient.loop();
171     }else{
172         connectMQTTServer();
173     }
174
175     Blynk.run();
176     timer.run();
177 }
```

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(){

```

```
44      String topicSensorValue = "buzzerStatus"; // here insert
    the topic which you want to publish.
45      char publishTopic[topicSensorValue.length() + 1];
46      strcpy(publishTopic, topicSensorValue.c_str());
47
48      String messageSensorValue = String(trigger); // here
    insert the message which you want to publish
49      char publishMsg[messageSensorValue.length() + 1];
50      strcpy(publishMsg, messageSensorValue.c_str());
51
52      if(mqttClient.publish(publishTopic,publishMsg)){
53          Serial.println("Topic: " + String(publishTopic));
54          Serial.println("Message: " + String(publishMsg));
55      }else{
56          Serial.println("Publish Failed!");
57      }
58  }
59
60  void receiveCallback(char* topic, byte* payload, unsigned int
      length) {
61      Serial.print("Message Received [");
62      Serial.print(topic);
63      Serial.print("]");
64
65      for (int i = 0; i < length; i++) {
66          Serial.println((char) payload[i]);
67      }
68
69      Serial.println("");
70      Serial.print("Message length(Bytes): ");
71      Serial.println(length);
72
73      if((char) payload[0] == 1){
74          buzzerTone();
75          trigger = 1;
76          pubMqttBuzzerStatueMsg();
77          Blynk.virtualWrite(V1,1);
78      } else {
79          tone(BUZZERPIN,0);
80          trigger = 0;
81          pubMqttBuzzerStatueMsg();
82      }
83  }
84
85  void subscribeTopic() {
86      String topicString = "buzzerTrigger"; // here insert the
    topic which you want to subscribed.
87      char subTopic[topicString.length() + 1];
88      strcpy(subTopic,topicString.c_str());
```

```
89
90      if(mqttClient.subscribe(subTopic)){
91          Serial.println("Subscrib Topic: ");
92          Serial.println(subTopic);
93      }else{
94          Serial.println("Sbuscribe Fail..");
95      }
96  }
97
98  void connectMQTTServer() {
99  //    String clientId = "esp32-Sensor-" + WiFi.macAddress();
100     if(mqttClient.connect("ggg", "
    uszYF0QvKzAJ5kSCZByNuCbKukAMVf4fxu12kIoS7Mq1U8tHxPkRhksAsQcdV4gg
    ","")){
101         Serial.println("MQTT Service connected!");
102         Serial.println("Server address: ");
103         Serial.println(mqttService);
104         subscribeTopic(); // subscribe the topic which this
    method have.
105     }else{
106         Serial.println("MQTT server connect fail.. ");
107         Serial.println("Client state: ");
108         Serial.println(mqttClient.state());
109         delay(3000);
110     }
111 }
112
113 void wifiConnect(){
114     WiFi.mode(WIFI_STA);
115     WiFi.begin(ssid, pass);
116     WiFi.setSleep(false);
117     Serial.println("Start to connect to wifi ..");
118
119     while (WiFi.status() != WL_CONNECTED) {
120         delay(500);
121         Serial.print(".");
122     }
123     Serial.println("");
124     Serial.println("WiFi connected");
125     Serial.println(WiFi.localIP());
126 }
127
128 BLYNK_WRITE(V1){
129     int value = param.asInt();
130     Serial.println(value);
131
132     if(value == 0){
133         tune(BUZZERPIN,0);
134         trigger = 0;
```

```
135         pubMqttBuzzerStatueMsg();
136     }
137 }
138 
139 void setup() {
140     Serial.begin(115200);
141     Serial.setDebugOutput(true);
142 
143     pinMode(BUZZERPIN, OUTPUT);
144 
145     wifiConnect();
146 
147     if (WiFi.status() == WL_CONNECTED) {
148         mqttClient.setServer(mqttService,1883);
149         mqttClient.setCallback(receiveCallback);
150         connectMQTTServer();
151     } else {
152         Serial.println("Waiting for WiFi .. ");
153     }
154 
155     Blynk.begin(BLYNK_AUTH_TOKEN,ssid,pass);
156 
157 }
158 
159 void loop() {
160     if(mqttClient.connected()){
161         mqttClient.loop();
162     }else{
163         connectMQTTServer();
164     }
165 
166     Blynk.run();
167     timer.run();
168 }
```

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
```

```
41 //#define VSYNC_GPIO_NUM    25
42 //#define HREF_GPIO_NUM     23
43 //#define PCLK_GPIO_NUM     22
44
45 // Insert Firebase project API Key
46 #define API_KEY "AIzaSyAV4fMHmipIzuH4o3etOOTfp8xCpgXHxo4"
47
48 // Insert Authorized Email and Corresponding Password
49 #define USER_EMAIL "feix0033@easv365.dk"
50 #define USER_PASSWORD "12345678"
51
52 // Insert Firebase storage bucket ID e.g bucket-name.appspot.
   com
53 #define STORAGE_BUCKET_ID "esp32-smartpants.appspot.com"
54
55 // Photo File Name to save in SPIFFS
56 #define FILE_PHOTO "/data/photo.jpg"
57
58 // Wi-Fi ssid and password
59 const char* ssid = "Evensnachi";
60 const char* pass = "12345678";
61 const char* mqttService = "mqtt.flespi.io";
62
63 WiFiClient wifiClient;
64 PubSubClient mqttClient(wifiClient);
65
66 BlynkTimer timer;
67 int v3Value;
68 int takeNewPhoto = 0;
69 bool taskCompleted = false;
70
71 //Define Firebase Data objects
72 FirebaseData fbdo;
73 FirebaseAuth auth;
74 FirebaseConfig configF;
75
76 // Check if photo capture was successful
77 bool checkPhoto(fs::FS&fs){
78     File f_pic = fs.open(FILE_PHOTO);
79     unsigned int pic_sz = f_pic.size();
80     return ( pic_sz > 100 );
81 }
82
83 // Capture Photo and Save it to SPIFFS
84 void capturePhotoSaveSpiffs( void ) {
85     camera_fb_t * fb = NULL; // pointer
86     bool ok = 0; // Boolean indicating if the picture has
   been taken correctly
87     do {
```

```
88          // Take a photo with the camera
89          Serial.println("Taking a photo...");
90
91          fb = esp_camera_fb_get();
92          if (!fb) {
93              Serial.println("Camera capture failed");
94              return;
95          }
96          // Photo file name
97          Serial.printf("Picture file name: %s\n", FILE_PHOTO);
98          File file = SPIFFS.open(FILE_PHOTO, FILE_WRITE);
99          // Insert the data in the photo file
100         if (!file) {
101             Serial.println("Failed to open file in writing
    mode");
102         }
103         else {
104             file.write(fb->buf, fb->len); // payload (image),
     payload length
105             Serial.print("The picture has been saved in ");
106             Serial.print(FILE_PHOTO);
107             Serial.print(" - Size: ");
108             Serial.print(file.size());
109             Serial.println(" bytes");
110         }
111         // Close the file
112         file.close();
113         esp_camera_fb_return(fb);
114
115         // check if file has been correctly saved in SPIFFS
116         ok = checkPhoto(SPIFFS);
117     } while ( !ok );
118 }
119
120 void pubMqttCamStatus(){
121
122     String topicSensorValue = "cameraStatus";
123     char publishTopic[topicSensorValue.length() + 1];
124     strcpy(publishTopic, topicSensorValue.c_str());
125
126     String messageSensorValue = String(1);
127     char publishMsg[messageSensorValue.length() + 1];
128     strcpy(publishMsg, messageSensorValue.c_str());
129
130     if(mqttClient.publish(publishTopic,publishMsg)){
131         Serial.println("Topic: " + String(publishTopic));
132         Serial.println("Message: " + String(publishMsg));
133     }else{
134         Serial.println("Publish Failed!");
```

```
135     }
136 }
137
138 void pubMqttCamIpMsg() {
139
140     String topicSensorValue = "cameraLink";
141     char publishTopic[topicSensorValue.length() + 1];
142     strcpy(publishTopic, topicSensorValue.c_str());
143
144     String messageSensorValue = WiFi.localIP().toString();
145     char publishMsg[messageSensorValue.length() + 1];
146     strcpy(publishMsg, messageSensorValue.c_str());
147
148     if(mqttClient.publish(publishTopic,publishMsg)){
149         Serial.println("Topic: " + String(publishTopic));
150         Serial.println("Message: " + String(publishMsg));
151     }else{
152         Serial.println("Publish Failed!");
153     }
154 }
155
156 void startCameraServer();
157
158 void receiveCallback(char* topic, byte* payload, unsigned int
    length) {
159     Serial.print("Message Received [");
160     Serial.print(topic);
161     Serial.print("]");
162
163     for (int i = 0; i < length; i++) {
164         Serial.println((char) payload[i]);
165     }
166
167     Serial.println("");
168     Serial.print("Message length(Bytes): ");
169     Serial.println(length);
170
171     if ((char) payload[0] == 1) {
172         Serial.println("Start camera server");
173         startCameraServer();
174
175         Serial.print("Camera Ready! Use 'http://");
176         Serial.print(WiFi.localIP());
177         Serial.println("' to connect");
178
179         Blynk.virtualWrite(V2,1);
180         Blynk.virtualWrite(V5,WiFi.localIP().toString());
181
182         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..");
    }
}
```

```
222
223 void connectMQTTServer() {
224
225     if(mqttClient.connect("ggg", "
    uszYF0QvKzAJ5kSCZByNuCbKukAMVf4fxu12kIoS7Mq1U8tHxPkRhksAsQcdV4gg
    ","")){
226         Serial.println("MQTT Service connected!");
227         Serial.println("Server address: ");
228         Serial.println(mqttService);
229         subscribeTopic(); // subscribe the topic which this
    method have.
230     }else{
231         Serial.println("MQTT server connect fail.. ");
232         Serial.println("Client state: ");
233         Serial.println(mqttClient.state());
234         delay(3000);
235     }
236 }
237
238 void wifiConnect(){
239     WiFi.mode(WIFI_STA);
240     WiFi.begin(ssid, pass);
241     WiFi.setSleep(false);
242     Serial.println("Start to connect to wifi ..");
243
244     while (WiFi.status() != WL_CONNECTED) {
245         delay(500);
246         Serial.print(".");
247     }
248     Serial.println("");
249     Serial.println("WiFi connected");
250     Serial.println(WiFi.localIP());
251 }
252
253 void initSPIFFS(){
254     if (!SPIFFS.begin(true)) {
255         Serial.println("An Error has occurred while mounting
    SPIFFS");
256         ESP.restart();
257     }
258     else {
259         delay(500);
260         Serial.println("SPIFFS mounted successfully");
261     }
262 }
263
264 void cameraInitProcess() {
265     camera_config_t config;
266     config.ledc_channel = LEDC_CHANNEL_0;
```

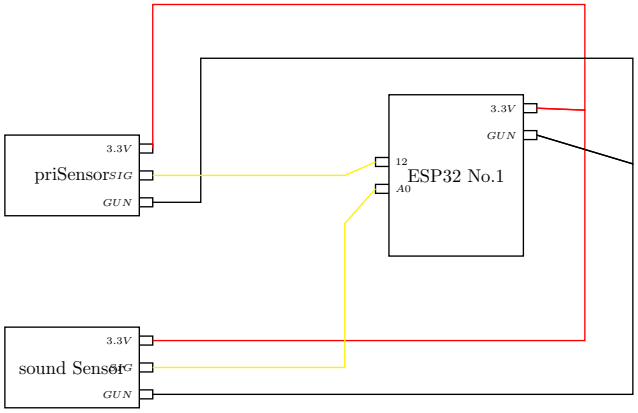
```
267     config.ledc_timer = LEDC_TIMER_0;
268     config.pin_d0 = Y2_GPIO_NUM;
269     config.pin_d1 = Y3_GPIO_NUM;
270     config.pin_d2 = Y4_GPIO_NUM;
271     config.pin_d3 = Y5_GPIO_NUM;
272     config.pin_d4 = Y6_GPIO_NUM;
273     config.pin_d5 = Y7_GPIO_NUM;
274     config.pin_d6 = Y8_GPIO_NUM;
275     config.pin_d7 = Y9_GPIO_NUM;
276     config.pin_xclk = XCLK_GPIO_NUM;
277     config.pin_pclk = PCLK_GPIO_NUM;
278     config.pin_vsync = VSYNC_GPIO_NUM;
279     config.pin_href = HREF_GPIO_NUM;
280     config.pin_sscb_sda = SIOD_GPIO_NUM;
281     config.pin_sscb_scl = SIOC_GPIO_NUM;
282     config.pin_pwdn = PWDN_GPIO_NUM;
283     config.pin_reset = RESET_GPIO_NUM;
284     config.xclk_freq_hz = 20000000;
285     config.frame_size = FRAMESIZE_UXGA;
286     config.pixel_format = PIXFORMAT_JPEG; // for streaming
287     //config.pixel_format = PIXFORMAT_RGB565; // for face
    detection/recognition
288 //    config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
289 //    config.fb_location = CAMERA_FB_IN_PSRAM;
290 //    config.jpeg_quality = 12;
291 //    config.fb_count = 1;
292 
293 //    // if PSRAM IC present, init with UXGA resolution and
    higher JPEG quality
294 //    //                      for larger pre-allocated frame
    buffer.
295 //    if(config.pixel_format == PIXFORMAT_JPEG){
296 //        if(psramFound()){
297 //            config.jpeg_quality = 10;
298 //            config.fb_count = 2;
299 //            config.grab_mode = CAMERA_GRAB_LATEST;
300 //        } else {
301 //            // Limit the frame size when PSRAM is not
    available
302 //            config.frame_size = FRAMESIZE_SVGA;
303 //            config.fb_location = CAMERA_FB_IN_DRAM;
304 //        }
305 //    } else {
306 //        // Best option for face detection/recognition
307 //        config.frame_size = FRAMESIZE_240X240;
308 //#if CONFIG_IDF_TARGET_ESP32S3
309 //        config.fb_count = 2;
310 //#endif
311 //    }
```

```
312 //
313 //#if defined(CAMERA_MODEL_ESP_EYE)
314 //    pinMode(13, INPUT_PULLUP);
315 //  pinMode(14, INPUT_PULLUP);
316 //#endif
317 //
318 //    // camera init
319 //    esp_err_t err = esp_camera_init(&config);
320 //    if (err != ESP_OK) {
321 //        Serial.printf("Camera init failed with error 0x%x",
    err);
322 //        return;
323 //    }
324 //
325 //    sensor_t * s = esp_camera_sensor_get();
326 //    // initial sensors are flipped vertically and colors
   are a bit saturated
327 //    if (s->id.PID == OV3660_PID) {
328 //        s->set_vflip(s, 1); // flip it back
329 //        s->set_brightness(s, 1); // up the brightness just
   a bit
330 //        s->set_saturation(s, -2); // lower the saturation
331 //    }
332 //    // drop down frame size for higher initial frame rate
333 //    if(config.pixel_format == PIXFORMAT_JPEG){
334 //        s->set_framesize(s, FRAMESIZE_QVGA);
335 //    }
336 //
337 //#if defined(CAMERA_MODEL_M5STACK_WIDE) || defined(
   CAMERA_MODEL_M5STACK_ESP32CAM)
338 //    s->set_vflip(s, 1);
339 //  s->set_hmirror(s, 1);
340 //#endif
341 //
342 //#if defined(CAMERA_MODEL_ESP32S3_EYE)
343 //    s->set_vflip(s, 1);
344 //#endif
345     if (psramFound()) {
346         config.frame_size = FRAMESIZE_UXGA;
347         config.jpeg_quality = 10;
348         config.fb_count = 2;
349     } else {
350         config.frame_size = FRAMESIZE_SVGA;
351         config.jpeg_quality = 12;
352         config.fb_count = 1;
353     }
354     // Camera init
355     esp_err_t err = esp_camera_init(&config);
356     if (err != ESP_OK) {
```

```
357         Serial.printf("Camera init failed with error 0x%x",
    err);
358         ESP.restart();
359     }
360 
361 }
362 
363 BLYNK_WRITE(V6){
364     takeNewPhoto = param.asInt();
365 }
366 
367 void setup() {
368     Serial.begin(115200);
369     Serial.setDebugOutput(true);
370 
371     initSPIFFS();
372     WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
373     cameraInitProcess();
374     wifiConnect();
375 
376     if (WiFi.status() == WL_CONNECTED) {
377         Serial.println("Start camera service .. ");
378 
379         mqttClient.setServer(mqttService, 1883); //set the
    mqtt service to connect
380         mqttClient.setCallback(receiveCallback); //set the
    callback method to keep running the method witch can
    receive mqtt message.
381         connectMQTTServer(); //connect to the mqtt server
382 
383         //Firebase
384         // Assign the api key
385         configF.api_key = API_KEY;
386         //Assign the user sign in credentials
387         auth.user.email = USER_EMAIL;
388         auth.user.password = USER_PASSWORD;
389         //Assign the callback function for the long running
    token generation task
390         configF.token_status_callback = tokenStatusCallback;
    //see addons/TokenHelper.h
391 
392         Firebase.begin(&configF, &auth);
393         Firebase.reconnectWiFi(true);
394 
395     }else {
396         Serial.println("Waiting for WiFi .. ");
397     }
398 
399     Blynk.begin(BLYNK_AUTH_TOKEN, ssid, pass);
```

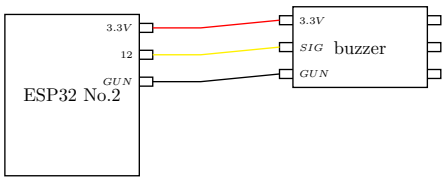
```
400 }
401
402 void loop() {
403     if(mqttClient.connected()){
404         mqttClient.loop();
405     }else{
406         connectMQTTServer();
407     }
408
409     Blynk.run();
410     timer.run();
411 //     delay(10000);
412 }
```

Figure 2: ESP32 no.1 connect to sensors



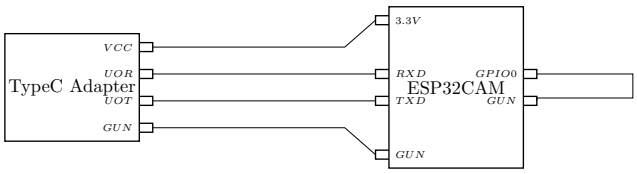
1

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



1

Figure 4: ESP32 CAM



1