**باسمه تعالی**

****

**امنیت در اینترنت اشیاء**

**دکتر احمدی**

**تمرین سوم**

**پیاده سازی MQTT با ESP**

**پوریا دادخواه**

**401201381**

1. اتصال ESP32 به بروکر

1.1 راه‌اندازی بروکر

در این قسمت ابتدا یک بروکرmosquitto با کانفیگ دلخواه mosquito\_esp.conf روی ویندوز بالا می‌آوریم. برای این بخش از تمرین بروکر باید موارد زیر را هندل کند:

1. پورت 1883 را پشتیبانی کند.
2. سایر دستگاه‌های متصل به شبکه محلی سیستم ویندوز ( که localhost نیستند) را نیز شناسایی و متصل کند.
3. نیازی به authentication از کاربران خارجی نداشته‌ باشد.

به این صورت کانفیگ زیر را که به همه رنج IP های موجود در شبکه محلی سیستم و روی پورت 1883 گوش می‌دهد را اعمال می‌کنیم: ( این کانفیگ در پوشه تمرین با نام mosquito\_esp.conf ضمیمه شده‌است.)

listener 1883 0.0.0.0

allow\_anonymous true

پس از راه‌اندازی بروکر باید client esp32 را آماده کنیم. پایه کد پیاده‌سازی مورد استفاده در این قسمت کد آماده mqtt\_tcp از کدهای نمونه espressif استفاده کردیم که در ادامه توابع مهم این کد را توضیح مختصر می‌دهیم:

1.2 توضیح ساختار اصلی کد mqtt tcp

mqtt\_event\_handler

این تابع پس از اجرای تابع اصلی اجرای mqtt فراخوانده می‌شود و همان‌طور که از نامش پیدا می‌باشد وظیفه هندل کردن و واکنش مناسب بر اساس رویدادهای اتفاق افتاده در سمت esp را دارد. در این تابع بسته به نوع رویداد ( مثل subscribe, publish, connect, disconnect, error) esp در پروتکل mqtt واکنشی که برای آن تنظیم می‌کنیم را نشان می‌دهد.

mqtt\_app\_start

این تابع اصلی اجرای mqtt بر روی esp می‌باشد. در این قسمت باید اطلاعات بروکر مدنظر و کانفیگ مربوطه برای اتصال ( مانند اطلاعات مربوط به احراز هویت client و ssl و ... ) به کانفیگ client مربوطه دهیم و یک client روی esp تعریف کنیم.

app\_main

در تابع اجرای برنامه هم پس از اتصال wifi، تابع mqtt\_app\_start را فراخوانده و اتصال را شروع می‌کنیم.

درضمن در هرقسمت از توابع فوق LOG های مناسب برای debugging بهتر نوشته شده‌اند که هرگونه اخطار منطقی یا پروتکلی را به ما نشان دهند.

* 1. شخصی‌سازی کد mqtt client

در این قسمت ابتدا باید آدرس بروکر را به کانفیگ esp mqtt client بدهیم و از آنجایی که ssl ,authentication یا .. برای این بخش از تمرین نیاز نداریم، صرفا همین آدرس کفایت می‌کند. آدرس مدنظر را نیز کافیست از ipconfig سیستم خود استخراج کنیم که ip wifi سیستم می‌باشد.

    esp\_mqtt\_client\_config\_t mqtt\_cfg = {

        .broker.address.uri = "mqtt://172.27.213.96:1883",

    };

سپس کافی است پس از اتصال mqtt، یعنی رویداد mqtt connect، کلاینت esp در تاپیک my\_topic پیغام خواسته شده را منتشر بکند: ( برای دریافت این پیام توسط خود کلاینت esp علاوه بر کلاینت‌های دیگر، خود esp را نیز در این تاپیک subscribe کردیم)

    case MQTT\_EVENT\_CONNECTED:

        ESP\_LOGI(TAG, "MQTT\_EVENT\_CONNECTED");

        msg\_id = esp\_mqtt\_client\_subscribe(client, "my\_topic/#", 0);

        ESP\_LOGI(TAG, "sent subscribe successful, msg\_id=%d", msg\_id);

        esp\_mqtt\_client\_publish(client, "my\_topic", "Hello! This is Pouria Dadkhah with 401201381, Enjoying HW3 of SIOT course :))", 0, 0, 0);

        ESP\_LOGI(TAG, "sent message successful");

        break;

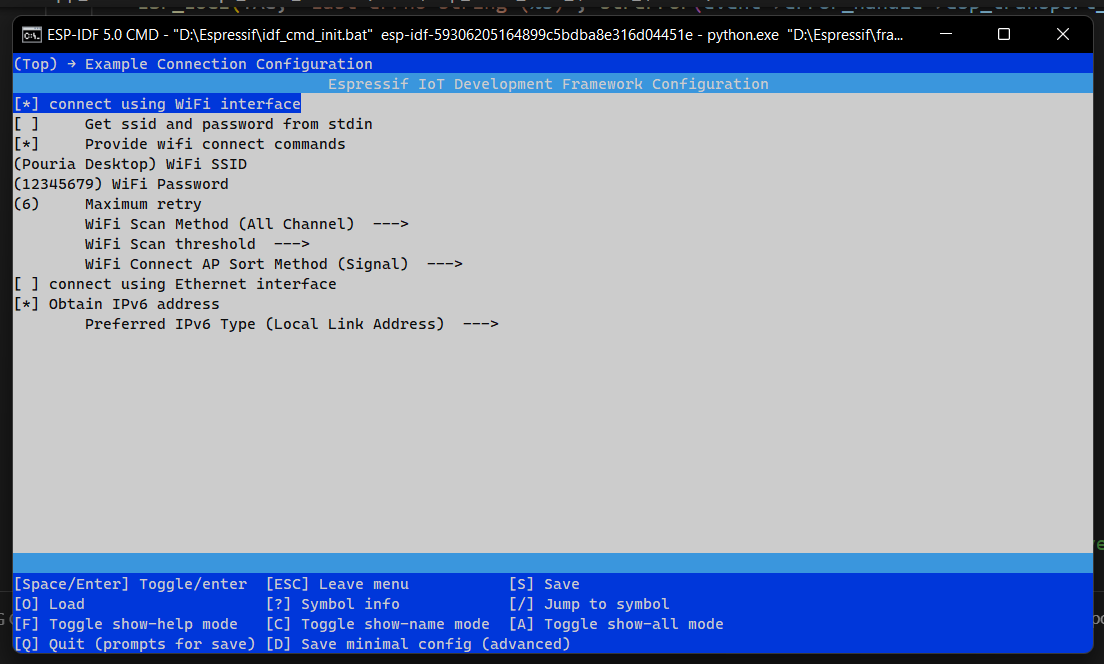
اکنون کد فوق آماده بوده و آن را build و روی برد flash می‌کنیم.

توجه می‌کنیم پیش از موارد گفته شده باید menuconfig برد را به درستی تنظیم کنیم که اتصال برقرار شود. به این منظور موارد زیر را در آن ست می‌کنیم:

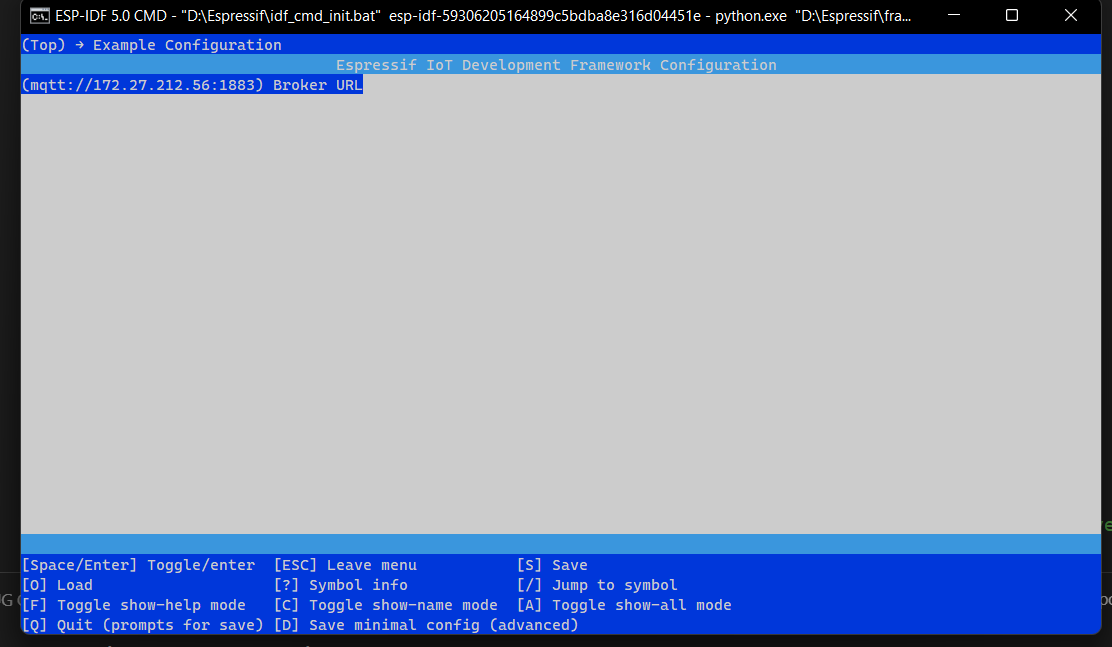
* اطلاعات ssid و password وای‌فای که باید به آن متصل شود. ( هرچند امکان ست کردن این مورد در کد هم وجود داشت ولی این بخش را دستی از طریق menuconfig ست کرده‌ایم)

دقت می‌کنیم در این تمرین esp و هر دستگاه خارجی که قصد اتصال به بروکر سیستم را دارد را به hotspot ویندوز متصل می‌کنیم که خود سیستم نقش روتر را بازی کند. دلیل این امر هم این هست که دسترسی به wifi عادی که بتوان سیستم وesp را ذیل آن اجرا کرد وجود نداشته است. Wifi دانشگاه نیاز به لاگین داشته و IP قابل دسترس به esp نمی‌دهد ( البته شاید از طریق Ethernet امکان اتصال وجود داشته باشد که راه سخت‌تری است) راه دیگر هم hotspot گوشی است که سیستم و esp را به آن متصل کنیم. این روش هم بنا به الگوریتم اختصاص IP توسط hotspot گوشی که دستگاه‌ها را در zone های مختلفی قرار می‌دهد جوابگو نبود.

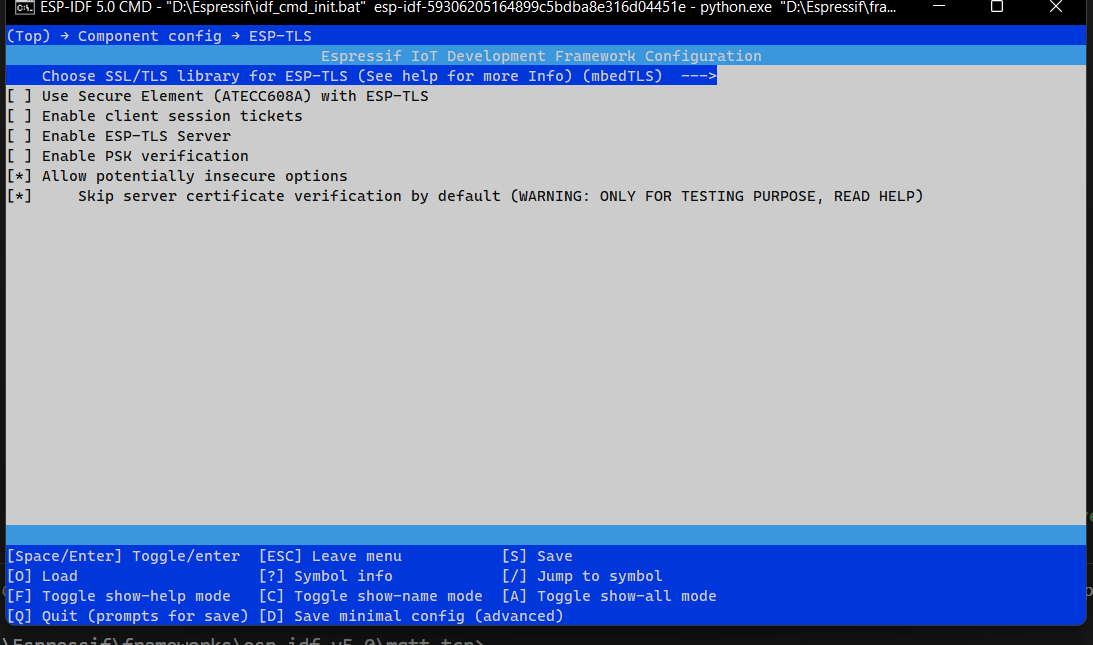
* از آن‌جایی که در این تمرین قصد امن کردن ارتباط از طریق ssl را نداریم، می‌توان برای جلوگیری از بروز خطای esp-tls، اجازه امکان اتصال غیر امن را به پروتکل بدهیم که مراحل آن در تصاویر ادامه آمده است.



مشخصات اتصال به شبکه wifi

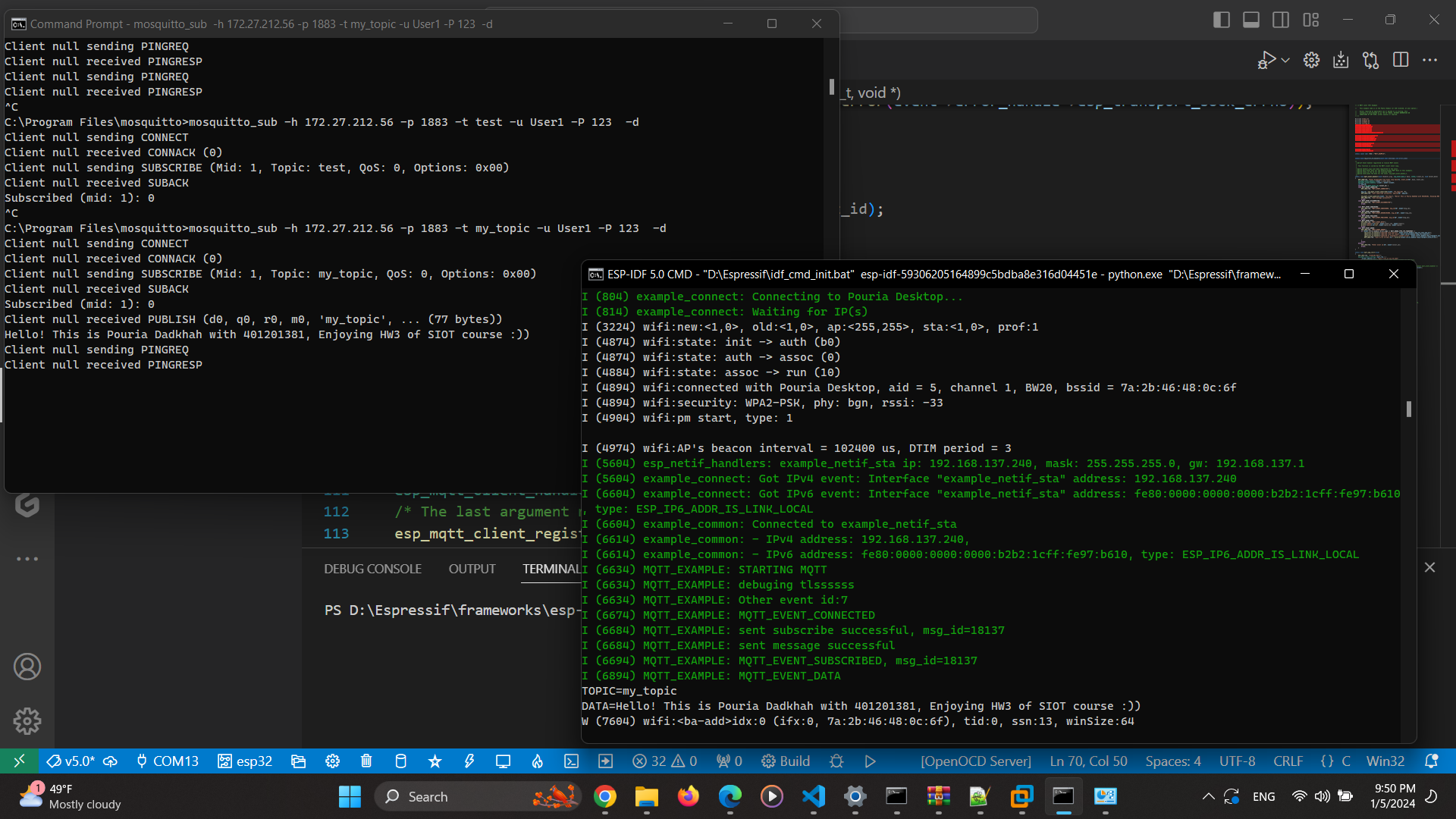


می‌توان آدرس بروکر را در این قسمت نیز مستقیما وارد کرد



اجازه اتصال ناامن

در آخر پس از build flash monitor کد، نتیجه ارسال و دریافت پیام خواسته شده را توسط esp و کلاینت دیگر cmd می‌بینیم:



ارسال پیام توسط esp در تاپیک my\_topic و دریافت توسط client cmd که در آن تاپیک subscribe کرده است.

1. کنترل LED

در این قسمت client esp در تاپیک SIOT/+/LED subscribe می‌کند. سایر کلاینت‌ها باید در تاپیک متناظر شماره‌دانشجویی خود در فرمت گفته شده، پیغامی مبنی بر on,off کردن led ارسال کنند و همه این پیام‌ها توسط client esp دریافت می‌شود. اما در قسمت تحلیل داده mqtt و یا متناظرا رویداد mqtt\_event\_data، باید با بررسی تاپیک دریافتی، اقدام به واکنش مناسب بکنیم. در ادامه ابتدا ساز و کار کانفیگ بروکر و سپس کد پیاده‌سازی client esp را توضیح می‌دهیم.

2.1 کانفیگ بروکر

در این قسمت نیازی به احراز هویت کاربران نبود و این نیاز مربوط به بخش بعدی می‌شد اما در همین قسمت ساز و کار user pass را برای کاربران ایجاد کردیم. در این راستا همانند روش گفته شده در تمرین قبل 3 کاربر در passwd تعریف کردیم که اجازه اتصال دارند. این فایل در تمرین ضمیمه شده است. (پسورد هر سه 123 می‌باشد)

Pouria:$7$101$FaMDcr1/arrP4vYo$rkWbfOvy7Hz1wqSAyCInmGuFNC3D7CmUa/GPu85qpnaOSC+kTQYqJ8kgyqshN1mt6bpxCgf6ZQT4TpUN9Ud0EQ==

espClient:$7$101$p+k/rFbKmtp3KaGa$xxt31A+TpD9HVD5OfYe8oXqs1HObdE/bcCbnxCmOc5vDscPWj1aW0MV8rzVnsA5ehrKn/2H5virmu5jCarAIEA==

Amin:$7$101$xhdFrd5T40iTkMOO$4lRKKtTlYtH5zoGSIU4xXKIKf+6HmGFaOh6yhcuTJp5MVTy2G3CPjvty/M1Vx52nLfwsAScFjVed13qlZTR/Bg==

و کانفیگ بروکر را نیز با نام esp\_led آورده ایم که در آن آدرس فایل passwd را داده و اجازه اتصال anonymous را غیر فعال کرده‌ایم.

2.2 کد esp

ابتدا کتابخانه gpio برای هندل کردن led را اضافه کرده و پین مربوط به led را ست می‌کنیم:

#include <driver/gpio.h>

// Define the GPIO pin to which the LED is connected

#define LED\_PIN GPIO\_NUM\_2

سپس در تاپیک گفته شده subscribe می‌کنیم که پیام همه دانشجویان را دریافت کنیم: ( جهت تست، خود esp client هم در تاپیک صاحب led - خود بنده ☺ - پیامی مبتنی بر خاموش کردن ارسال می‌کند .

 msg\_id = esp\_mqtt\_client\_subscribe(client, "SIOT/+/LED", 0);

        ESP\_LOGI(TAG, "sent subscribe successful, msg\_id=%d", msg\_id);

        esp\_mqtt\_client\_publish(client, "SIOT/401201381/LED", "off", 0, 0, 0);

        ESP\_LOGI(TAG, "sent message successful");

اکنون در رویداد دریافت data، ابتدا از تاپیک دریافتی قسمت شماره دانشجویی را استخراج کرده و در صورت برابر بودن با شماره دانشجویی صاحب ( 401201381) به اعمال وضعیت خواسته شده برای LED می‌پردازیم. درغیر این صورت پیام دسترسی غیر مجاز را نشان می‌دهیم. جزییات این مراحل را در کد زیر با کامنت‌های مناسب می‌توان تحلیل کرد:

    case MQTT\_EVENT\_DATA:

        ESP\_LOGI(TAG, "MQTT\_EVENT\_DATA");

        printf("TOPIC=%.\*s\r\n", event->topic\_len, event->topic);

        printf("DATA=%.\*s\r\n", event->data\_len, event->data);

        // ----- process input data ----------

        char student\_id[10];

        int pos = 0;

        int start\_pos = -1;

        // Iterate through the topic to find the position of the second slash

        for (int i = 0; i < event->topic\_len; i++)

        {

            if (event->topic[i] == '/')

            {

                if (start\_pos >= 0)

                {

                    // Found the second slash, end the extraction

                    student\_id[pos] = '\0';

                    break;

                }

                // Found the first slash, start extracting from the next character

                start\_pos = i + 1;

            }

            else if (start\_pos >= 0)

            {

                // Extract the Student ID

                student\_id[pos++] = event->topic[i];

            }

        }

        ESP\_LOGI(TAG, "%s", student\_id);

        // Check ownership and control LED owner student id is 401201381

        if (strcmp(student\_id, "401201381") == 0)

        {

            // Ensure the received data is null-terminated

            char payload[event->data\_len + 1];

            memcpy(payload, event->data, event->data\_len);

            payload[event->data\_len] = '\0';

            if (strcmp(payload, "on") == 0)

            {

                ESP\_LOGI(TAG, "MQTT Command: Turnining on the LED");

                // Turn on the LED

                gpio\_set\_level(LED\_PIN, 1);

                ESP\_LOGI(TAG, "LED turned ON");

            }

            else if (strcmp(payload, "off") == 0)

            {

                ESP\_LOGI(TAG, "MQTT Command: Turnining off the LED");

                // Turn off the LED

                gpio\_set\_level(LED\_PIN, 0);

                ESP\_LOGI(TAG, "LED turned OFF");

            }

        }

        else

        {

            ESP\_LOGI(TAG, "UnAuthourized MQTT Command: You don't have access!");

        }

        // --------- end of proccess --------------------

بخش دیگری که باید در کد اضافه کنیم، کانفیگ‌های مناسب دسترسی به LED است که در قسمت بالا آن را با مقدار خواسته شده فراخوانی کرده‌ایم:

// Function to configure GPIO for the LED

static void configure\_gpio()

{

    gpio\_config\_t io\_conf = {

        .pin\_bit\_mask = (1ULL << LED\_PIN),

        .mode = GPIO\_MODE\_OUTPUT,

        .intr\_type = GPIO\_INTR\_DISABLE,

        .pull\_up\_en = GPIO\_PULLUP\_DISABLE,

        .pull\_down\_en = GPIO\_PULLDOWN\_DISABLE,

    };

    gpio\_config(&io\_conf);

}

در نهایت در قسمت تعریف client باید فیلدهای user pass را نیز که زیرمجموعه بخش credentials از config هستند را اضافه بکنیم:

    configure\_gpio();  // Configure GPIO for LED

    ESP\_LOGI(TAG, "STARTING MQTT");

    esp\_mqtt\_client\_config\_t mqtt\_cfg = {

        .broker.address.uri = "mqtt://172.27.213.96:1883",

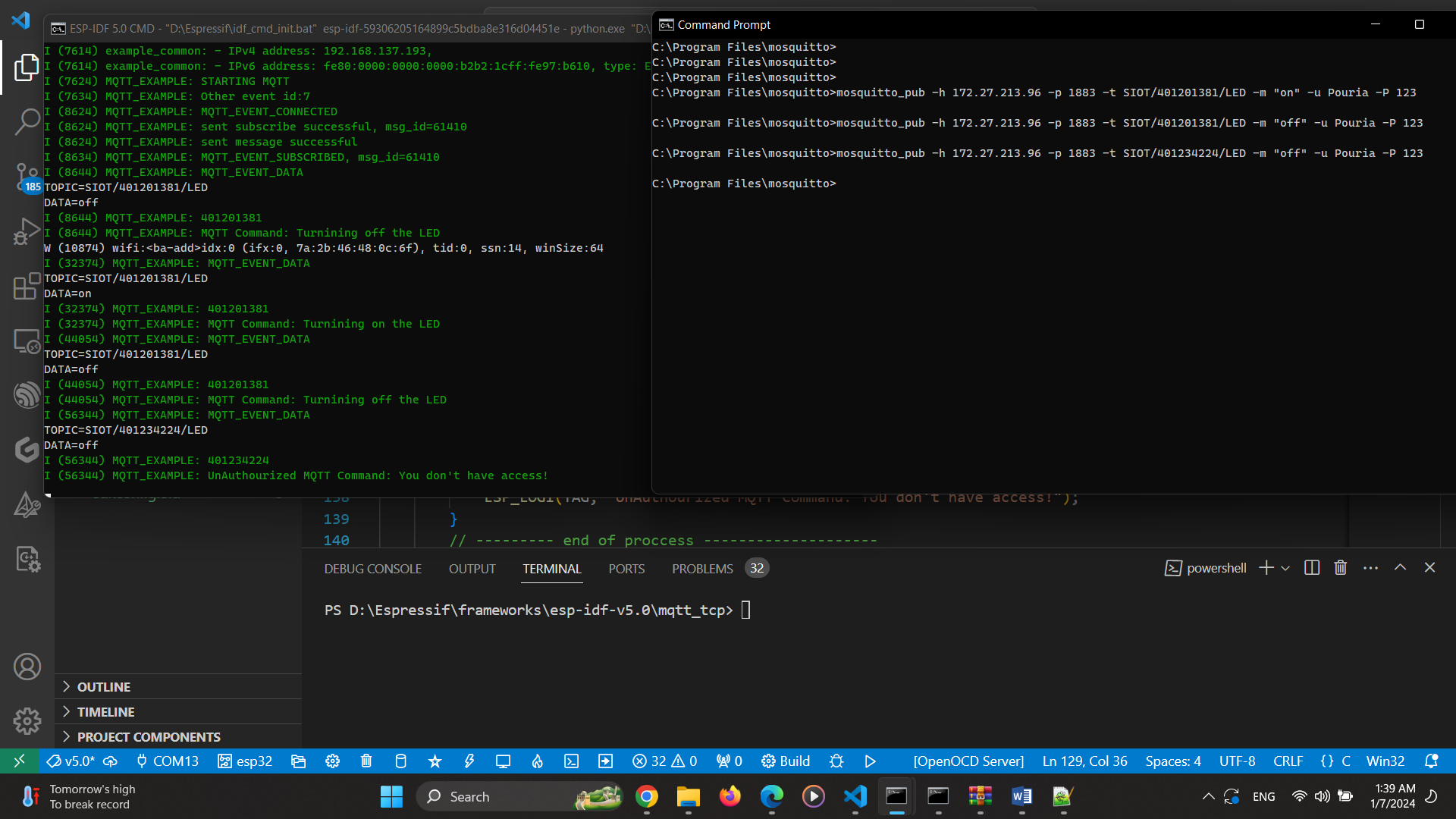
        .credentials.username="espClient",/\*your username\*/ //your username

        .credentials.authentication.password="123"

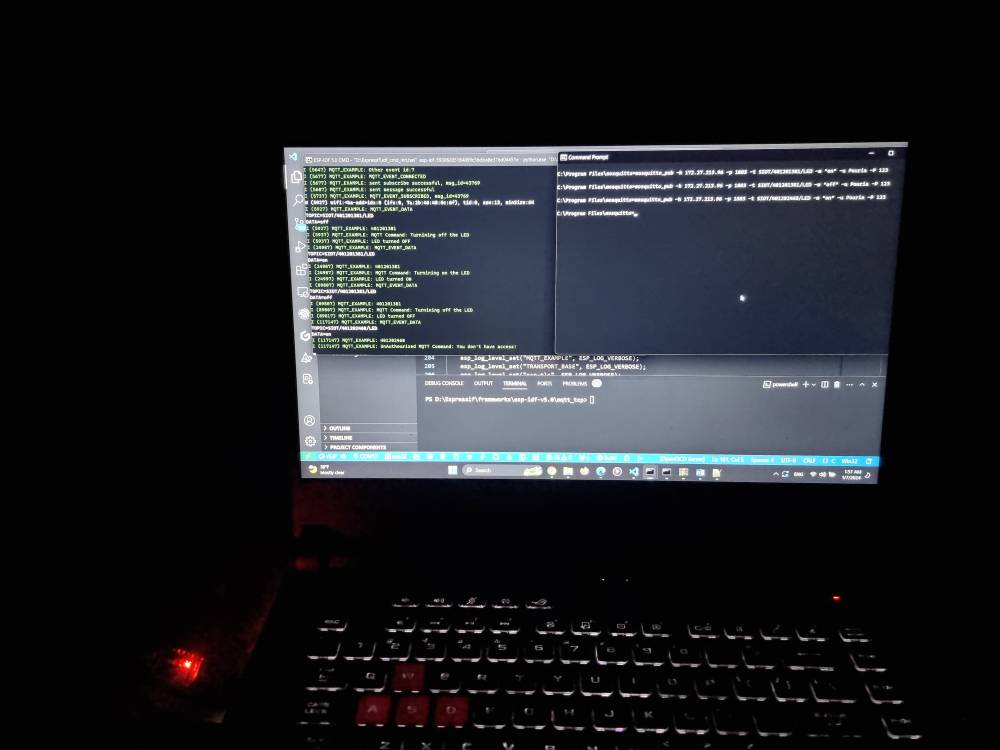
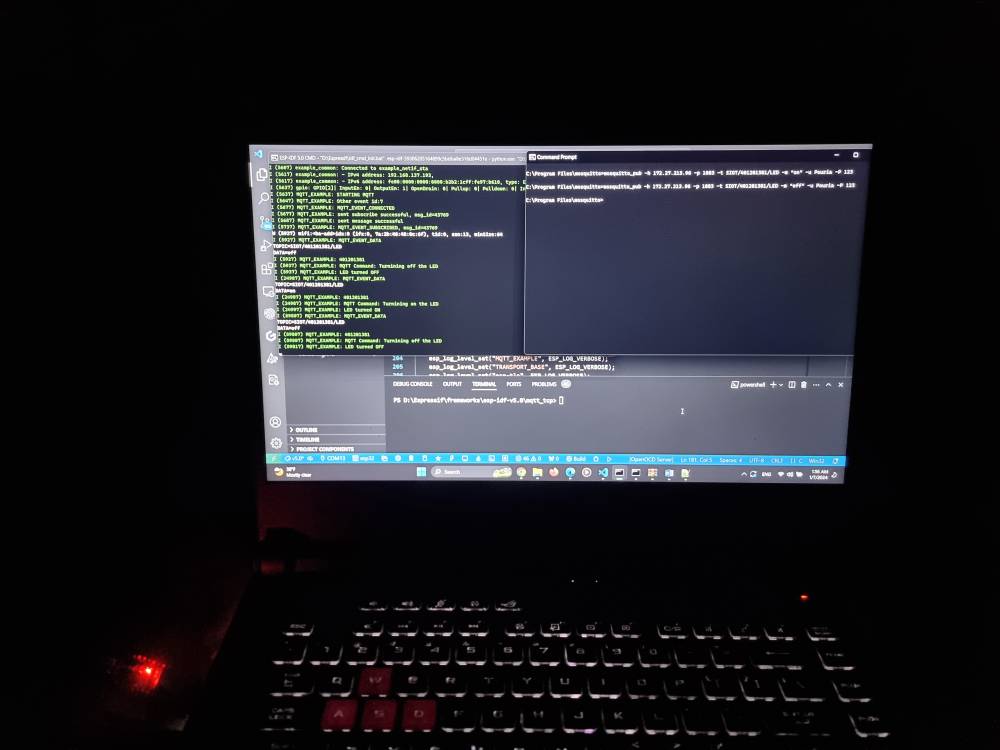
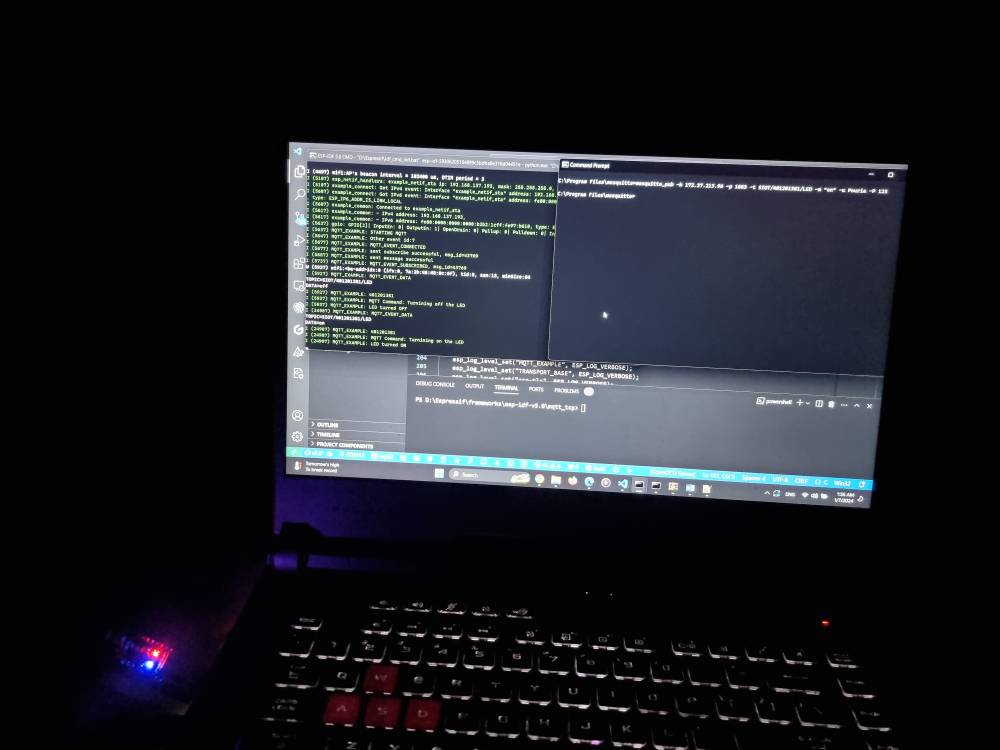
    };

پس از اجرای کد فوق چند نمونه از publish های سایر کلاینت‌هایی که با نام کاربری و رمز خود وارد شده و سعی در روشن خاموش کردن را داشته اند و واکنش esp را می‌بینیم:

( عکس‌های با کیفیت بیشتر به صورت اسکرین‌شات و عکس‌های led اورده شده اند)



اقدام به روشن خاموش کردن در تاپیک‌های مختلف و واکنش esp به آن



وضعیت چراغ که به درستی روشن، خاموش و بدون تغییر مانده است.

1. ارتقا امنیتی

بخشی از ارتقا خواسته شده، در بخش قبل اعمال شد که همان authentication کاربران از جمله esp می‌باشد. در این قسمت با اضافه کردن سطح کنترل دسترسی، این امنیت را افزایش می‌دهیم.

مطابق تمرین قبل فایل acl.acl را تعریف کرده و در آن کاربر esp فقط اجازه خواندن و نه نوشتن را دارد ( در همه SIOT/+/LED)

همچنین Pouria و Amin نیز تنها در تاپیک شماره دانشجویی خود توانایی خواندن نوشتن را دارند. فایل کلی به صورت زیر می‌شود:

# acl.acl

# User "Pouria" permissions

user Pouria

topic readwrite SIOT/401201381/LED

# User "Amin" permissions

user Amin

topic readwrite SIOT/401200224/LED

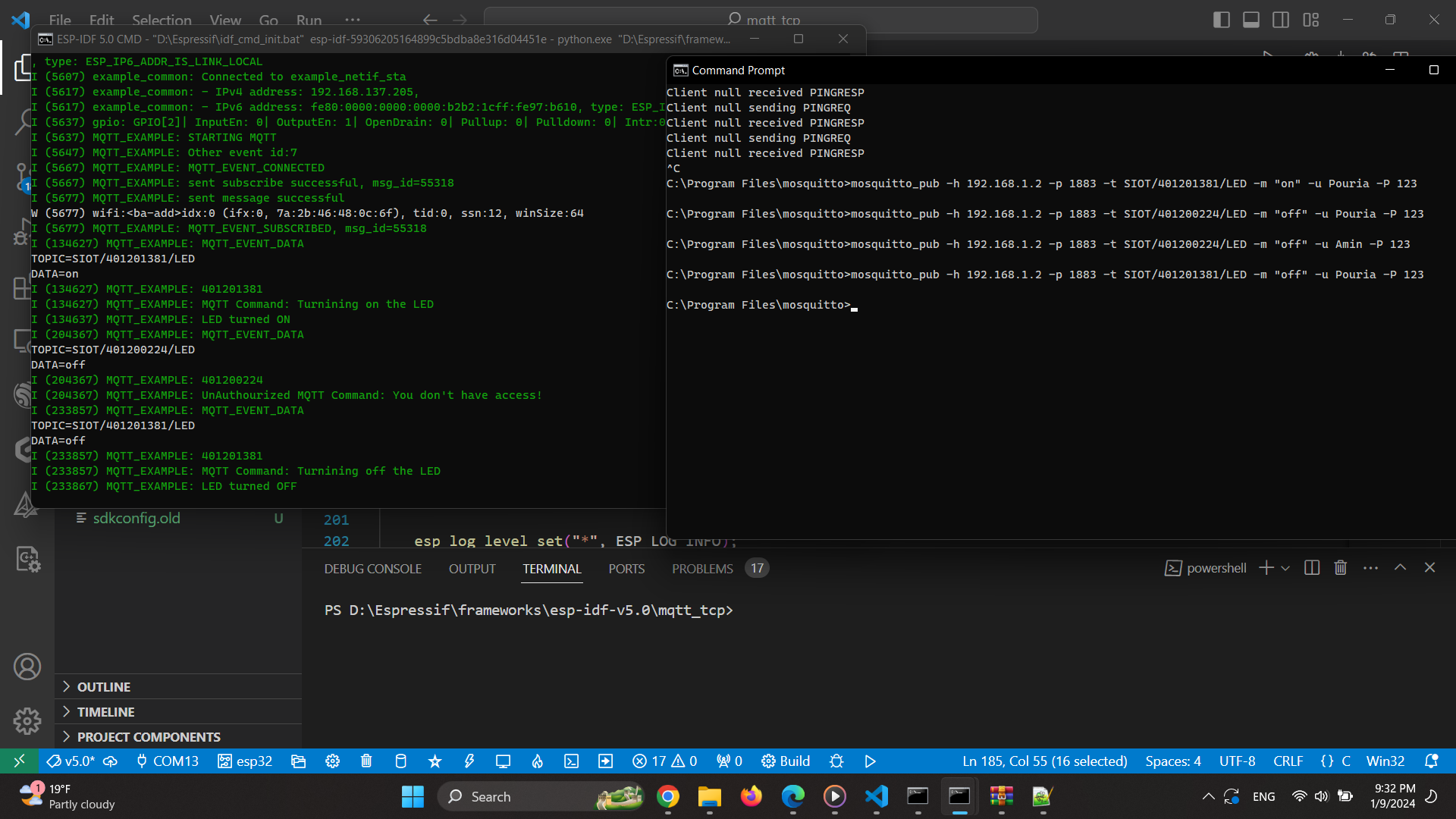
# User "espClient" permissions. it can only read over all students commands but can not write on any off them

user espClient

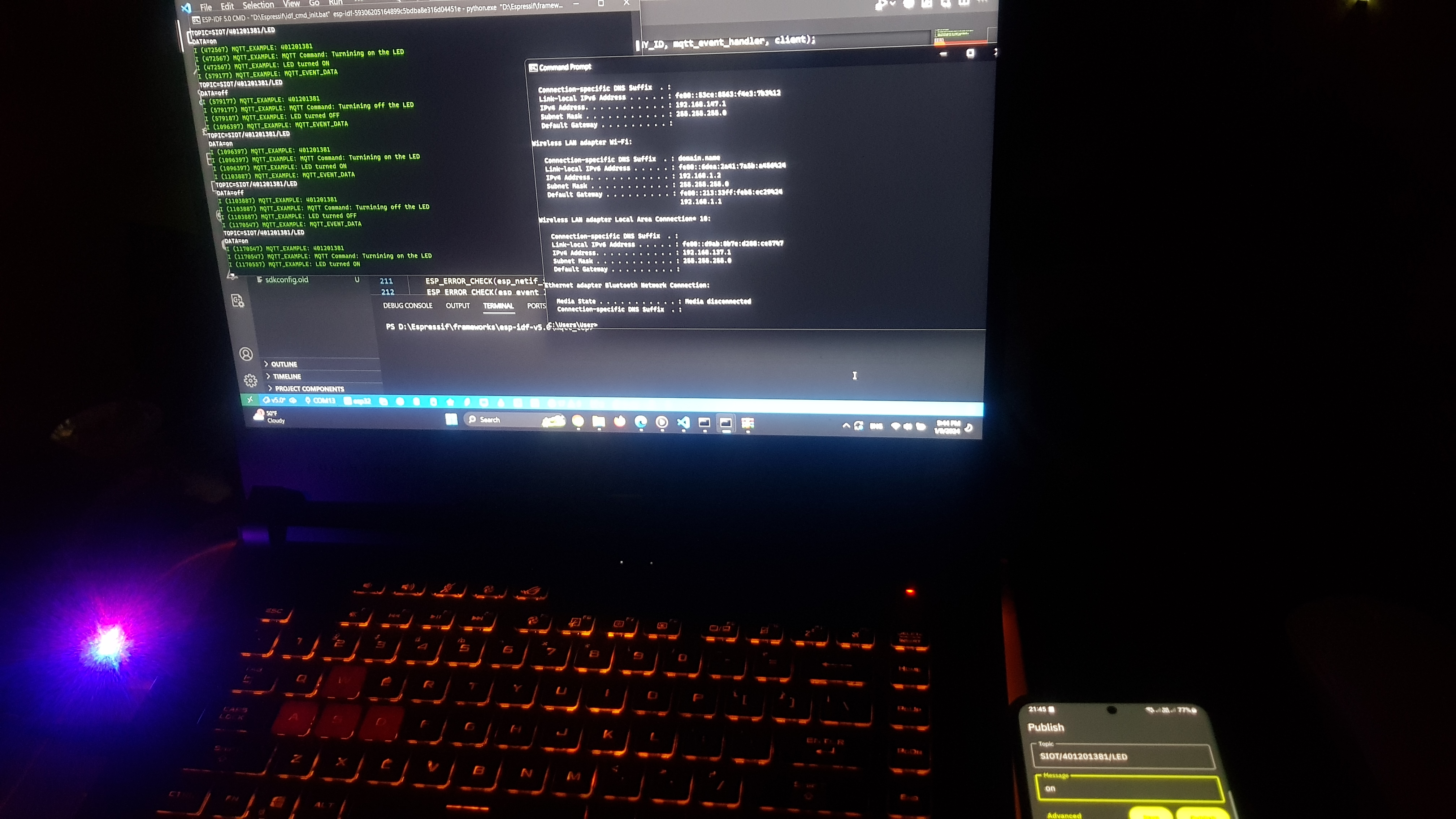
topic read SIOT/+/LED

کد esp نیز نیازی به تغییر ندارد.

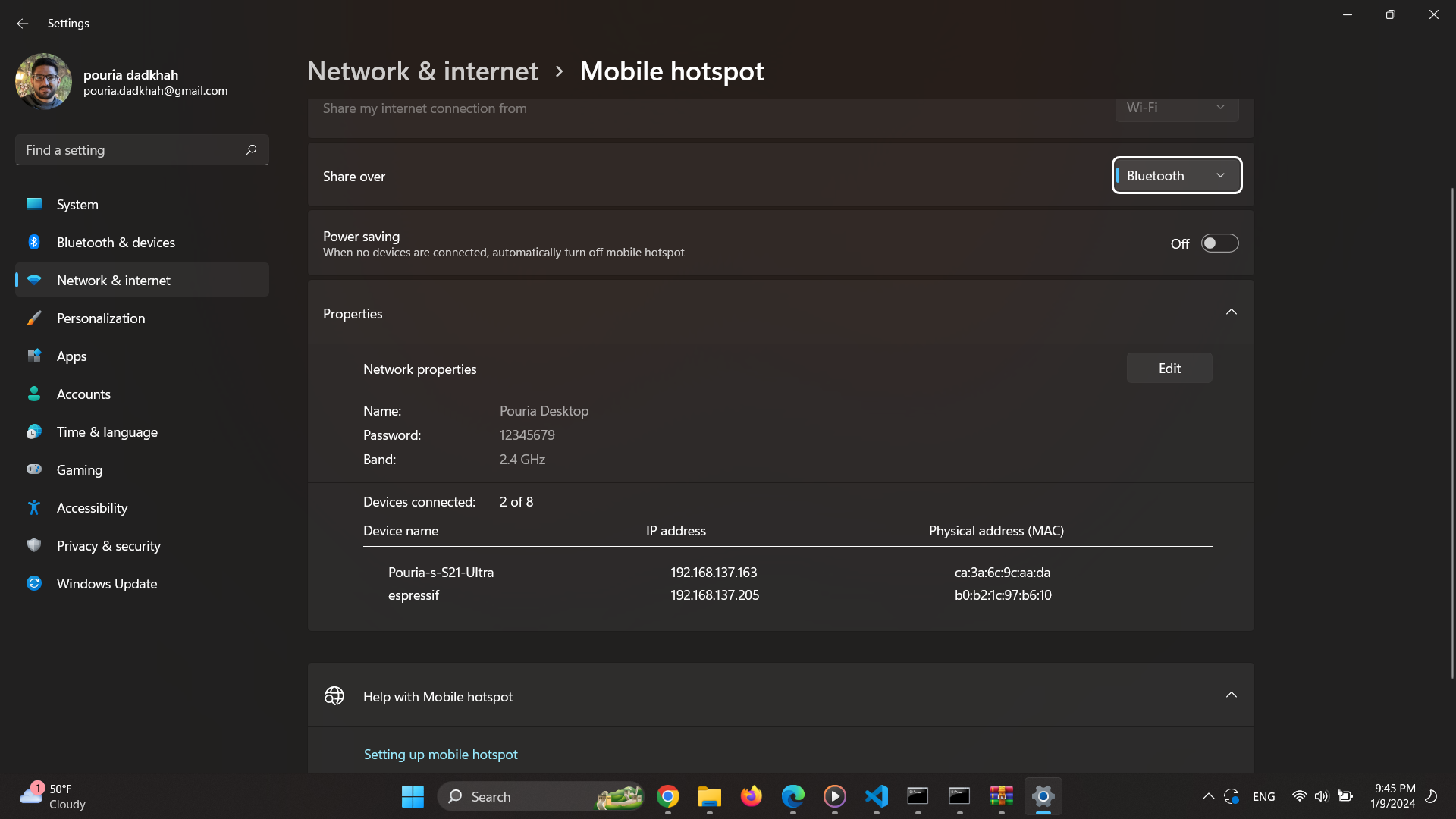
پس از اجرای کد و تست حالات مختلف می‌توان نتایج را مشاهده کرد:



در این عکس نتایج تلاش‌های ناموفق amin در ارسال پیام در 401201381 و همچنین عدم عملکرد پیام خود esp client در این تاپیک را می‌بینیم. هرچند ارسال پیام در تاپیکهای مجاز منجر به واکنش از طرف esp شده است.



نمونه‌ای از اتصال با گوشی و user Pouria و روشن کردن LED توسط آن



همان‌گونه که گفته‌شد از hotspot لپتاپ برای ایفای نقش روتر استفاده شد.

* فایل های کانفیگ هر قسمت و کدهای esp مربوط ضمیمه شده است.