باسمه تعالی

امنیت در اینترنت اشیاء

دکتر احمدی

تمرین پنجم

پیاده سازی CoAP

پوریا دادخواه

401201381

1. راه‌اندازی CoAP Server

برای ایجاد یک سرور روی esp32 با استفاده از esp-idf از نمونه پیاده‌سازی coap_server از مجموعه مثال‌های esp-idf به عنوان پایه پیاده‌سازی خود استفاده کرده و سپس برای هدف تمرین خود شخصی‌سازی می‌کنیم.

سرور coap توسط تابع coap_example_server() پیاده‌سازی شده و به هندل کردن اتصال کلاینت‌ها و درخواست‌های آن‌ها می‌پردازد.

در این تابع پس از تعریف متغیرهای لازم در طول برنامه و لاگ‌ها در یک حلقه true ارتباط سرور را آغاز می‌کنیم؛ به این‌صورت که پس از اتصال به شبکه محلی (مثل وای‌فای هات‌اسپات لپ‌تاپ که در این تمرین استفاده می‌کنیم) IP , Port سرور را طبق پیشفرض coap تنظیم می‌کنیم و یک endpoint جدید برای برقراری ارتباط از سمت کلاینت‌ها ایجاد می‌شود که آن را در مسیر "/espressif" می‌سازیم.

سپس توابع handler مربوط به هر متد (GET, PUT,DELETE,POST) را فراخوانی می‌کنیم.

هندلرهای GET,PUT,DELETE در نمونه کد پیاده شده بودند که نیازی به توضیح مفصل آن‌ها نیست و تنها به شرح هدف هرکدام اکتفا می‌کنیم:

- در این نمونه یک مقدار اولیه Initial data با مقدار "hello world" تعریف شده است که با درخواست GET از کلاینت‌ها این مقدار برای او در پاسخ ارسال می شود و صرفا جنبه تست و برقرار بودن صحیح ارتباط را دارد. با استفاده از متد PUT و ارسال مقدار جدید، این data تغییر یافته و از آن به بعد مقدار جدید در response بازگردانده می شود. با متد DELETE هم این مقدار جدید اختصاص یافته را حذف کرده و دوباره همان مقدار اولیه جایگزین data می‌شود.

در آخر به توضیح هندلر hnd_espressif_post() که خودمان به کد اضافه کردیم می‌پردازیم که باید دستور on , off را دریافت کرده و LED را روشن و خاموش کنیم.

در این تابع ابتدا مقدار data ارسالی از کاربر را دریافت می‌کنیم و بر اساس آن به اقدام واکنش مناسب می‌پردازیم. توضیحات on, off کردن led در تمرینات قبل آورده شده و کافیست پس از ست کردن led gpio و نوشتن و فراخوانیconfig آن در ابتدای اجرای برنامه در این قسمت، سطح led را به 0 و 1 تغییر دهیم.

در صورت دریافت مقداری غیر از on, off هم پیام ارور مناسب به کلاینت بازگردانده می‌شود.

در response هم پیام تغییر سطح روشنایی led را به کلاینت ارسال می‌کنیم:

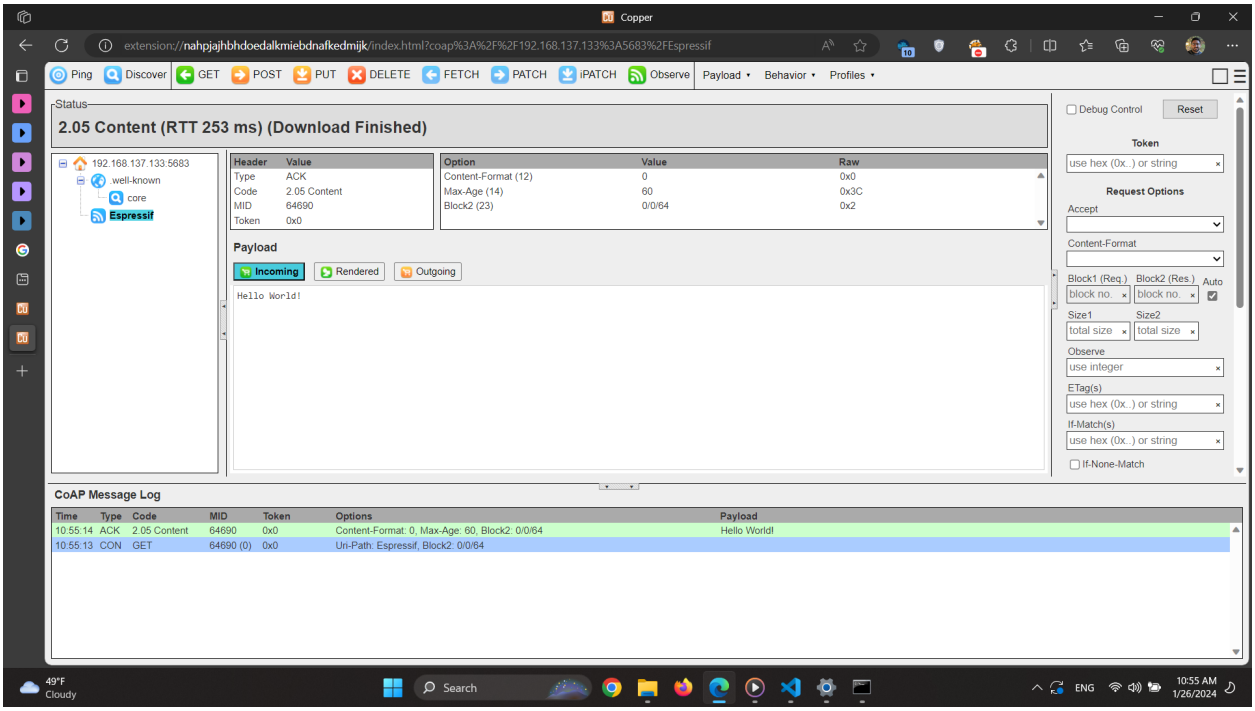
```c
static void
hnd_espressif_post(coap_resource_t *resource,
                   coap_session_t *session,
                   const coap_pdu_t *request,
                   const coap_string_t *query,
                   coap_pdu_t *response)
{
    size_t size;
    const unsigned char *data;

    // Get the payload data from the request
    (void)coap_get_data(request, &size, &data);

    if (size > 0) {
        // Check the payload for "on" or "off"
        if (strncmp((const char *)data, "on", size) == 0) {
            // Turn on the LED
            gpio_set_level(LED_PIN, 1);
            ESP_LOGI(TAG, "LED turned ON!");
        } else if (strncmp((const char *)data, "off", size) == 0) {
            // Turn off the LED
            gpio_set_level(LED_PIN, 0);
            ESP_LOGI(TAG, "LED turned OFF!");
        } else {
            // Invalid payload, respond with an error
            coap_pdu_set_code(response, COAP_RESPONSE_CODE_BAD_REQUEST);
            ESP_LOGE(TAG, "Invalid Payload!!");
            return;
        }
    } else {
        // Empty payload, respond with an error
        coap_pdu_set_code(response, COAP_RESPONSE_CODE_BAD_REQUEST);
        return;
    }

    // Notify observers about the change
    coap_resource_notify_observers(resource, NULL);

    // Set the response code to indicate success
    coap_pdu_set_code(response, COAP_RESPONSE_CODE_CHANGED);
}
```

پس از اجرای برنامه coap_server_example_main، مشابه تمرین قبل از طریق تنظیم SSID , pass menuconfig به وای‌فای سیستم خود متصل می‌شود و IP سرور ایجاد شده را نمایش می‌دهد و آماده‌است که کلاینت‌ها به آن متصل شوند. در پیاده‌سازی ما این آدرس برابر 192.168.137.133:5683 بوده که Port پیشفرض coap می‌باشد. در ادامه کلاینت را نیز آماده می‌کنیم و نتایج را نمایش می‌دهیم.

2. راه اندازی CoAP Client

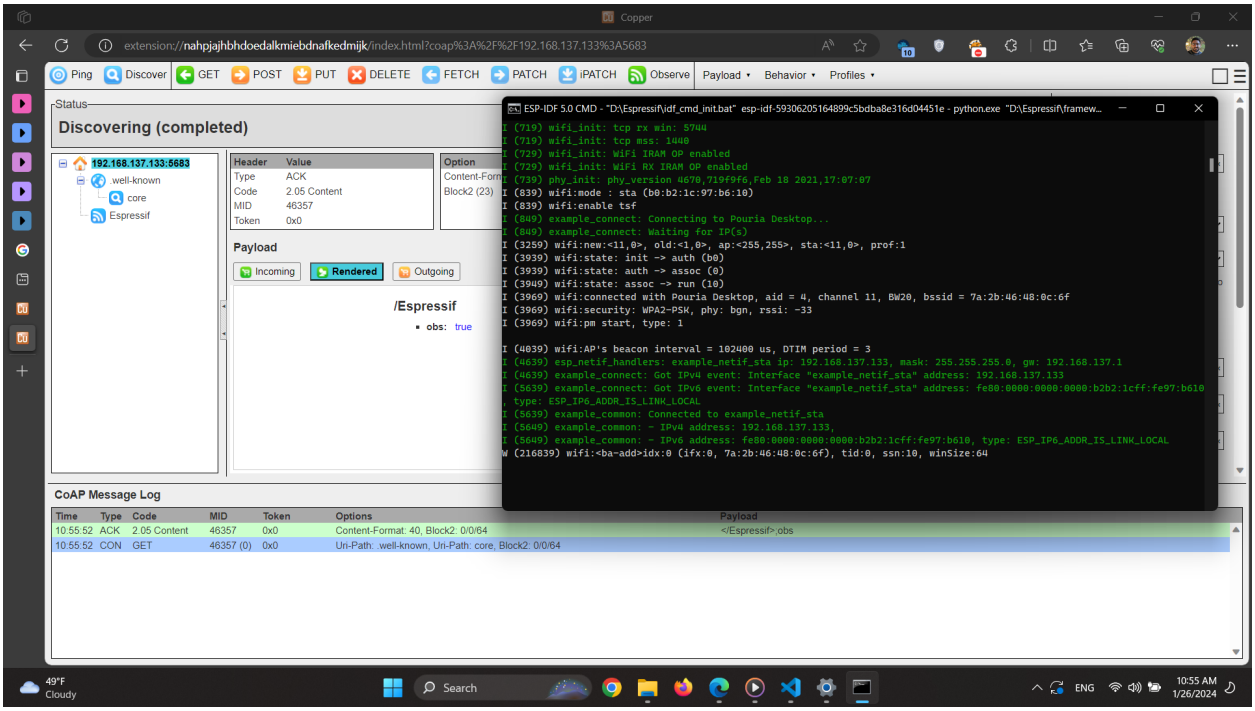
برای این قسمت از extention Copper روی مرورگر edge استفاده کردیم (چرا که نسخه chrome آن deprecate شده‌بود) و مراحل نصب و راه‌اندازی آن را مطابق دستورالعمل خود توسعه‌دهندگان اجرا کردیم.[1] پس از نصب copper کافی است آدرس سرور coap که در قسمت قبل به‌دست‌آوردیم را وارد کنیم تا با صفحه کلاینت مواجه شویم.

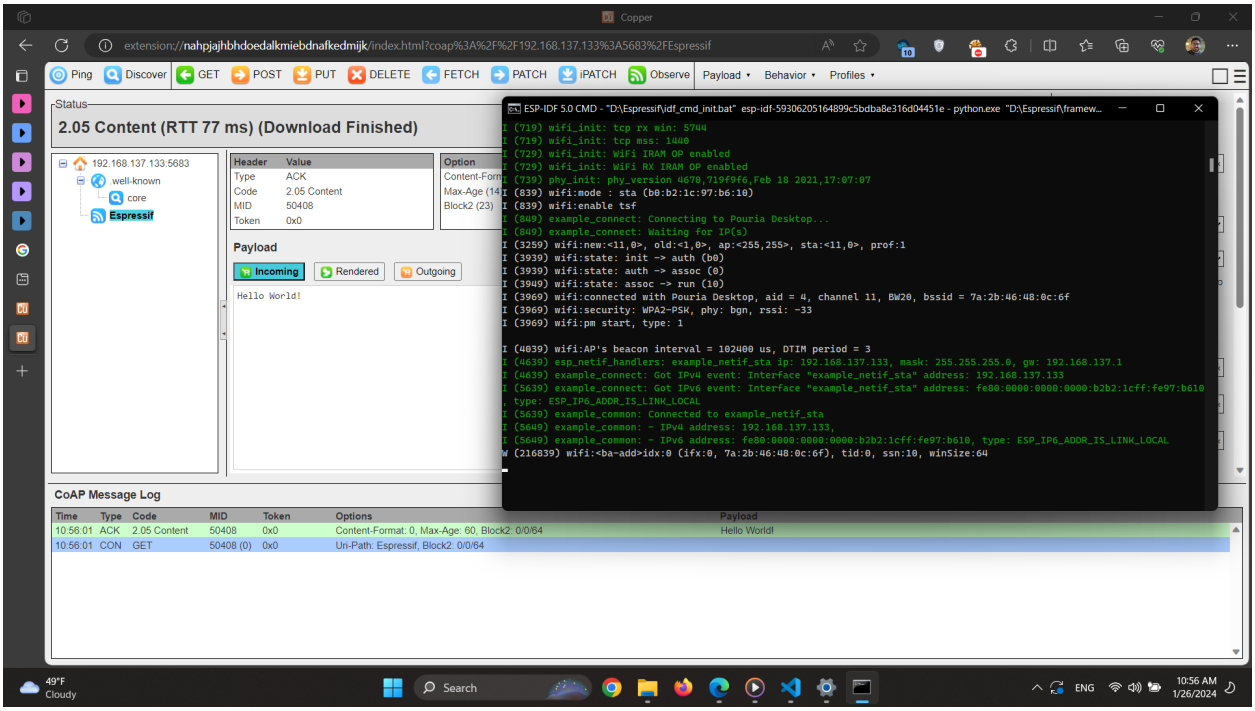
در این قسمت همان‌طور که انتظار داشتیم یک endpoint با نام espressif ایجاد شده که می‌توان در آن با متدهای مختلف به سرور درخواست ارسال کرد.



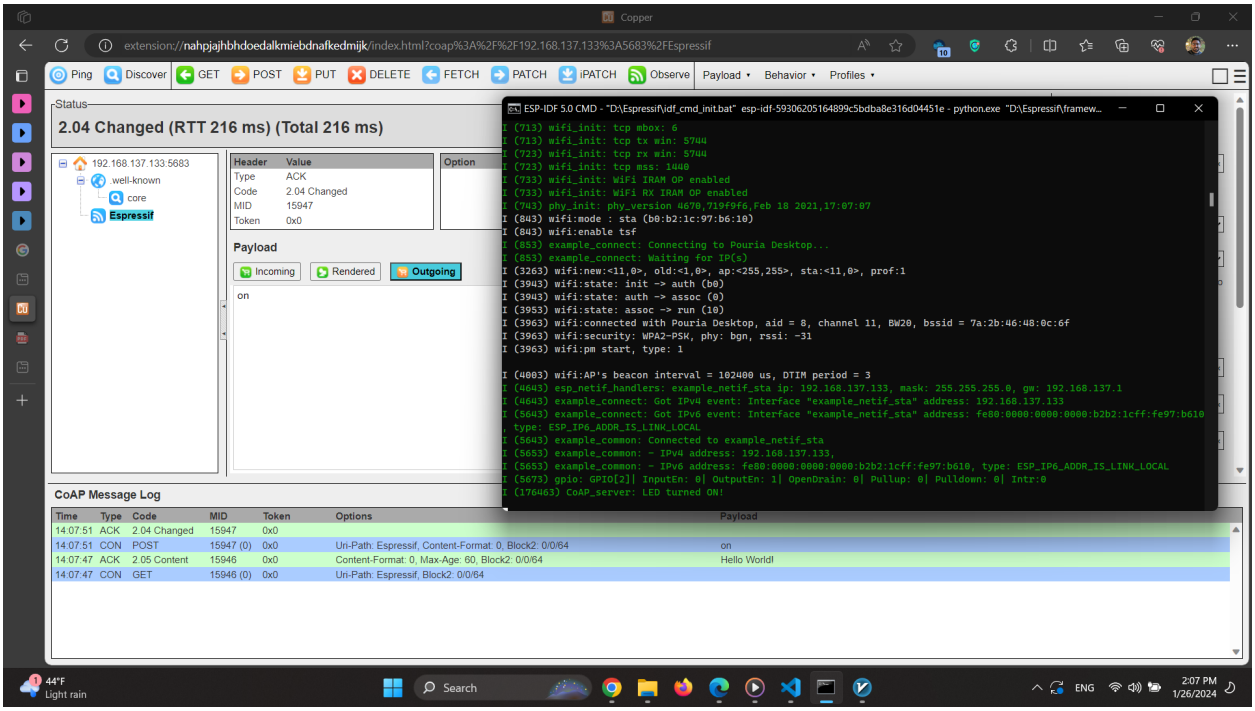
صفحه کلاینت copper و نمونه‌ای از درخواست GET ساده، که همان initial data تعریف شده در سرور را در پاسخ بازمی‌گرداند.

[1] https://github.com/mkovatsc/Copper4Cr

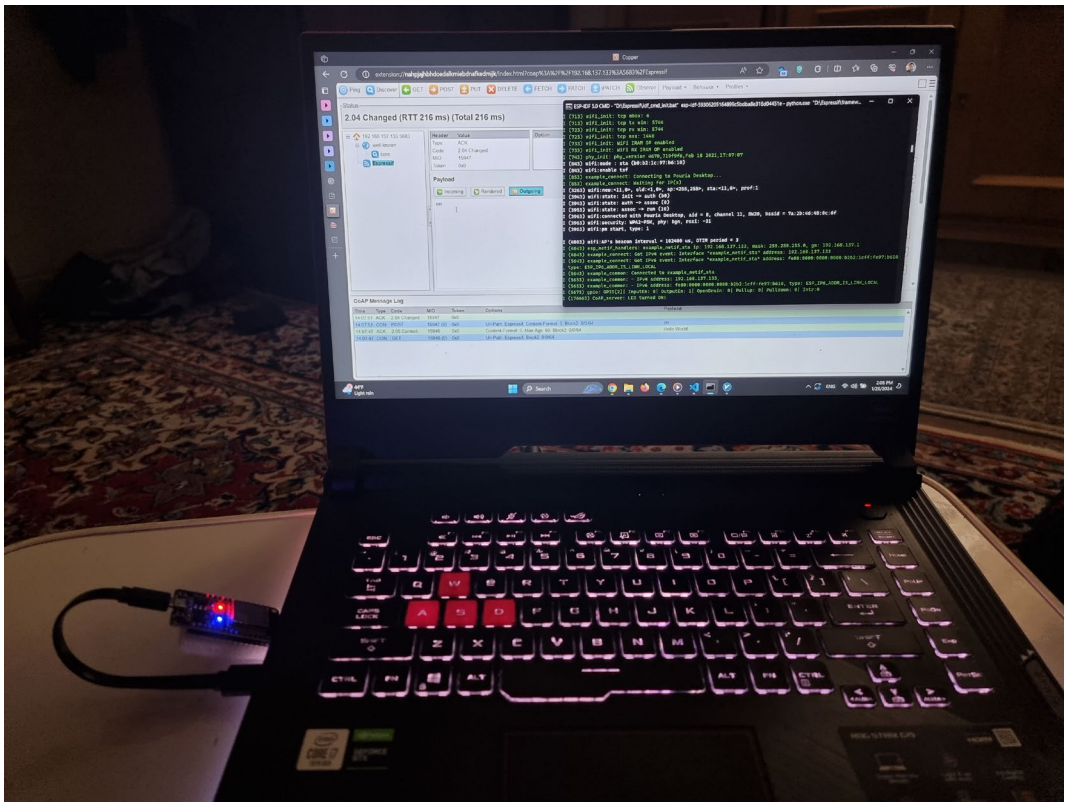




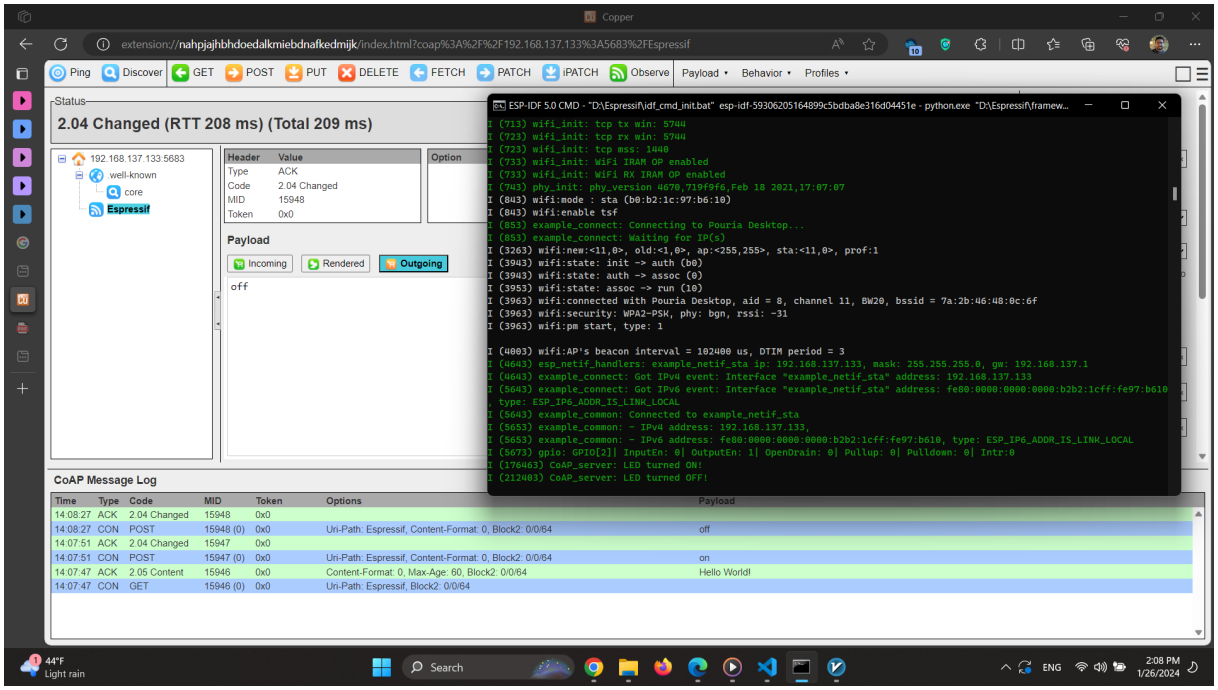
صفحه اصلی و صفحه endpoint ایجاد شده برای کلاینت پس از ارسال درخواست



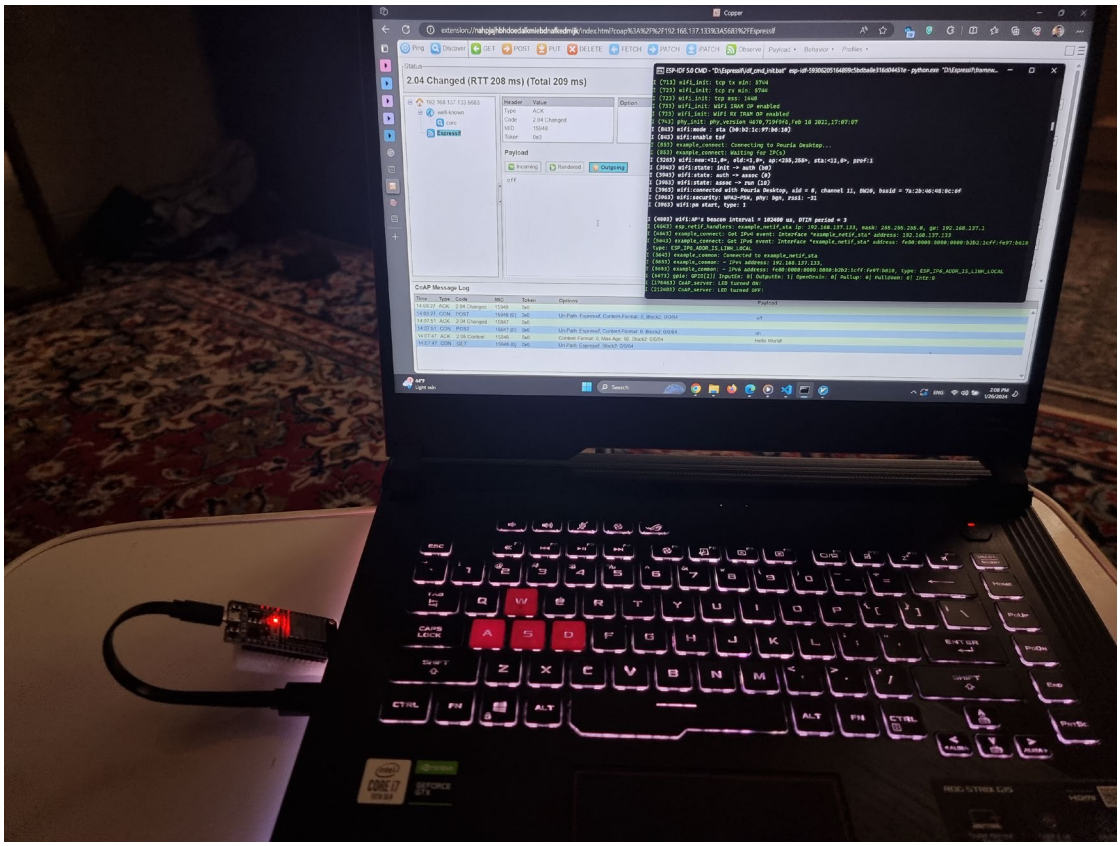
روشن کردن LED با ارسال دستور POST و payload = on



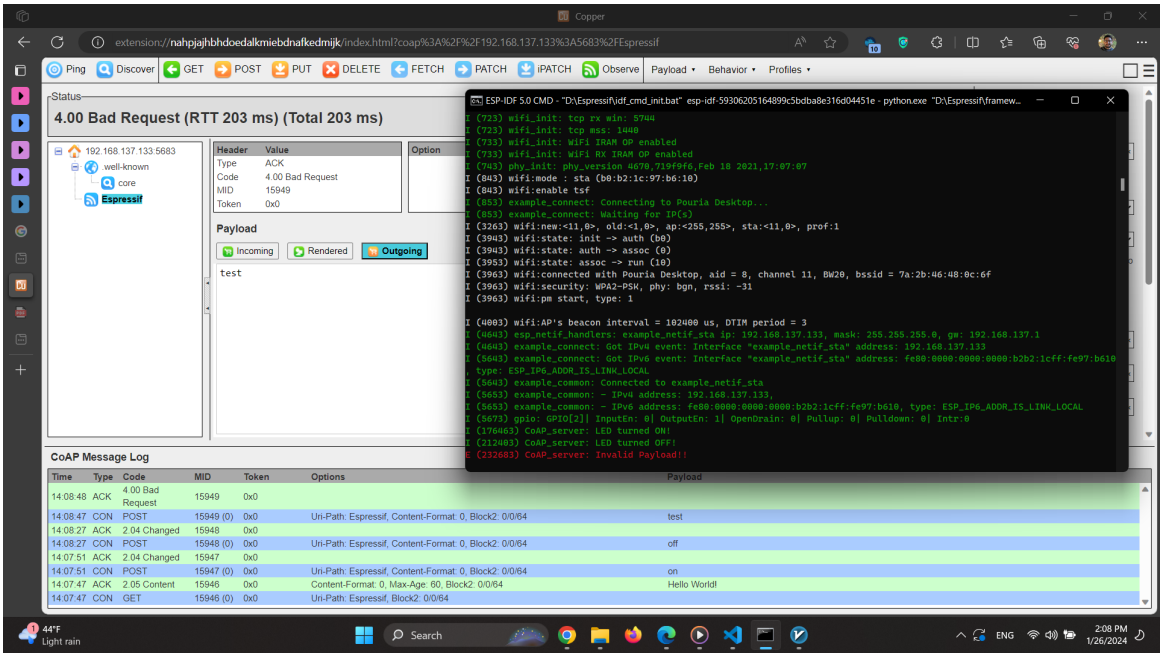
وضعیت LED



خاموش کردن LED با ارسال دستور POST و payload = off



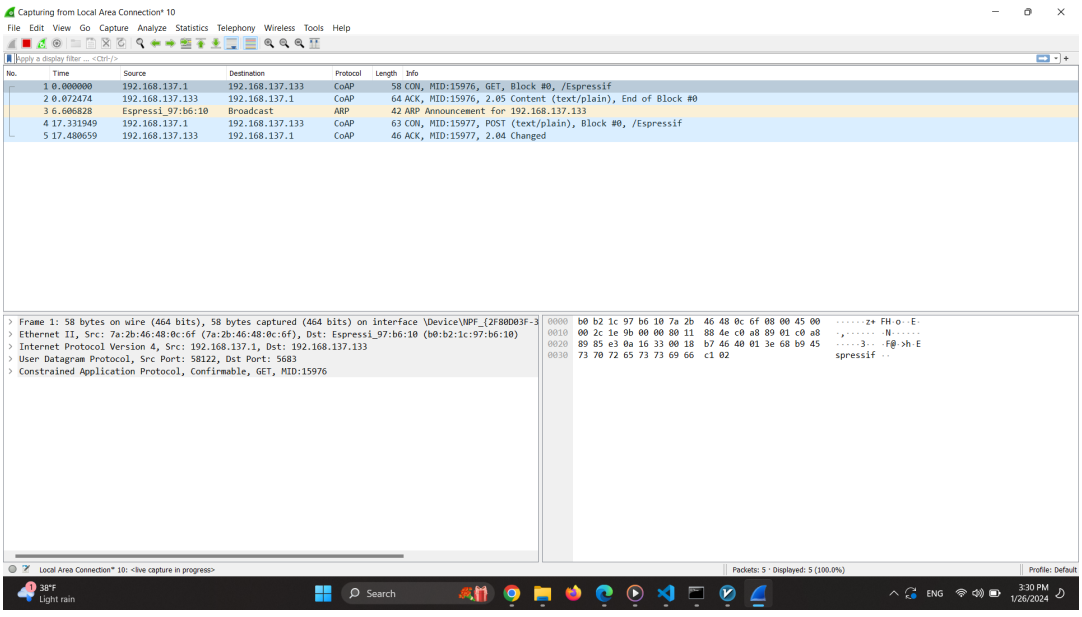
وضعیت LED



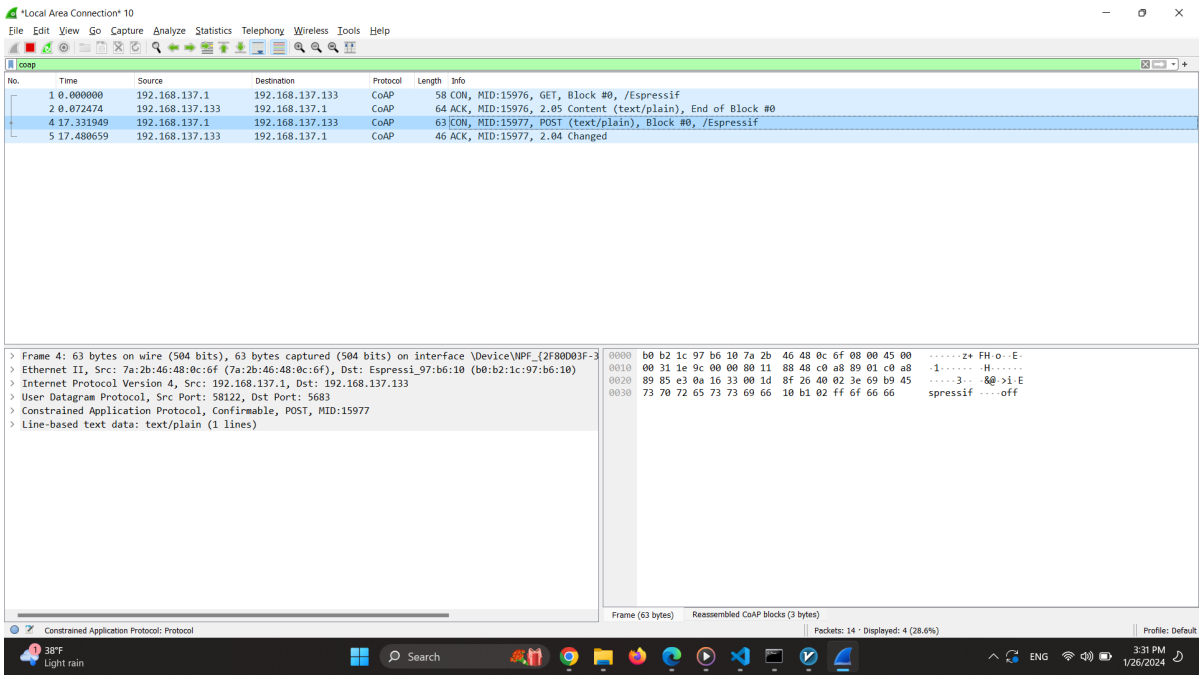
ارسال Payload غیرقابل قبول

3. Wireshark

در آخر با استفاده از wirshark و capture کردن ترافیک local area connection سیستم (چرا که به hotspot لپ‌تاپ متصل شده و می‌توان از شبکه داخلی ترافیک را مورد بررسی قرار داد) را می‌بینیم (شکل اول) و سپس تنها بسته‌های coap را با اعمال فیلتر coap روی ترافیک جدا می‌کنیم.

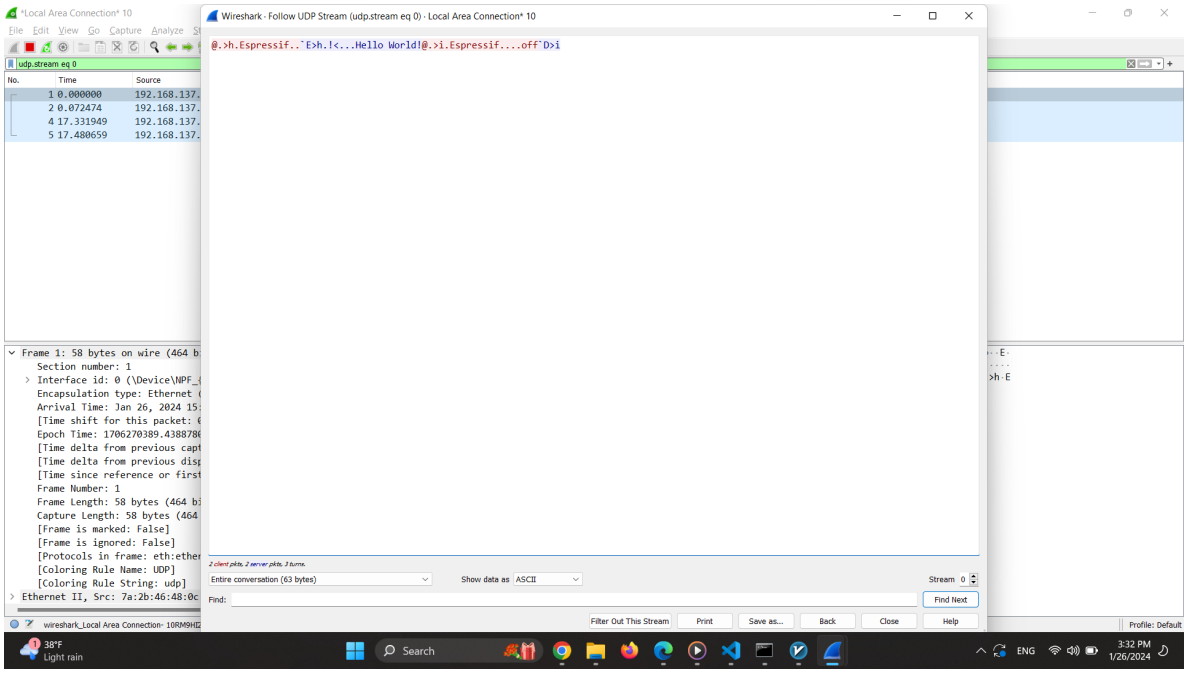


کل ترافیک عبوری از این شبکه hotspot

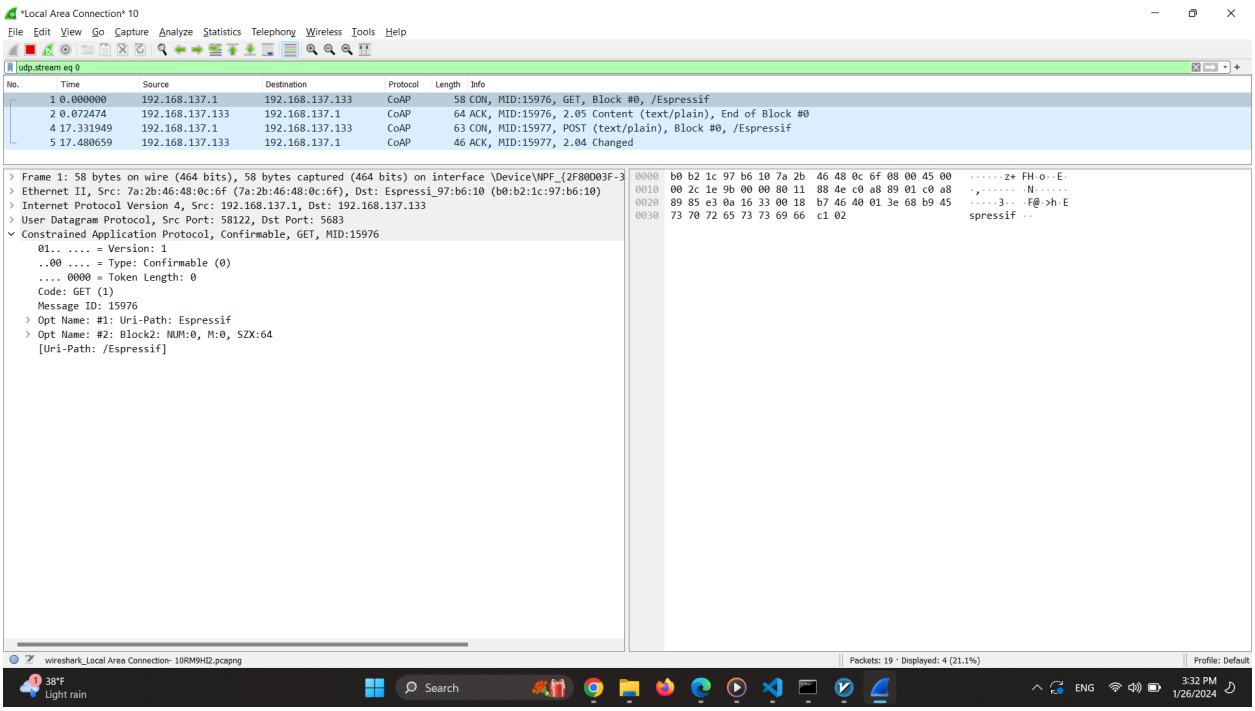


ترافیک coap

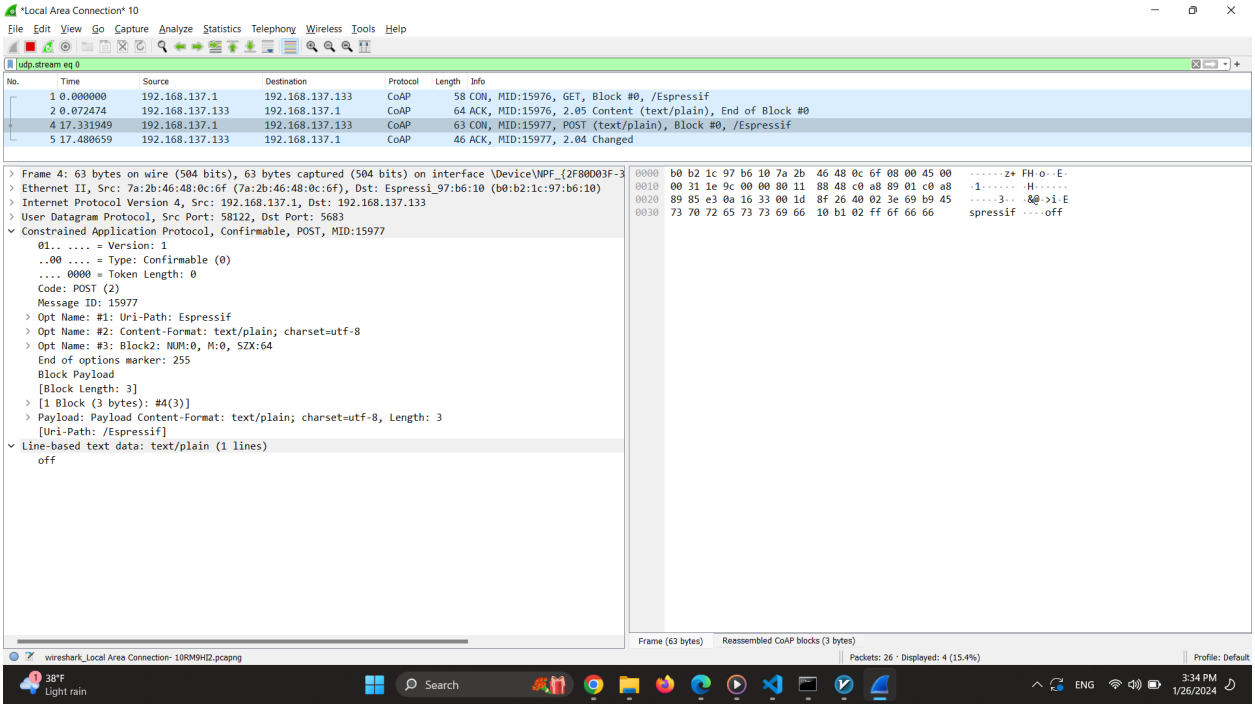
اکنون می‌توان هر بسته را با جزییات بیشتری آنالیز کرد و محتوای بسته، فریم کلی، payload (در صورت plain بودن) و follow کردن udp stream بسته‌ها را مشاهده کرد که در ادامه برخی از این موارد را برای مثال می‌آوریم.



دنبال (follow) کردن udp stream درخواست post با payload = off



جزییات یک بسته حاوی درخواست GET از کلاینت که شامل فریم، اطلاعات لایه ethernet 2، پروتکل و جزییات بستهcoap می‌باشد.



جزییات بسته درخواست POST که به دلیل plain بودن و عدم استفاده از DTLS می‌توان Payload درخواست را که برابر off است با sniff کردن بسته، مشاهده کرد.

- کد سرور coap_server و پوشه copper extention در فایل تمرین ضمیمه شده است.