

باسمه تعالی



امنیت در اینترنت اشياء

دکتر احمدی

تمرین پنجم

پیاده سازی CoAP

پوریا دادخواه

401201381

## 1. راه اندازی CoAP Server

برای ایجاد یک سرور روی esp32 با استفاده از esp-idf از نمونه پیاده سازی coap\_server از مجموعه مثال های esp-idf به عنوان پایه پیاده سازی خود استفاده کرده و سپس برای هدف تمرین خود شخصی سازی می کنیم.

سرور coap توسط تابع coap\_example\_server() پیاده سازی شده و به هندل کردن اتصالات کلاینت ها و درخواست های آن ها می پردازد.

در این تابع پس از تعریف متغیرهای لازم در طول برنامه و لاگ ها در یک حلقه true ارتباط سرور را آغاز می کنیم؛ به این صورت که پس از اتصال به شبکه محلی ( مثل وای فای هات اسپات لپ تاپ که در این تمرین استفاده می کنیم ) IP , Port سرور را طبق پیش فرض coap تنظیم می کنیم و یک endpoint جدید برای برقراری ارتباط از سمت کلاینت ها ایجاد می شود که آن را در مسیر "/espressif" می سازیم.

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

هندلرهای GET,PUT,DELETE در نمونه کد پیاده شده بودند که نیازی به توضیح مفصل آن ها نیست و تنها به شرح هدف هر کدام اکتفا می کنیم:

- در این نمونه یک مقدار اولیه Initial data با مقدار "hello world" تعریف شده است که با درخواست GET از کلاینت ها این مقدار برای او در پاسخ ارسال می شود و صرفاً 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 می شود.

در آخر به توضیح هندلر hnd\_espressif\_post() که خودمان به کد اضافه کردیم می پردازیم 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 می کنیم و بر اساس آن به اقدام واکنش مناسب می پردازیم. توضیحات on, off کردن led در تمرینات قبل آورده شده و کفایت پس از ست کردن led gpio و نوشتن و فراخوانی config 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 می شود.

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

```

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);
}

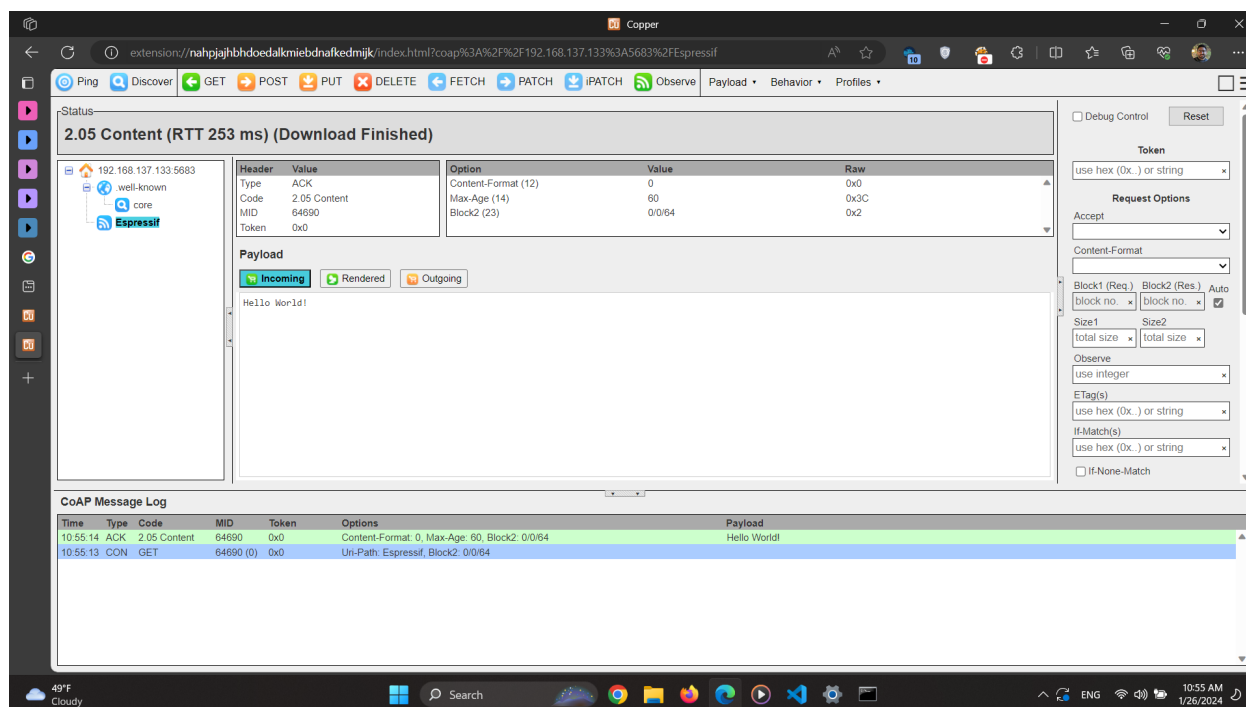
```

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.<sup>1</sup> 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 باز می‌گرداند.

<sup>1</sup> <https://github.com/mkovatsc/Copper4Cr>

**Status**  
Discovering (completed)

192.168.137.133:5683

well-known  
core  
Espressif

Header Value Option  
Type ACK  
Code 2.05 Content  
MID 46357  
Token 0x0  
Content-Format: 40, Block2 (23)

Payload  
Incoming Rendered Outgoing  
/Espressif  
obs: true

CoAP Message Log

Time	Type	Code	MID	Token	Options	Payload
10:55:52	ACK	2.05 Content	46357	0x0	Content-Format: 40, Block2: 0/0/64	<Espressif>obs
10:55:52	CON	GET	46357 (0)	0x0	Uri-Path: well-known, Uri-Path: core, Block2: 0/0/64	

```

ESP-IDF 5.0 CMD - "D:\Espressif\idf_cmd_init.bat" esp-idf-59306205164899c5bdba8e316d04431e - python.exe "D:\Espressif\framew...
(719) wifi_init: tcp rx win: 5700
(719) wifi_init: tcp mss: 1040
(729) wifi_init: WiFi IRAM OP enabled
(729) wifi_init: WiFi RX IRAM OP enabled
(729) phy_init: phy_version 4670, 719f9f6, Feb 18 2021, 17:07:07
(839) wifi_mode: sta (b0:b2:1c:97:b6:10)
(839) wifi_enable tsf
(849) example_connect: Connecting to Pouria Desktop...
(849) example_connect: Waiting for IP(s)
(3259) wifi:new=<11,0>, old=<1,0>, ap=<255,255>, sta=<11,0>, prof:1
(3939) wifi:state: init -> auth (b0)
(3939) wifi:state: auth -> assoc (0)
(3949) wifi:state: assoc -> run (10)
(3969) wifi:connected with Pouria Desktop, aid = 4, channel 11, BW20, bssid = 7a:2b:46:48:0c:6f
(3969) wifi:security: WPA2-PSK, phy: bgn, rssi: -33
(3969) wifi:pm start, type: 1

(4039) wifi:AP's beacon interval = 182400 us, DTIM period = 3
(4039) esp_netif_handlers: example_netif_sta ip: 192.168.137.133, mask: 255.255.255.0, gw: 192.168.137.1
(4039) example_connect: Got IPv4 event: Interface "example_netif_sta" address: 192.168.137.133
(5639) example_connect: Got IPv6 event: Interface "example_netif_sta" address: fe80:0000:0000:b2b2:1c97:fe97:b610
type: ESP_IP6_ADDR_IS_LINK_LOCAL
(5639) example_common: Connected to example_netif_sta
(5649) example_common: - IPv4 address: 192.168.137.133,
(5649) example_common: - IPv6 address: fe80:0000:0000:b2b2:1c97:fe97:b610, type: ESP_IP6_ADDR_IS_LINK_LOCAL
(216839) wifi->ba-addidx:0 (ifx:0, 7a:2b:46:48:0c:6f), tid:0, ssn:10, winSize:64
  
```

**Status**  
2.05 Content (RTT 77 ms) (Download Finished)

192.168.137.133:5683

well-known  
core  
Espressif

Header Value Option  
Type ACK  
Code 2.05 Content  
MID 50408  
Token 0x0  
Content-Format: 0, Max-Age: 60, Block2 (23)

Payload  
Incoming Rendered Outgoing  
Hello World!

CoAP Message Log

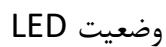
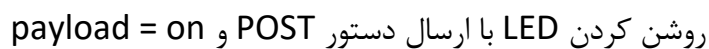
Time	Type	Code	MID	Token	Options	Payload
10:56:01	ACK	2.05 Content	50408	0x0	Content-Format: 0, Max-Age: 60, Block2: 0/0/64	Hello World!
10:56:01	CON	GET	50408 (0)	0x0	Uri-Path: Espressif, Block2: 0/0/64	

```

ESP-IDF 5.0 CMD - "D:\Espressif\idf_cmd_init.bat" esp-idf-59306205164899c5bdba8e316d04431e - python.exe "D:\Espressif\framew...
(719) wifi_init: tcp rx win: 5700
(719) wifi_init: tcp mss: 1040
(729) wifi_init: WiFi IRAM OP enabled
(729) wifi_init: WiFi RX IRAM OP enabled
(729) phy_init: phy_version 4670, 719f9f6, Feb 18 2021, 17:07:07
(839) wifi_mode: sta (b0:b2:1c:97:b6:10)
(839) wifi_enable tsf
(849) example_connect: Connecting to Pouria Desktop...
(849) example_connect: Waiting for IP(s)
(3259) wifi:new=<11,0>, old=<1,0>, ap=<255,255>, sta=<11,0>, prof:1
(3939) wifi:state: init -> auth (b0)
(3939) wifi:state: auth -> assoc (0)
(3949) wifi:state: assoc -> run (10)
(3969) wifi:connected with Pouria Desktop, aid = 4, channel 11, BW20, bssid = 7a:2b:46:48:0c:6f
(3969) wifi:security: WPA2-PSK, phy: bgn, rssi: -33
(3969) wifi:pm start, type: 1

(4039) wifi:AP's beacon interval = 182400 us, DTIM period = 3
(4039) esp_netif_handlers: example_netif_sta ip: 192.168.137.133, mask: 255.255.255.0, gw: 192.168.137.1
(4039) example_connect: Got IPv4 event: Interface "example_netif_sta" address: 192.168.137.133
(5639) example_connect: Got IPv6 event: Interface "example_netif_sta" address: fe80:0000:0000:b2b2:1c97:fe97:b610
type: ESP_IP6_ADDR_IS_LINK_LOCAL
(5639) example_common: Connected to example_netif_sta
(5649) example_common: - IPv4 address: 192.168.137.133,
(5649) example_common: - IPv6 address: fe80:0000:0000:b2b2:1c97:fe97:b610, type: ESP_IP6_ADDR_IS_LINK_LOCAL
(216839) wifi->ba-addidx:0 (ifx:0, 7a:2b:46:48:0c:6f), tid:0, ssn:10, winSize:64
  
```

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





The screenshot displays a web browser window with a REST client extension. The main interface shows a '2.04 Changed (RTT 208 ms) (Total 209 ms)' status. The 'Header' section lists 'Type: ACK', 'Code: 2.04 Changed', 'MID: 15948', and 'Token: 0x0'. The 'Payload' section shows 'off'. The 'CoAP Message Log' table below lists several messages, including a '2.04 Content' message with a 'Hello World' payload.

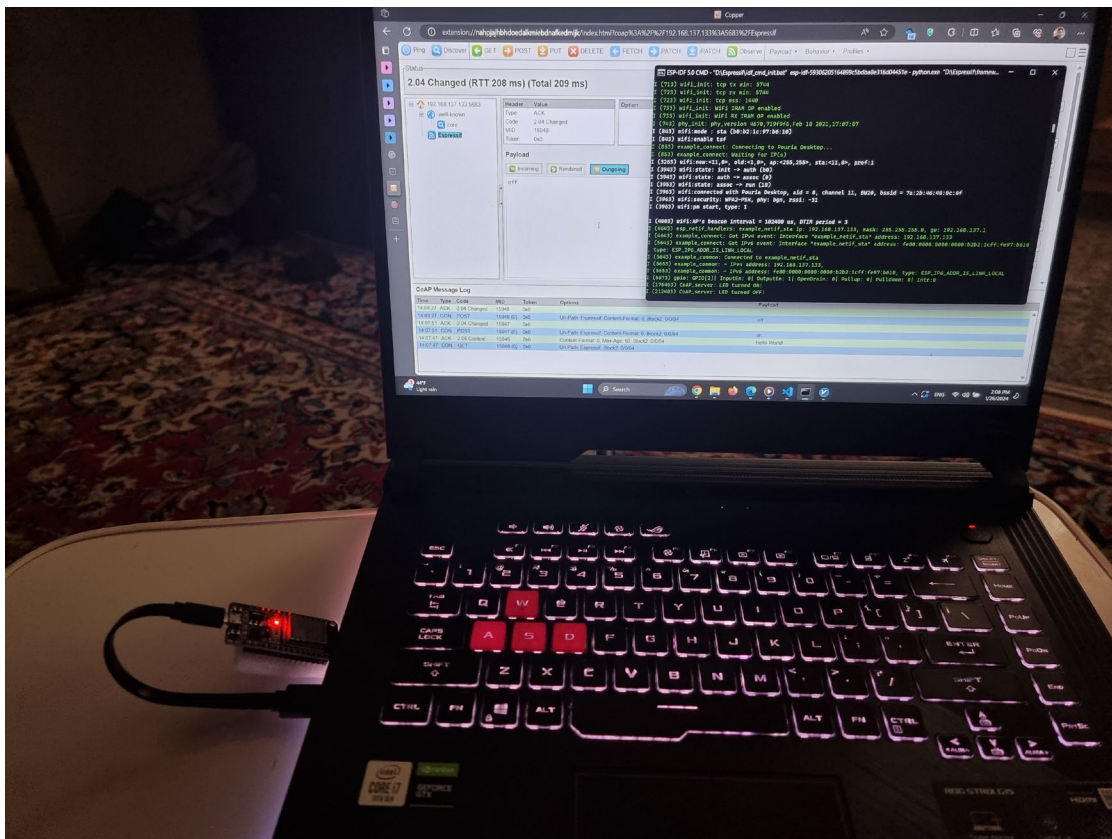
Overlaid on the right is a terminal window showing ESP-IDF 5.0 logs. The logs indicate the module is connecting to a network, receiving an IP address, and turning an LED off. Key log entries include:

```

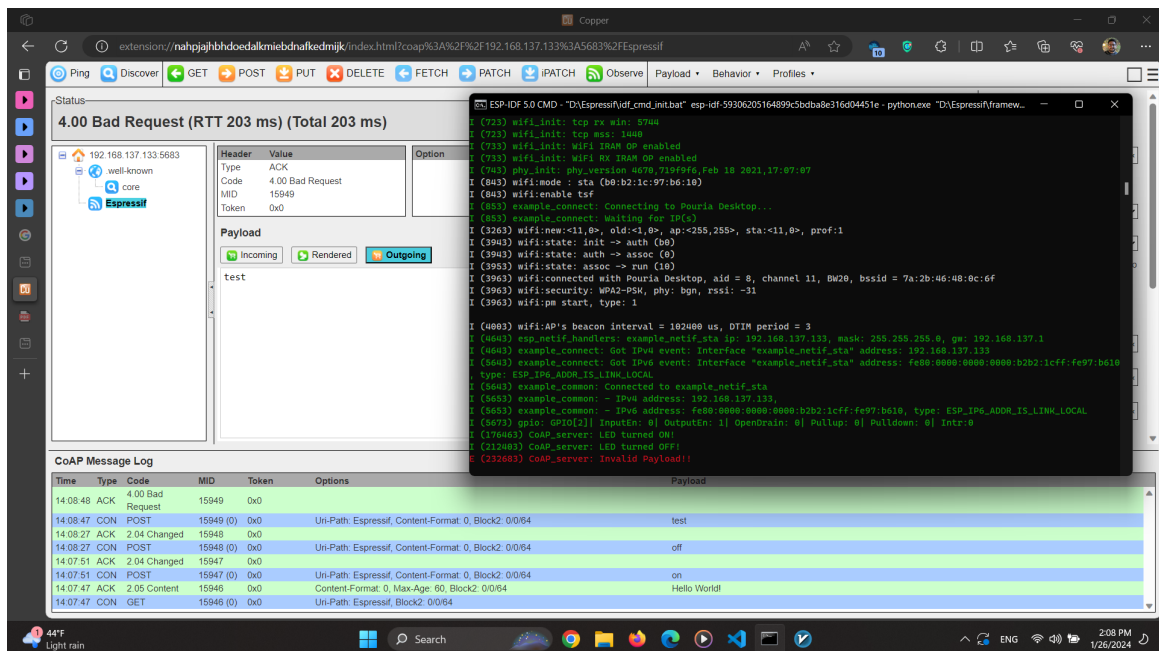
(713) wifi_init: tcp rx win: 5744
(723) wifi_init: tcp rx win: 5744
(723) wifi_init: tcp rx win: 5744
(723) wifi_init: WiFi IRAM OP enabled
(733) wifi_init: WiFi IRAM OP enabled
(743) phy_init: phy_version 4070,719f9f6, Feb 18 2021,17:07:07
(843) wifi_mode: sta (b6:b2:1c:97:b6:10)
(843) wifi_enable: wifi
(853) example_connect: Connecting to Pouria Desktop...
(853) example_connect: Waiting for IP(s)
(3263) wifi:new=<1,0>, old=<1,0>, ap:<255,255>, sta:<1,0>, prof:1
(3943) wifi:state: init -> auth (0)
(3943) wifi:state: auth -> assoc (0)
(3953) wifi:state: assoc -> run (10)
(3963) wifi:connected with Pouria Desktop, aid = 8, channel 11, BW20, bssid = 7a:2b:46:48:0c:6f
(3963) wifi:security: WPA2-PSK, phy: bgn, rssi: -31
(3963) wifi:pm start, type: 1

(4083) wifi:AP's beacon interval = 102400 us, DTIM period = 3
(4083) esp_netif_handlers: example_netif_sta ip: 192.168.137.133, mask: 255.255.255.0, gw: 192.168.137.1
(4083) example_connect: Got IPv4 event: Interface "example_netif_sta" address: 192.168.137.133
(5643) gpio: GPIO[21] Input: 0 Output: 1 OpenDrain: 0 Pullup: 0 Pulldown: 0 Intr: 0
(5643) example_common: Connected to example_netif_sta
(5643) example_common: ~ IPv4 address: 192.168.137.133,
(5643) example_common: ~ IPv6 address: fe80:0000:0000:0000:b2b2:1c9f:fe97:b610, type: ESP_IPv4_ADDR_IS_LINK_LOCAL
(5673) gpio: GPIO[21] Input: 0 Output: 1 OpenDrain: 0 Pullup: 0 Pulldown: 0 Intr: 0
(17643) CoAP_server: LED turned ON!
(21243) CoAP_server: LED turned OFF!
  
```

خاموش کردن LED با ارسال دستور POST و payload = off



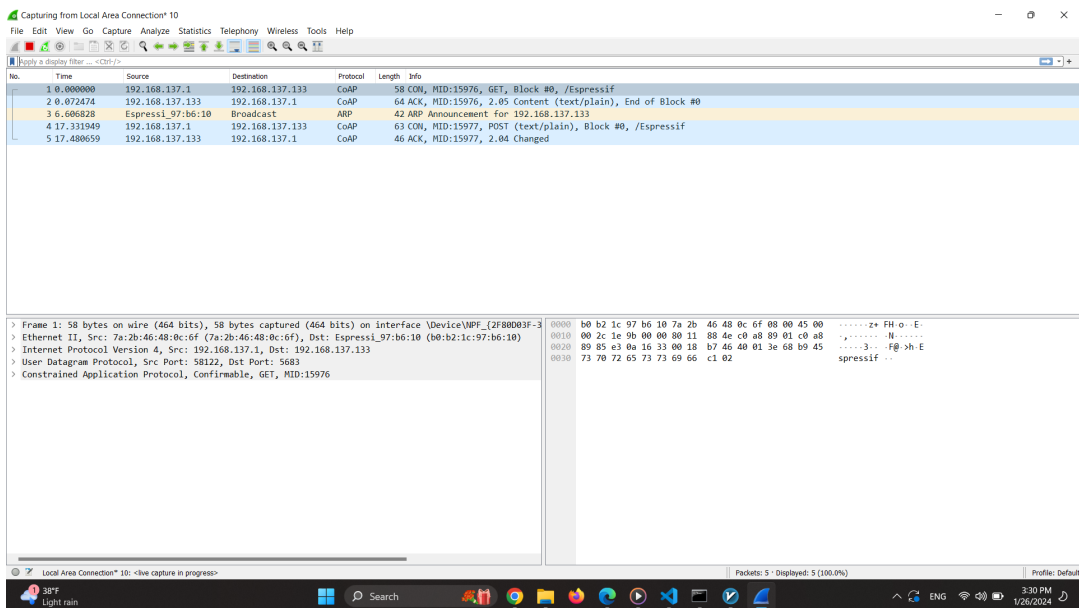
وضعیت LED



ارسال Payload غیر قابل قبول

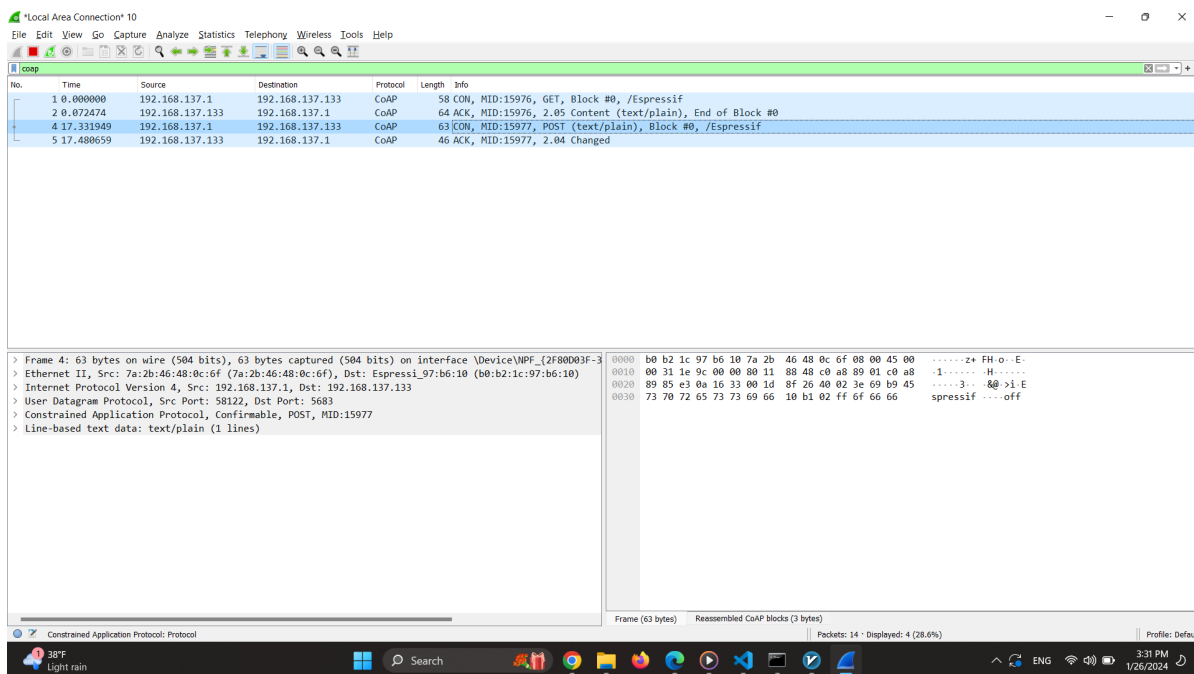
### 3. Wireshark

در آخر با استفاده از wirshark و capture کردن ترافیک local area connection سیستم ( چرا که به hotspot لپ تاپ متصل شده و می توان از شبکه داخلی ترافیک را مورد بررسی قرار داد ) را می بینیم ( شکل اول ) و سپس تنها بسته های coap را با اعمال فیلتر coap روی ترافیک جدا می کنیم.



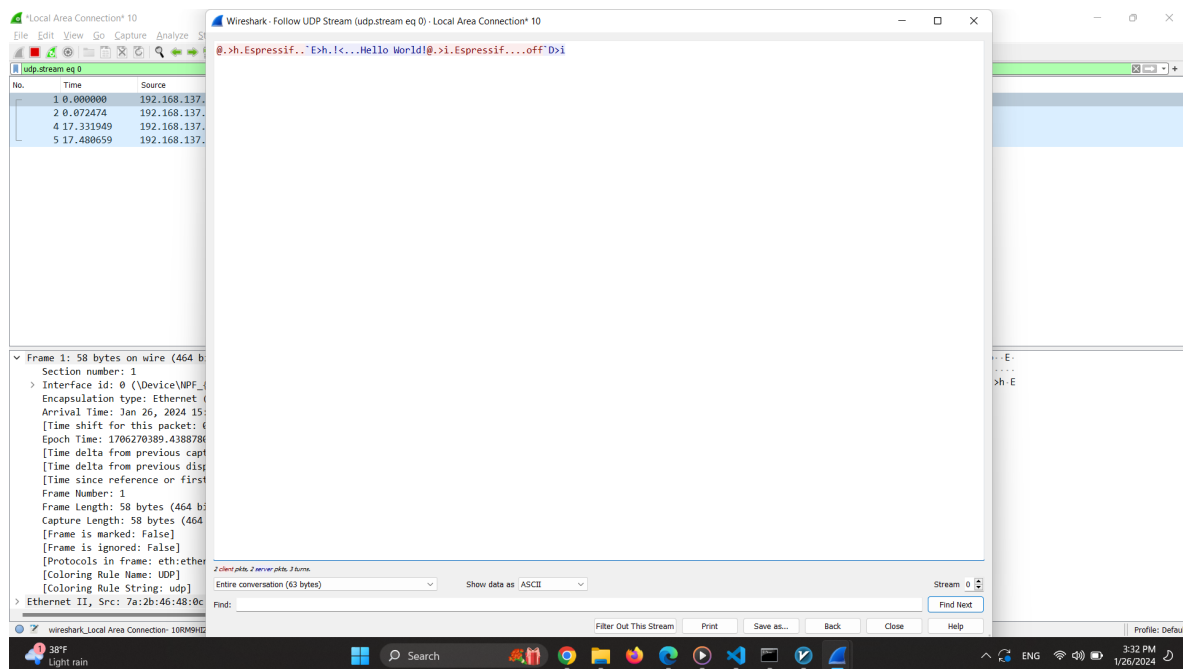
کل ترافیک عبوری از این شبکه hotspot



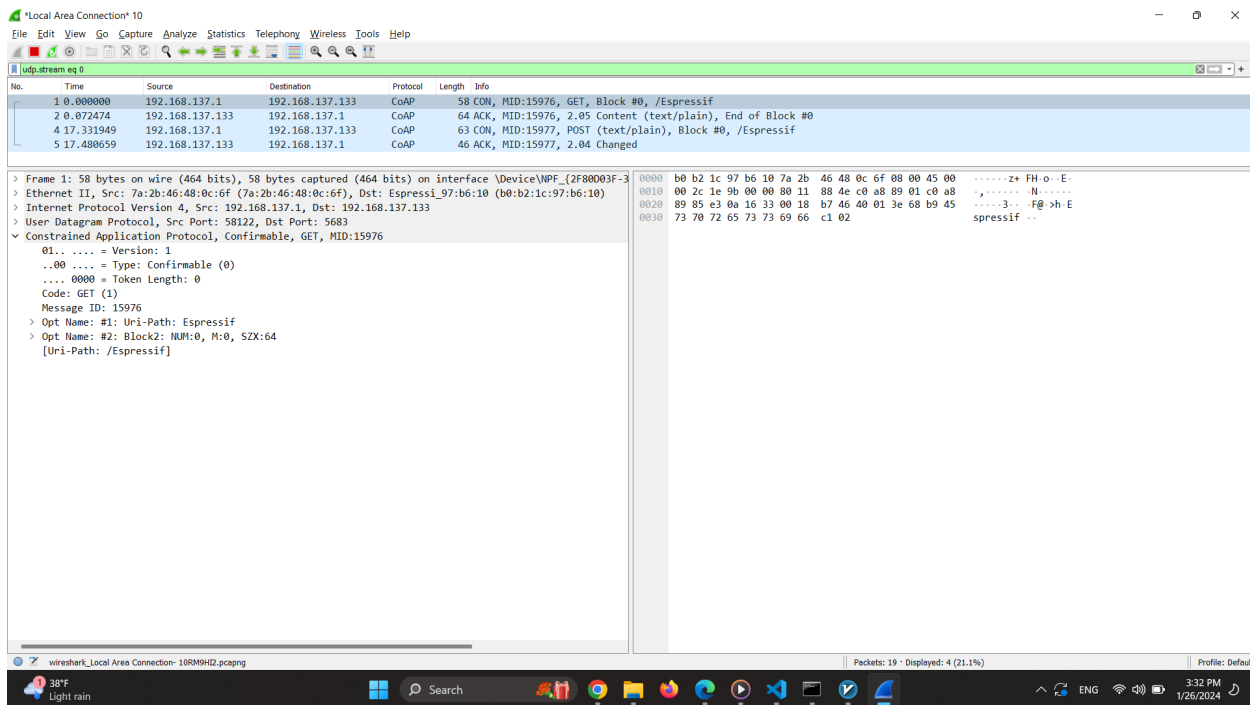


## ترافیک coap

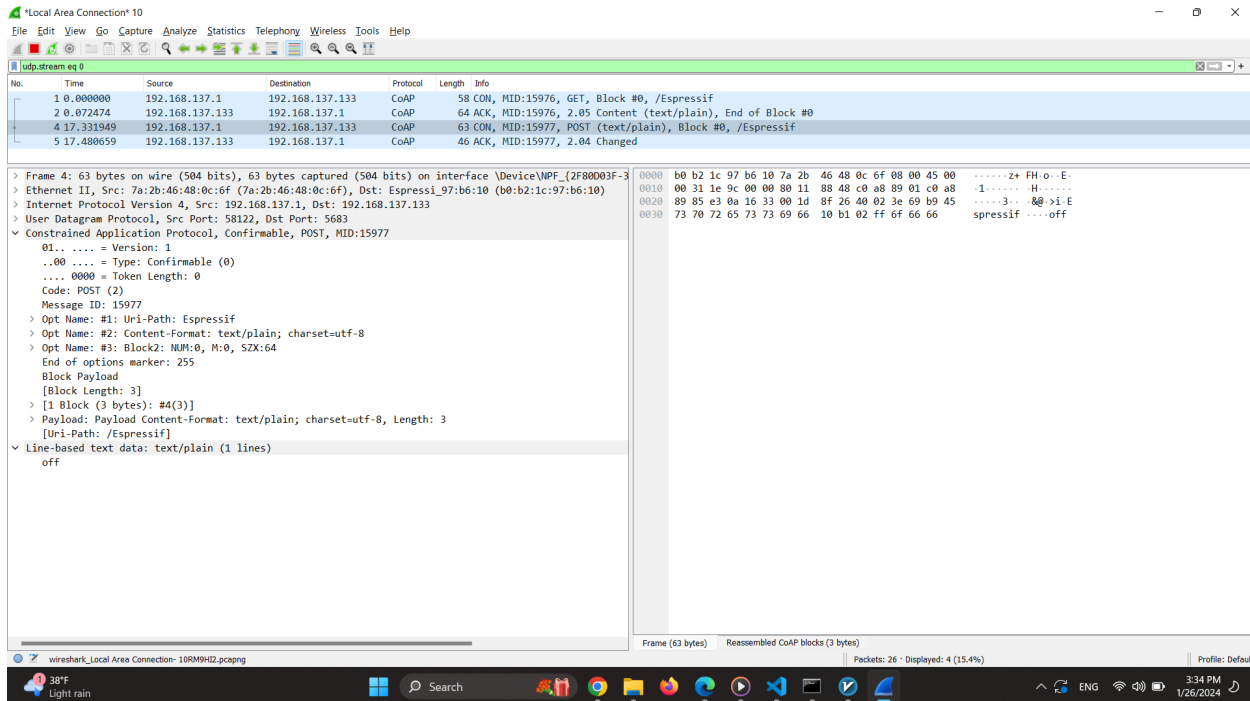
اکنون می‌توان هر بسته را با جزئیات بیشتری آنالیز کرد و محتوای بسته، فریم کلی، payload (در صورت plain بودن) و follow کردن udp stream بسته‌ها را مشاهده کرد که در ادامه برخی از این موارد را برای مثال می‌آوریم.



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



جزییات یک بسته حاوی درخواست GET از کلاینت که شامل فریم، اطلاعات لایه 2 ethernet، پروتکل و جزئیات بسته coap می باشد.



جزییات بسته درخواست POST که به دلیل plain بودن و عدم استفاده از DTLS می توان 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.