SIoT →HW 5 →CoAP
Parisa Toumari
99101857

This report discusses the development of a CoAP (Constrained Application Protocol) server on an ESP module. The server is designed to operate over a local Wi-Fi network and respond to client requests. Similar to a previous exercise that used HTTP, this CoAP server allows clients to control an LED on the ESP module based on received commands. The server's implementation involves handling CoAP requests, processing client commands, and updating the state of an onboard LED accordingly.

- CoAP Context Initialization:

```
ctx = coap_new_context(NULL);
```

Creates a new CoAP context. The context is a central data structure that holds information about the CoAP server and its configuration.

- Block-wise Transfer Configuration:

```
coap_context_set_block_mode(ctx, COAP_BLOCK_USE_LIBCOAP |
COAP_BLOCK_SINGLE_BODY);
```

Configures block-wise transfer mode, allowing the server to handle large payloads efficiently.

- Endpoint Configuration:

```
esp_netif_init(); // TCP/IP initiation
esp_event_loop_create_default(); // event loop
esp_netif_create_default_wifi_sta(); // WiFi station
```

Initializes the ESP network interface, event loop, and sets up Wi-Fi for station mode.

- Event Handlers:

```
esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID,
wifi_event_handler, NULL);
esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP,
wifi_event_handler, NULL);
```

Registers event handlers for Wi-Fi events, such as start, connection, disconnection, and obtaining an IP address.

- Wi-Fi Configuration:

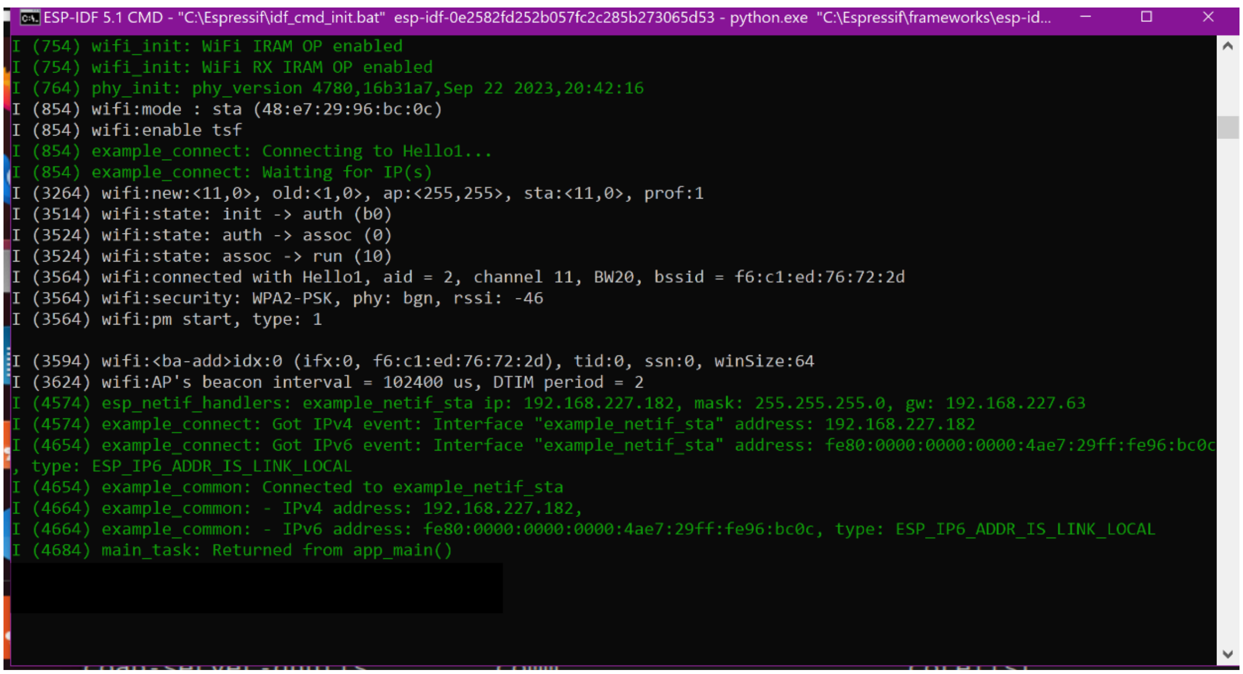
```
wifi_config_t wifi_configuration = {
    .sta = {
        .ssid = "Parisa's iPhone",
        .password = "ParisPass"}};
esp_wifi_set_config(ESP_IF_WIFI_STA, &wifi_configuration);
```

Configures Wi-Fi with the SSID and password.

And other configurations …

1.

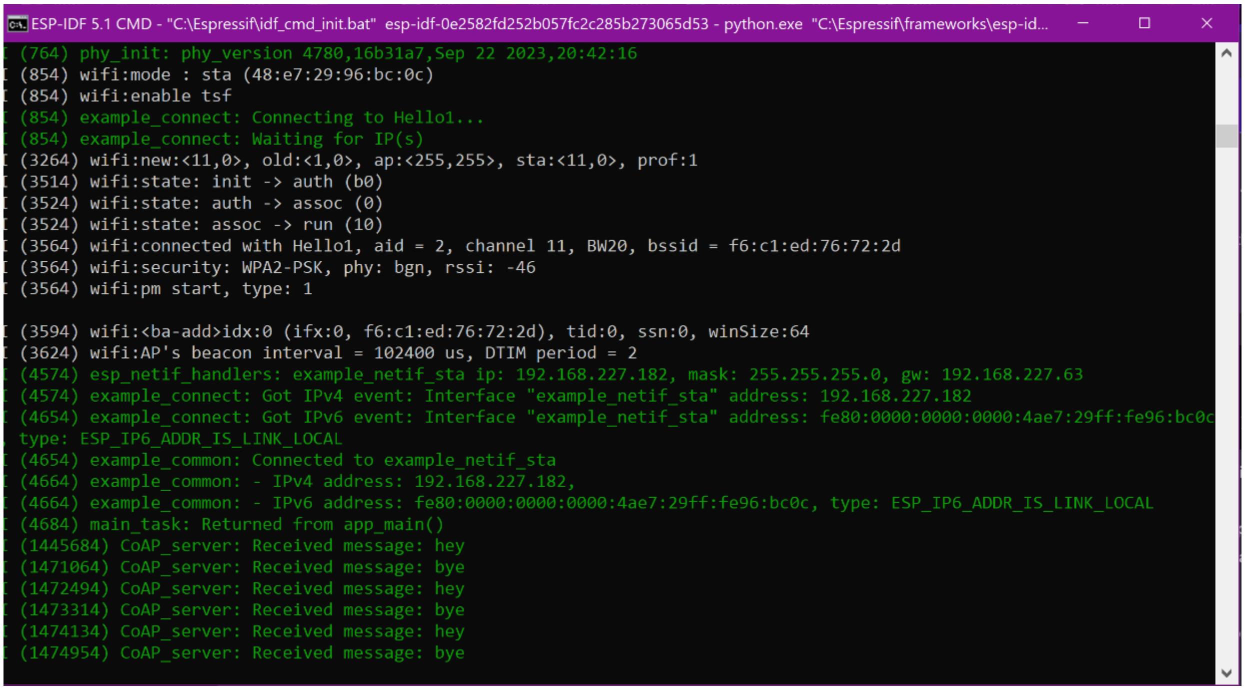
Idf.py build flash monitor ➔



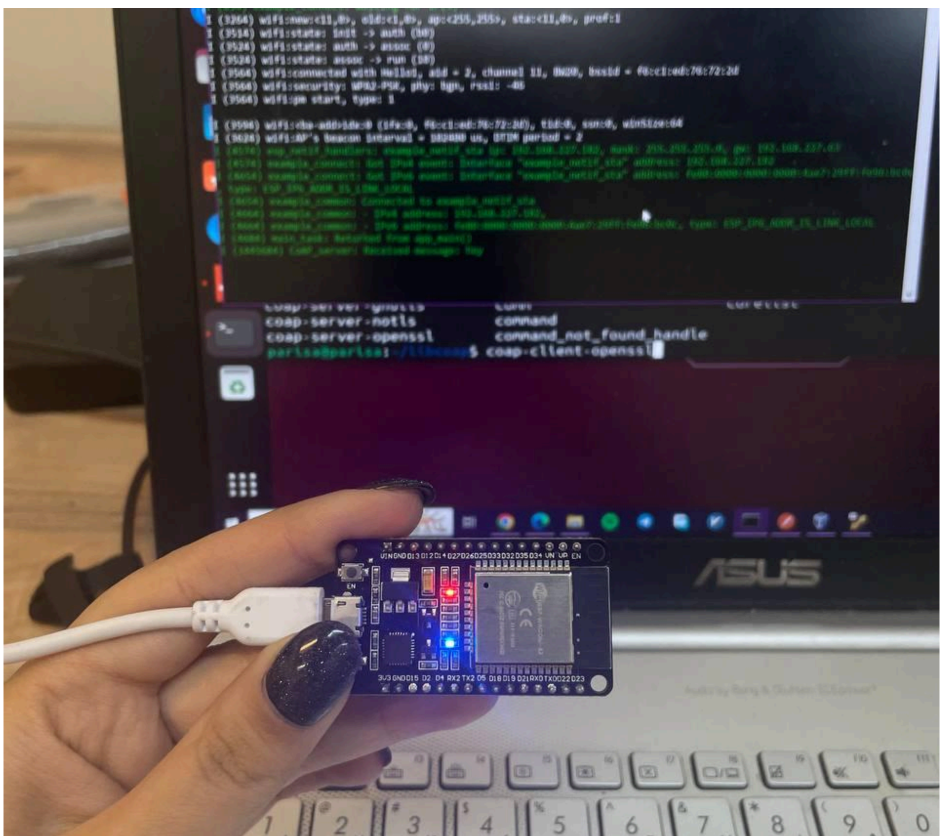
It's listening..

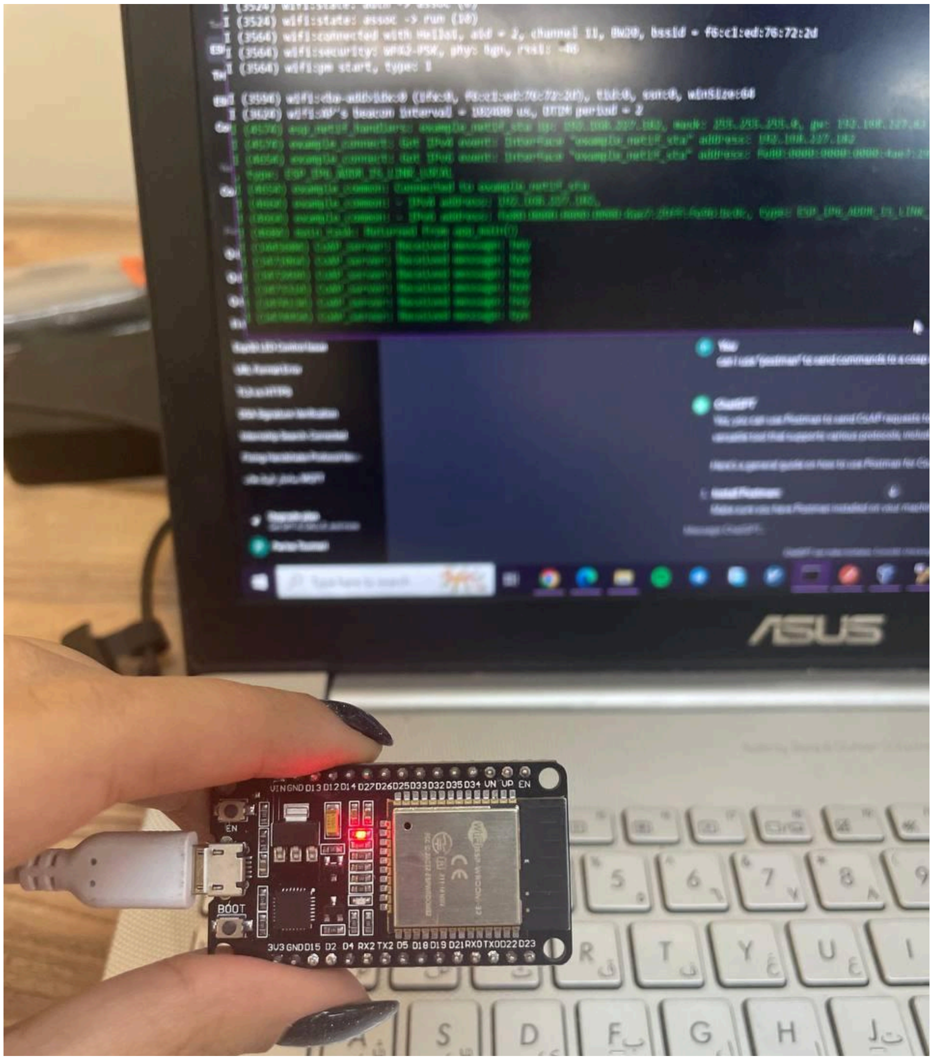
Sending the message using CoAP client (installed on an Ubuntu system)

Receiving the commands



Results:





2. (using WireShark)
Wireshark is a powerful tool widely employed for the analysis of messages and packets exchanged within a network. This report discusses the application of Wireshark to monitor and filter messages based on the CoAP (Constrained Application Protocol) protocol. The goal is to demonstrate the bidirectional communication between a system and an ESP module, focusing on CoAP protocol messages.
Wireshark is an open-source packet analyzer that allows users to capture and inspect the data traveling back and forth on a network. It provides detailed information about the communication patterns, protocols used, and the content of the exchanged packets. Wireshark supports a wide range of network protocols, making it a versatile tool for network troubleshooting, analysis, and debugging.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::4ae7:29ff:fe9…	ff02::1:ff96:bc0c	ICMPv6	86	Multicast Listener Report
2	10.883972559	192.168.227.221	91.189.91.157	NTP	90	NTP Version 4, client
3	11.457536607	91.189.91.157	192.168.227.221	NTP	90	NTP Version 4, server
4	14.414649022	192.168.227.221	192.168.227.182	CoAP	60	CON, MID:15436, PUT, /Espressif
5	14.904447558	192.168.227.182	192.168.227.221	CoAP	46	ACK, MID:15436, 2.04 Changed
6	15.511820351	48:e7:29:96:bc:0c	Broadcast	ARP	42	ARP Announcement for 192.168.227.182
7	21.209975334	fe80::4ae7:29ff:fe9…	ff02::1:ff9f:7598	ICMPv6	86	Multicast Listener Report
8	21.619262955	f6:c1:ed:76:72:2d	LiteonTe_ef:f9:41	ARP	42	Who has 192.168.227.221? Tell 192.168.227.63
9	21.619289640	LiteonTe_ef:f9:41	f6:c1:ed:76:72:2d	ARP	42	192.168.227.221 is at e8:d0:fc:ef:f9:41
10	23.460601268	192.168.227.221	192.168.227.182	CoAP	59	CON, MID:5840, PUT, /Espressif
11	23.863154764	192.168.227.182	192.168.227.221	CoAP	46	ACK, MID:5840, 2.04 Changed
12	28.582096140	LiteonTe_ef:f9:41	48:e7:29:96:bc:0c	ARP	42	Who has 192.168.227.182? Tell 192.168.227.221
13	28.971893416	48:e7:29:96:bc:0c	LiteonTe_ef:f9:41	ARP	42	192.168.227.182 is at 48:e7:29:96:bc:0c

```
   Frame Number: 4
   Frame Length: 60 bytes (480 bits)
   Capture Length: 60 bytes (480 bits)
   [Frame is marked: False]
   [Frame is ignored: False]
   [Protocols in frame: eth:ethertype:ip:udp:coap:data]
   [Coloring Rule Name: UDP]
   [Coloring Rule String: udp]
▾ Ethernet II, Src: LiteonTe_ef:f9:41 (e8:d0:fc:ef:f9:41), Dst: 48:e7:29:96:bc:0c (48:e7:29:96:bc:0c)
 ▸ Destination: 48:e7:29:96:bc:0c (48:e7:29:96:bc:0c)
 ▸ Source: LiteonTe_ef:f9:41 (e8:d0:fc:ef:f9:41)
   Type: IPv4 (0x0800)
▾ Internet Protocol Version 4, Src: 192.168.227.221, Dst: 192.168.227.182
   0100 .... = Version: 4
   .... 0101 = Header Length: 20 bytes (5)
 ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
   Total Length: 46
   Identification: 0x0758 (1880)
 ▸ Flags: 0x40, Don't fragment
   ...0 0000 0000 0000 = Fragment Offset: 0
   Time to Live: 64
   Protocol: UDP (17)
   Header Checksum: 0xea81 [validation disabled]
   [Header checksum status: Unverified]
   Source Address: 192.168.227.221
   Destination Address: 192.168.227.182
▾ User Datagram Protocol, Src Port: 56862, Dst Port: 5683
   Source Port: 56862
   Destination Port: 5683
   Length: 26
   Checksum: 0x4911 [unverified]
   [Checksum Status: Unverified]
   [Stream index: 1]
 ▸ [Timestamps]
   UDP payload (18 bytes)
▾ Constrained Application Protocol, Confirmable, PUT, MID:15436
   01.. .... = Version: 1
   ..00 .... = Type: Confirmable (0)
   .... 0000 = Token Length: 0
   Code: PUT (3)
   Message ID: 15436
 ▸ Opt Name: #1: Uri-Path: Espressif
   End of options marker: 255
 ▸ Payload: Payload Content-Format: application/octet-stream (no Content-Format), Length: 3
   [Uri-Path: /Espressif]
▾ Data (3 bytes)
   Data: 6f6666
   [Length: 3]

0000  48 e7 29 96 bc 0c e8 d0  fc ef f9 41 08 00 45 00   H·)····· ···A··E·
0010  00 2e 07 58 40 00 40 11  ea 81 c0 a8 e3 dd c0 a8   ·.·X@·@· ········
0020  e3 b6 de 1e 16 33 00 1a  49 11 40 03 3c 4c b9 45   ·····3·· I·@·<L·E
```