HW3 SIOT

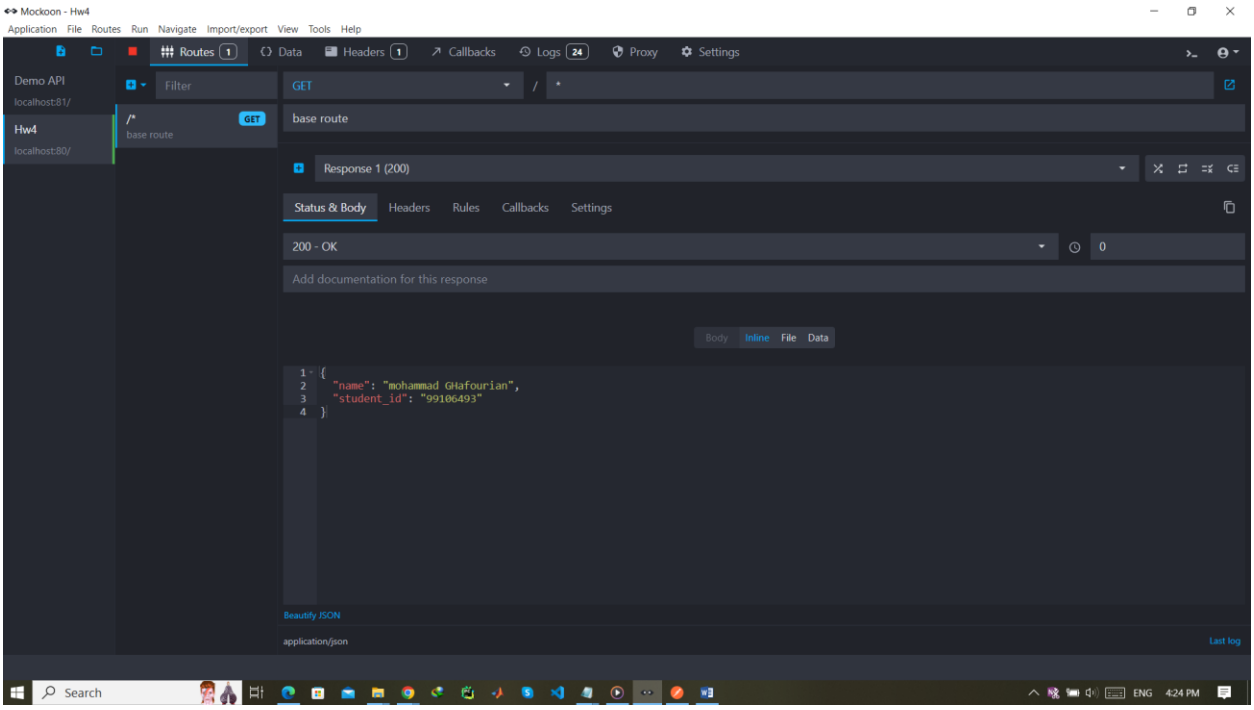
MOHAMMAD GHAFOURIANAHAHASSANPOUR

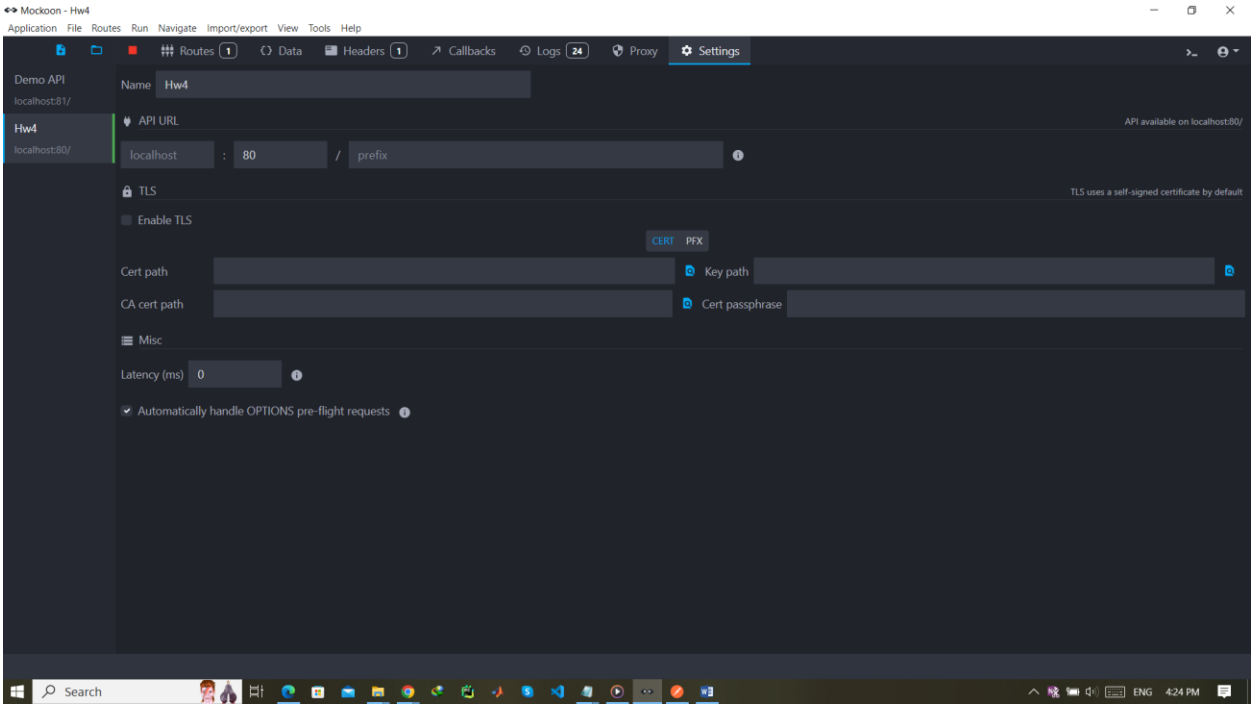
STUDENT NUMBER: 99106493

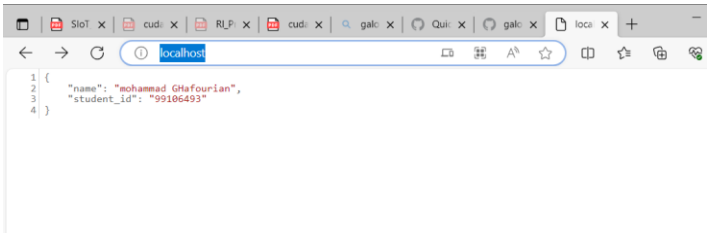
DR SIAVASH AHMADI

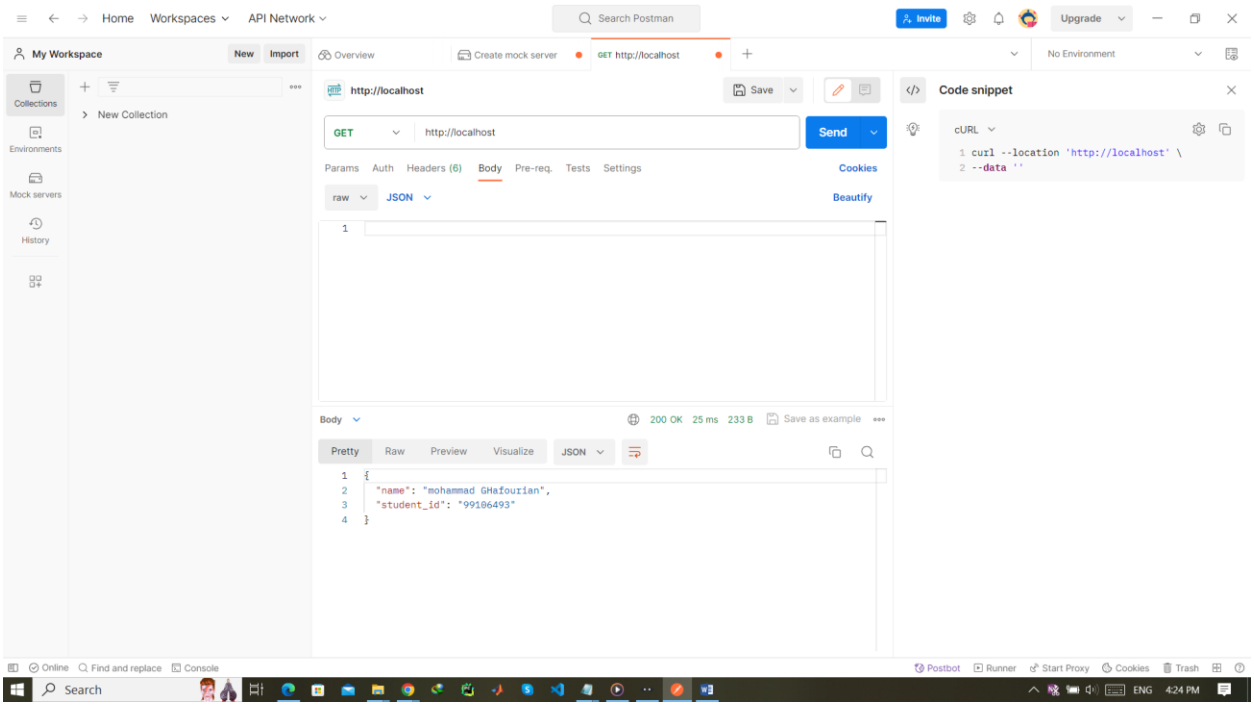
1: INITIAL SETUP:

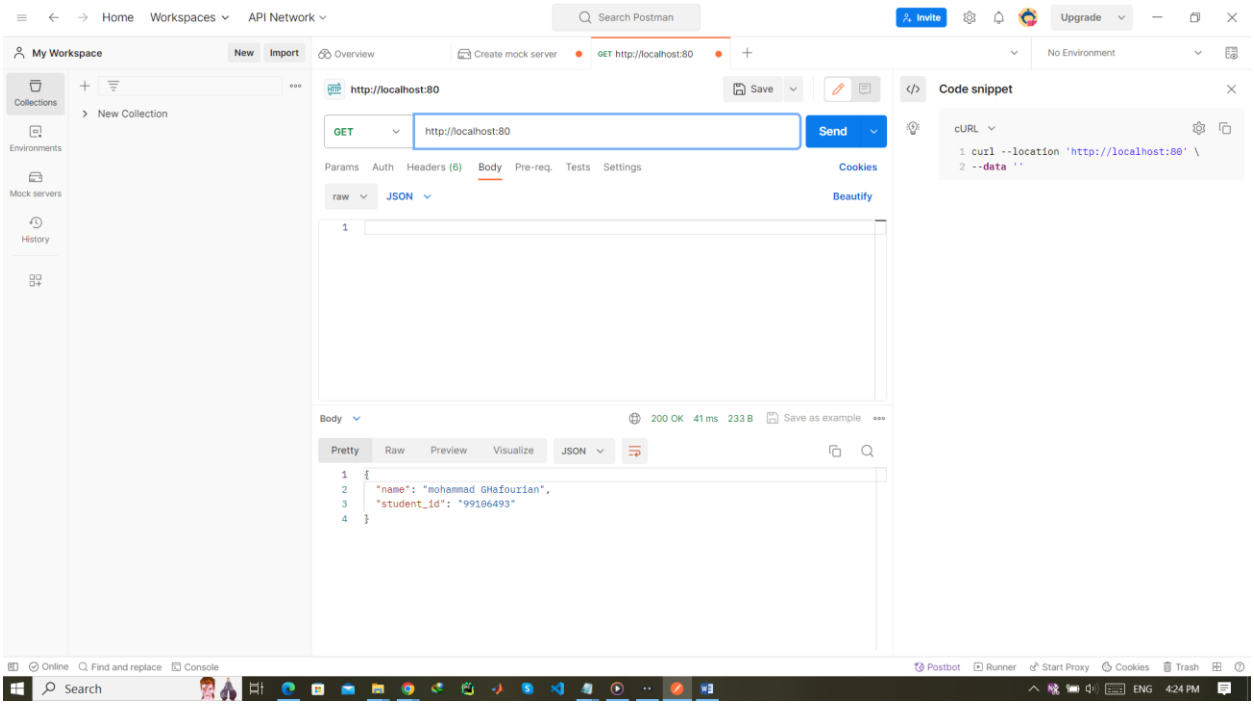
Using Mockoon we setup a mock server and the we loaded the site with edge and for being sure we GET the site with postman.






localhost
{
"name": "mohammad GHafourian",
"student_id": "99106493"
}


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
Create mock server
GET http://localhost
No Environment
Collections
Environments
Mock servers
History
New Collection
http://localhost
Save
Code snippet
GET
http://localhost
Send
Params
Auth
Headers (6)
Body
Pre-req.
Tests
Settings
Cookies
raw
JSON
Beautify
cURL
curl --location 'http://localhost' \
--data ''
Body
200 OK
25 ms
233 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"name": "mohammad GHafourian",
"student_id": "99106493"
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
Search
ENG
4:24 PM


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
Create mock server
GET http://localhost:80
No Environment
Collections
Environments
Mock servers
History
New Collection
http://localhost:80
Save
Code snippet
GET
http://localhost:80
Send
Params
Auth
Headers (6)
Body
Pre-req.
Tests
Settings
Cookies
raw
JSON
Beautify
cURL
curl --location 'http://localhost:80' \
--data ''
Body
200 OK
41 ms
233 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"name": "mohammad GHafourian",
"student_id": "99106493"
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
Search
ENG
4:24 PM

2 ESP AS CLIENT:

The IP for the computer

```
Unknown adapter Local Area Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 9:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 12:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::1699:1d02:429f:5d26%5
   IPv4 Address. . . . . . . . . . . : 192.168.50.149
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.50.184

C:\Users\Admin>
```

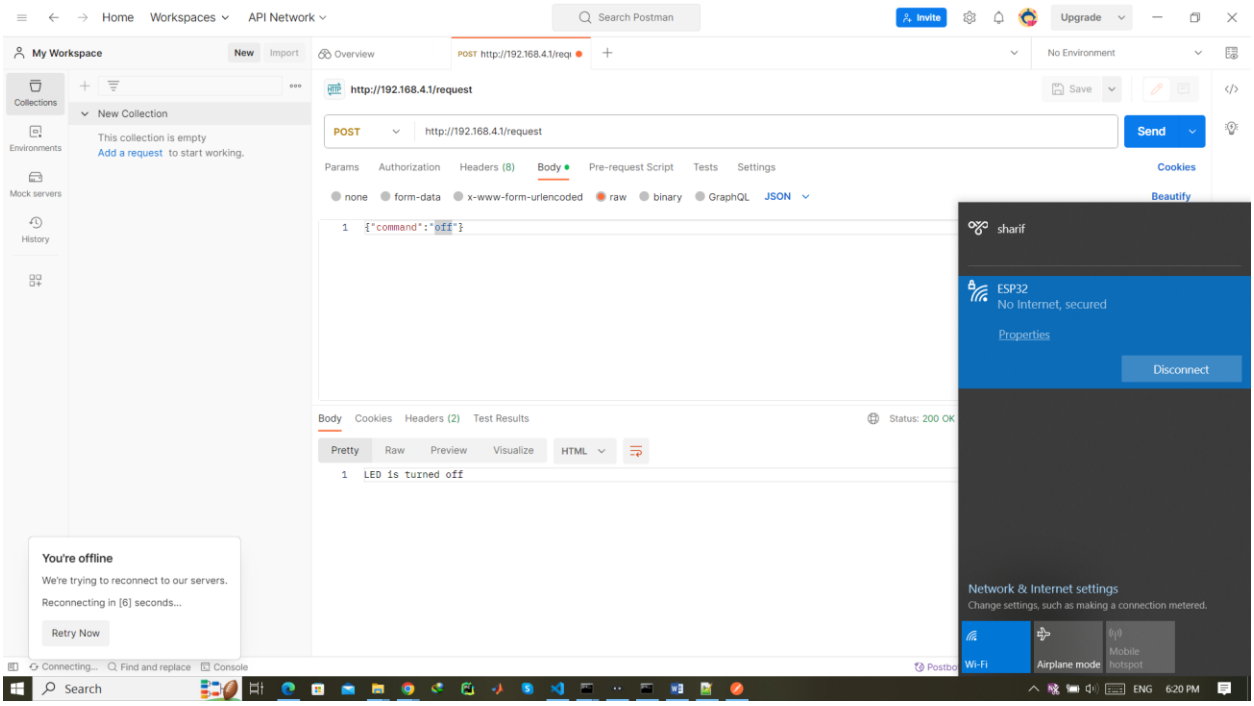
```
I (699) wifi_init: tcp mss: 1440
I (699) wifi_init: WiFi IRAM OP enabled
I (699) wifi_init: WiFi RX IRAM OP enabled
I (709) phy_init: phy_version 4771,450c73b,Aug 16 2023,11:03:10
I (789) wifi:mode : sta (48:e7:29:96:bc:0c)
I (789) wifi:enable tsf
WiFi connecting ...
I (809) wifi:new:<6,0>, old:<1,0>, ap:<255,255>, sta:<6,0>, prof:1
I (809) wifi:state: init -> auth (b0)
I (819) wifi:state: auth -> assoc (0)
I (829) wifi:state: assoc -> run (10)
I (899) wifi:connected with Moh, aid = 2, channel 6, BW20, bssid = ea:a9:73:27:ab:26
I (899) wifi:security: WPA2-PSK, phy: bgn, rssi: -27
I (909) wifi:pm start, type: 1

WiFi connected ...
I (929) wifi:AP's beacon interval = 102400 us, DTIM period = 2
I (929) wifi:<ba-add>idx:0 (ifx:0, ea:a9:73:27:ab:26), tid:0, ssn:0, winSize:64
I (1909) esp_netif_handlers: sta ip: 192.168.50.182, mask: 255.255.255.0, gw: 192.168.50.184
WiFi got IP ...

WIFI was initiated ...........

HTTP_EVENT_ON_DATA: {
  "name": "mohammad GHafourian",
  "student_id": "99106493"
}
```

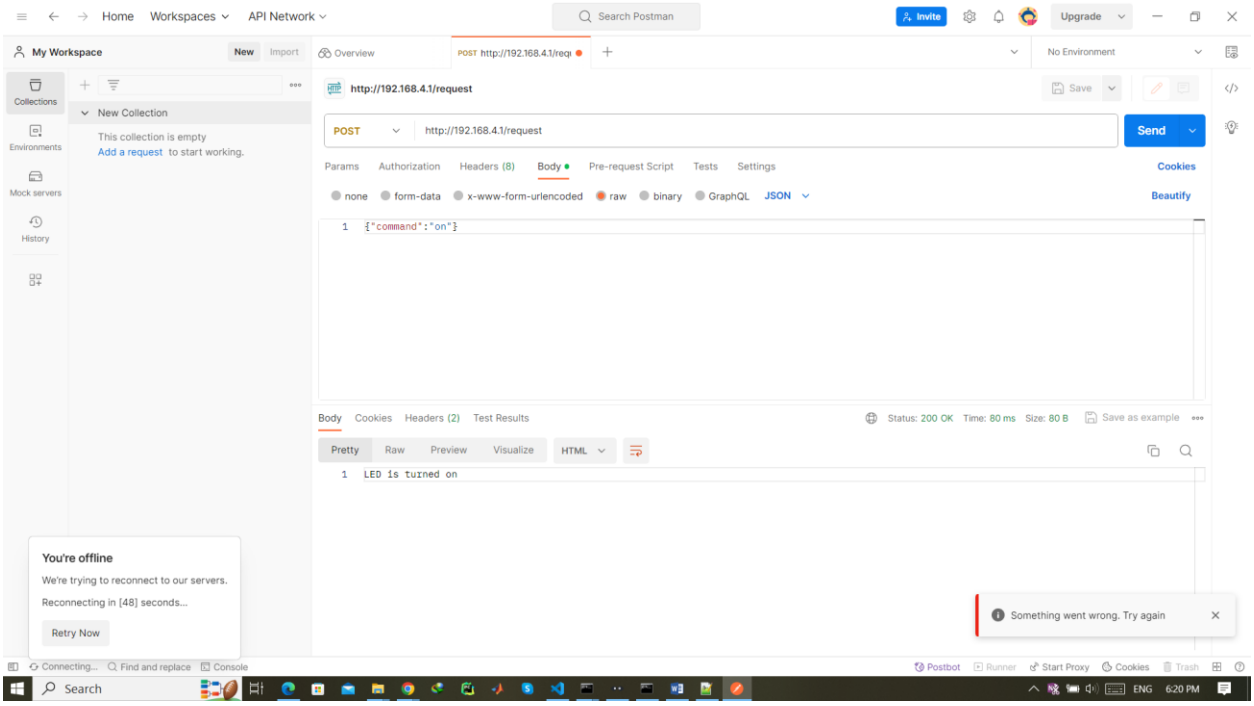
The first step is to connect to network is as same as we did in MQTT protocol then we should load the site page and in the config the method has been setted to GET method and finally in **client_event_get_handler** function we print the data in that site page.

```
}

esp_err_t client_event_get_handler(esp_http_client_event_handle_t evt)
{
    switch (evt->event_id)
    {
    case HTTP_EVENT_ON_DATA:
        printf("HTTP_EVENT_ON_DATA: %.*s\n", evt->data_len, (char *)evt->data);
        break;

    default:
        break;
    }
    return ESP_OK;
}

static void rest_get()
{
    esp_http_client_config_t config_get = {
        .url = "http://192.168.50.149:80/",
        .method = HTTP_METHOD_GET,
        .cert_pem = NULL,
        .event_handler = client_event_get_handler};

    esp_http_client_handle_t client = esp_http_client_init(&config_get);
    esp_http_client_perform(client);
    esp_http_client_cleanup(client);
}
```

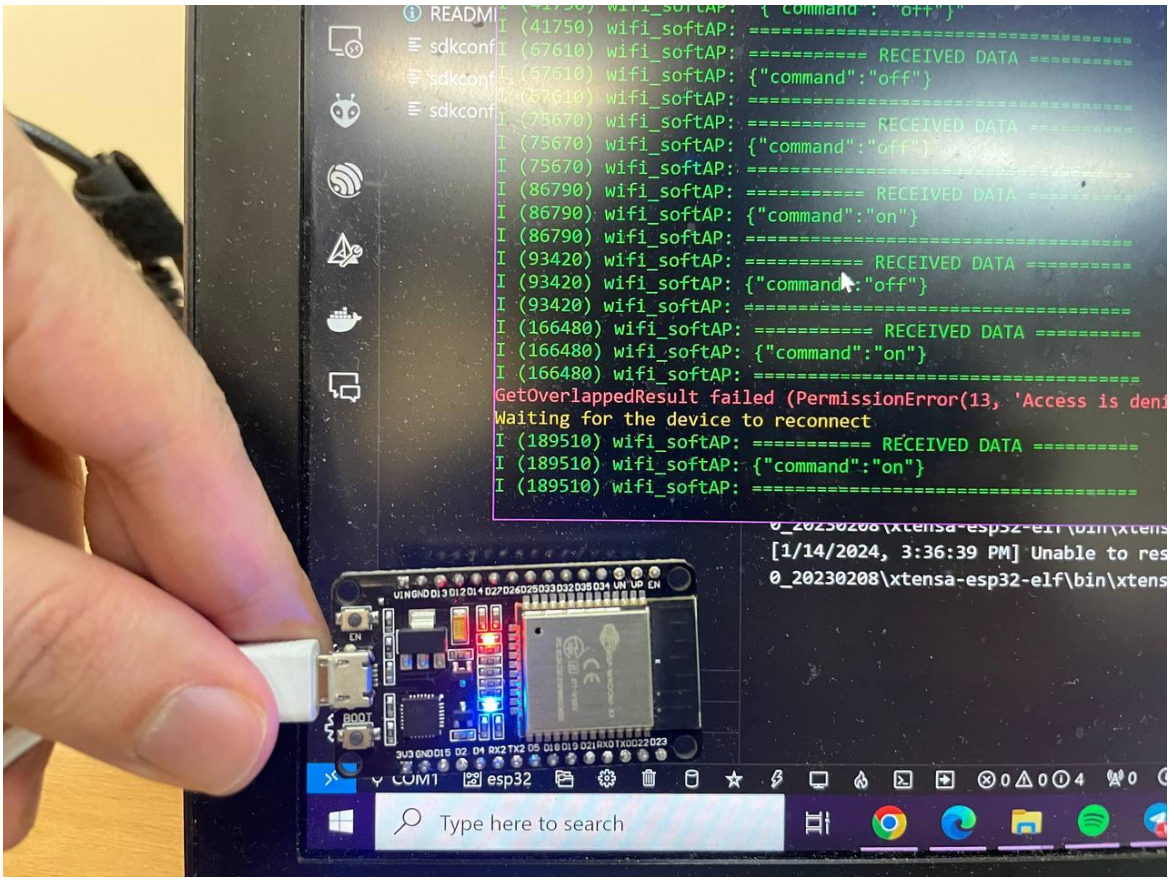
3 ESP AS SERVER:

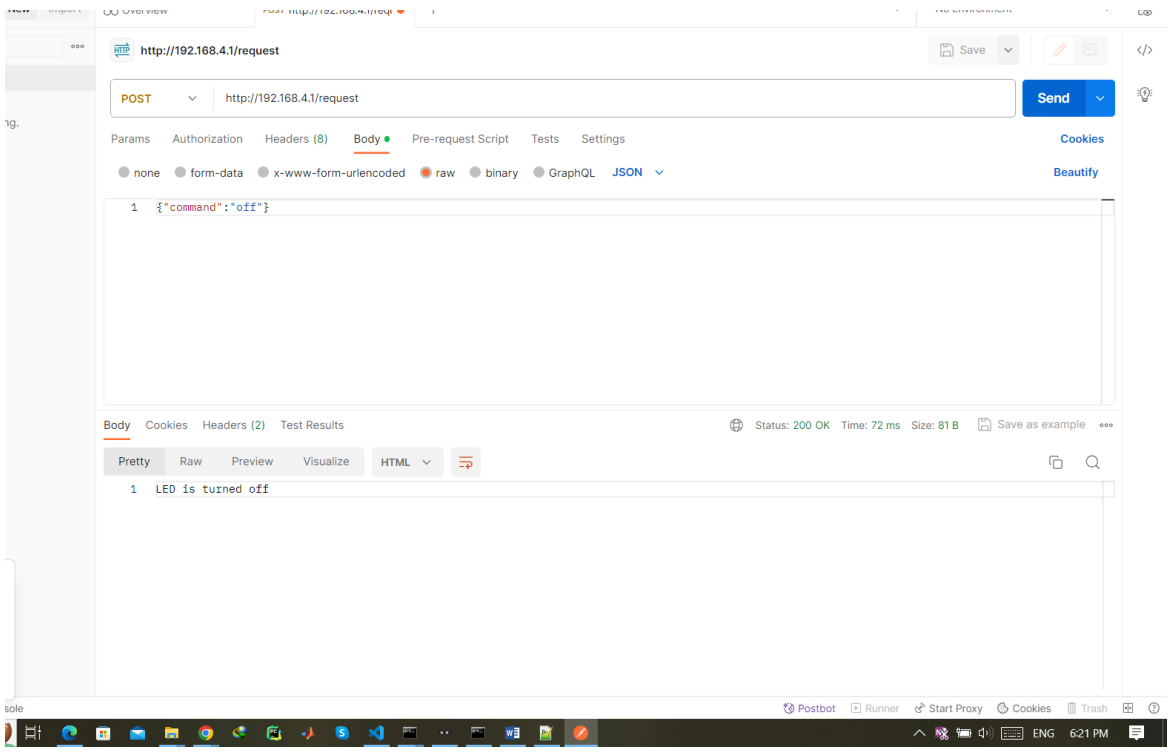
First we connect the pc to

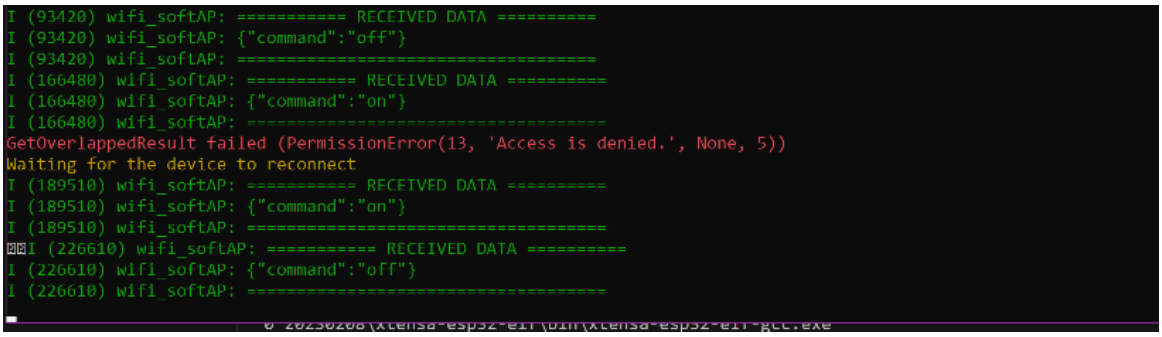


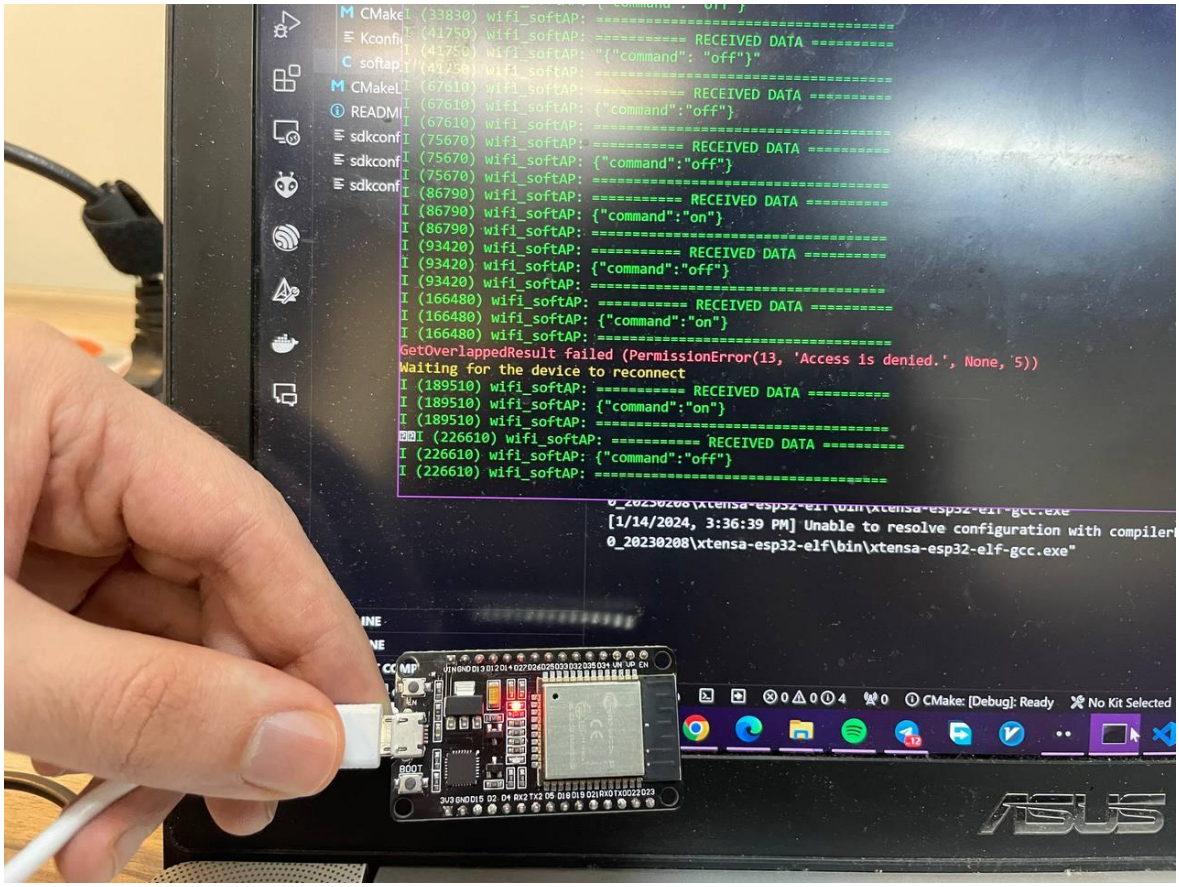



```
I (86790) wifi_softAP: ========== RECEIVED DATA ==========
I (86790) wifi_softAP: {"command":"on"}
I (86790) wifi_softAP: ==================================
I (93420) wifi_softAP: ========== RECEIVED DATA ==========
I (93420) wifi_softAP: {"command":"off"}
I (93420) wifi_softAP: ==================================
I (166480) wifi_softAP: ========== RECEIVED DATA ==========
I (166480) wifi_softAP: {"command":"on"}
I (166480) wifi_softAP: ==================================
GetOverlappedResult failed (PermissionError(13, 'Access is denied.', None, 5))
Waiting for the device to reconnect
I (189510) wifi_softAP: ========== RECEIVED DATA ==========
I (189510) wifi_softAP: {"command":"on"}
I (189510) wifi_softAP: ==================================
```


I (41750) wifi_softAP: ==========
I (67610) wifi_softAP: ========== RECEIVED DATA ==========
I (67610) wifi_softAP: {"command":"off"}
I (67610) wifi_softAP: ==========
I (75670) wifi_softAP: ========== RECEIVED DATA ==========
I (75670) wifi_softAP: {"command":"off"}
I (75670) wifi_softAP: ==========
I (86790) wifi_softAP: ========== RECEIVED DATA ==========
I (86790) wifi_softAP: {"command":"on"}
I (86790) wifi_softAP: ==========
I (93420) wifi_softAP: ========== RECEIVED DATA ==========
I (93420) wifi_softAP: {"command":"off"}
I (93420) wifi_softAP: ==========
I (166480) wifi_softAP: ========== RECEIVED DATA ==========
I (166480) wifi_softAP: {"command":"on"}
I (166480) wifi_softAP: ==========
GetOverlappedResult failed (PermissionError(13, 'Access is deni
Waiting for the device to reconnect
I (189510) wifi_softAP: ========== RECEIVED DATA ==========
I (189510) wifi_softAP: {"command":"on"}
I (189510) wifi_softAP: ==========
[1/14/2024, 3:36:39 PM] Unable to res
0_20230208\xtensa-esp32-elf\bin\xtens
COM1
esp32
Type here to search


http://192.168.4.1/request
Save
POST
http://192.168.4.1/request
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
1 {"command":"off"}
Body
Cookies
Headers (2)
Test Results
Status: 200 OK
Time: 72 ms
Size: 81 B
Save as example
Pretty
Raw
Preview
Visualize
HTML
1 LED is turned off
Postbot
Runner
Start Proxy
Cookies
Trash
ENG
6:21 PM





In wifi_init_softap function the server has been initialized, the function which is the network we are going to connect to it the network name is “ESP32” and it password is “12345678” and we set a condition to have an open internet network to connect to it without password. As you can see it is an open network when the password is less than 8 digits (it should be at least 8 character to have a secure authenticated needed network)

```c
#define EXAMPLE_ESP_WIFI_SSID      "ESP32"
#define EXAMPLE_ESP_WIFI_PASS      "12345678"
#define EXAMPLE_MAX_STA_CONN       4

static const char *TAG = "wifi_softAP";

void wifi_init_softap() {
    ESP_ERROR_CHECK(esp_netif_init());
    ESP_ERROR_CHECK(esp_event_loop_create_default());
    esp_netif_create_default_wifi_ap();

    wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK(esp_wifi_init(&cfg));

    wifi_config_t wifi_config = {
        .ap = {
            .ssid = EXAMPLE_ESP_WIFI_SSID,
            .ssid_len = strlen(EXAMPLE_ESP_WIFI_SSID),
            .password = EXAMPLE_ESP_WIFI_PASS,
            .max_connection = EXAMPLE_MAX_STA_CONN,
            .authmode = WIFI_AUTH_WPA_WPA2_PSK
        },
    };

    if (strlen(EXAMPLE_ESP_WIFI_PASS) < 8) {
        // Set to open network if password is less than 8 characters
        wifi_config.ap.authmode = WIFI_AUTH_OPEN;
    }

    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_AP));
    ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_AP, &wifi_config));
    ESP_ERROR_CHECK(esp_wifi_start());

    ESP_LOGI(TAG, "Wi-Fi AP set with SSID: %s", EXAMPLE_ESP_WIFI_SSID);
}
```

```c
/* Function to handle HTTP POST requests */
esp_err_t echo_post_handler(httpd_req_t *req)
{
    char buf[100];
    int ret, remaining = req->content_len;

    while (remaining > 0) {
        /* Read the data for the request */
        if ((ret = httpd_req_recv(req, buf, MIN(remaining, sizeof(buf)))) <= 0) {
            if (ret == HTTPD_SOCK_ERR_TIMEOUT) {
                /* Retry receiving if timeout occurred */
                continue;
            }
            return ESP_FAIL;
        }
        buf[ret] = '\0';

        // Check for LED command
        if (strcmp(buf,"{\"command\":\"on\"}") == 0) {
            gpio_set_level(GPIO_NUM_2, 1); // Turn LED on
            httpd_resp_send(req, "LED is turned on", HTTPD_RESP_USE_STRLEN);
        }
        else if (strcmp(buf, "{\"command\":\"off\"}") == 0) {
            gpio_set_level(GPIO_NUM_2, 0); // Turn LED off
            httpd_resp_send(req, "LED is turned off", HTTPD_RESP_USE_STRLEN);
        }
        else {
            httpd_resp_send(req, "Command not recognized", HTTPD_RESP_USE_STRLEN);
        }
        /* Send back the same data */
        httpd_resp_send_chunk(req, buf, ret);
        remaining -= ret;

        /* Log data received */
        ESP_LOGI(TAG, "========== RECEIVED DATA ==========");
        ESP_LOGI(TAG, "%.*s", ret, buf);
        ESP_LOGI(TAG, "====================================");
    }

    // End response
    httpd_resp_send_chunk(req, NULL, 0);
    return ESP_OK;
}
```

In echo_post_handler we are going to handle the posts request and send the responds to the response URL, in this function we check the massage that has been received and control the LED.

```
httpd_handle_t start_webserver(void) {
httpd_config_t config = HTTPD_DEFAULT_CONFIG();

    // Start the httpd server
    httpd_handle_t server = NULL;
    if (httpd_start(&server, &config) == ESP_OK) {
        // Set URI handlers
        httpd_uri_t get_uri = {
            .uri       = "/response",
            .method    = HTTP_GET,
            .handler   = get_handler,
            .user_ctx  = NULL
        };
        httpd_register_uri_handler(server, &get_uri);

        httpd_uri_t post_uri = {
            .uri       = "/request",
            .method    = HTTP_POST,
            .handler   = echo_post_handler,
            .user_ctx  = NULL
        };
        httpd_register_uri_handler(server, &post_uri);
    }
    return server;
}
```

In start_server function we initial our servers, which are 192.168.4.1/response and 192.168.4.1/request, we request to request page .