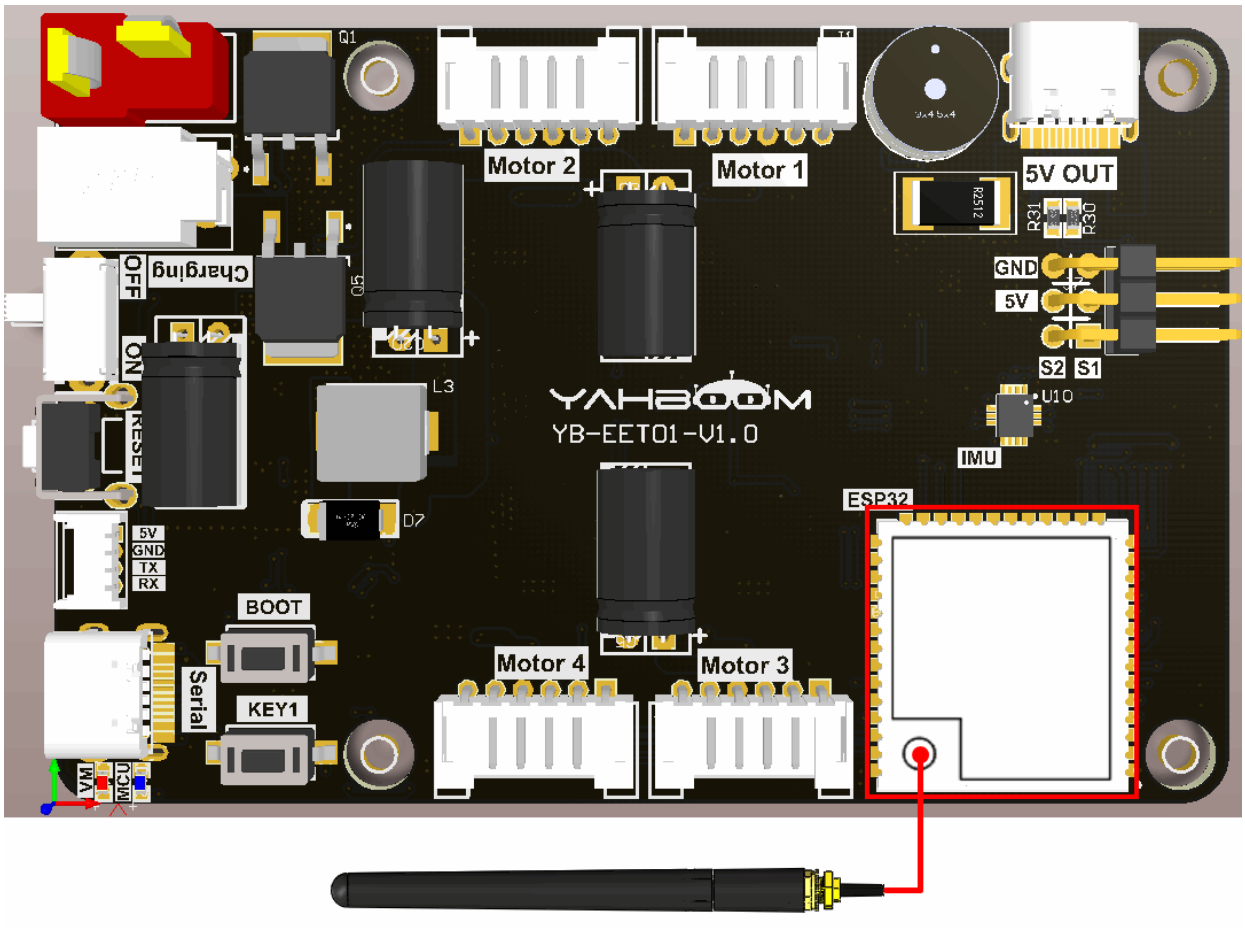
WiFi networking

1. Experimental purpose

Use the ESP32S3 core module of the microROS control board to learn the function of ESP32 connecting to WiFi.

2. Hardware connection

As shown in the figure below, the microROS control board integrates the ESP32-S3-WROOM core module, which has its own wireless WiFi function. The ESP32-S3 core module needs to be connected to an antenna, and a type-C data cable needs to be connected to the computer and the microROS control board as Burn firmware function.



3. Core code analysis

The virtual machine path corresponding to the program source code is as follows

```
~/esp/Samples/esp32_samples/wifi_sta
```

First, get the WiFi name and password to connect from the IDF configuration tool.

```
#define EXAMPLE_ESP_WIFI_SSID      CONFIG_ESP_WIFI_SSID
#define EXAMPLE_ESP_WIFI_PASS      CONFIG_ESP_WIFI_PASSWORD
#define EXAMPLE_ESP_MAXIMUM_RETRY  CONFIG_ESP_MAXIMUM_RETRY
```

Initialize WiFi to STA mode, and configure the WiFi hotspot name, password, etc. to be connected.

```
void wifi_init_sta(void)
{
    s_wifi_event_group = xEventGroupCreate();

    ESP_ERROR_CHECK(esp_netif_init());

    ESP_ERROR_CHECK(esp_event_loop_create_default());
    esp_netif_create_default_wifi_sta();

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

    esp_event_handler_instance_t instance_any_id;
    esp_event_handler_instance_t instance_got_ip;
    ESP_ERROR_CHECK(esp_event_handler_instance_register(WIFI_EVENT,
                                                        ESP_EVENT_ANY_ID,
                                                        &event_handler,
                                                        NULL,
                                                        &instance_any_id));
    ESP_ERROR_CHECK(esp_event_handler_instance_register(IP_EVENT,
                                                        IP_EVENT_STA_GOT_IP,
                                                        &event_handler,
                                                        NULL,
                                                        &instance_got_ip));

    wifi_config_t wifi_config = {
        .sta = {
            .ssid = EXAMPLE_ESP_WIFI_SSID,
            .password = EXAMPLE_ESP_WIFI_PASS,
            /* Authmode threshold resets to WPA2 as default if password matches WPA2
standards (pasword len => 8).
             * If you want to connect the device to deprecated WEP/WPA networks,
Please set the threshold value
```

```
             * to WIFI_AUTH_WEP/WIFI_AUTH_WPA_PSK and set the password with length
and format matching to
             * WIFI_AUTH_WEP/WIFI_AUTH_WPA_PSK standards.
             */
            .threshold.authmode = ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD,
            .sae_pwe_h2e = ESP_WIFI_SAE_MODE,
            .sae_h2e_identifier = EXAMPLE_H2E_IDENTIFIER,
        },
    };
    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA) );
    ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_STA, &wifi_config) );
    ESP_ERROR_CHECK(esp_wifi_start() );

    ESP_LOGI(TAG, "wifi_init_sta finished.");

    /* Waiting until either the connection is established (WIFI_CONNECTED_BIT) or
connection failed for the maximum
     * number of re-tries (WIFI_FAIL_BIT). The bits are set by event_handler() (see
above) */
    EventBits_t bits = xEventGroupWaitBits(s_wifi_event_group,
            WIFI_CONNECTED_BIT | WIFI_FAIL_BIT,
            pdFALSE,
            pdFALSE,
            portMAX_DELAY);

    /* xEventGroupWaitBits() returns the bits before the call returned, hence we can
test which event actually
     * happened. */
    if (bits & WIFI_CONNECTED_BIT) {
        ESP_LOGI(TAG, "connected to ap SSID:%s password:%s",
                 EXAMPLE_ESP_WIFI_SSID, EXAMPLE_ESP_WIFI_PASS);
    } else if (bits & WIFI_FAIL_BIT) {
        ESP_LOGI(TAG, "Failed to connect to SSID:%s, password:%s",
                 EXAMPLE_ESP_WIFI_SSID, EXAMPLE_ESP_WIFI_PASS);
    } else {
        ESP_LOGE(TAG, "UNEXPECTED EVENT");
    }
}
```

Try to connect to the WiFi hotspot. If the connection is successful, go to the next step. If the connection is not successful after exceeding the number of times, it will automatically exit.

```
static void event_handler(void* arg, esp_event_base_t event_base,
                                int32_t event_id, void* event_data)
{
    if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_START) {
        esp_wifi_connect();
    } else if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_DISCONNECTED)
{
        if (reconnect_count < EXAMPLE_ESP_MAXIMUM_RETRY) {
            esp_wifi_connect();
```

```
            reconnect_count++;
            ESP_LOGI(TAG, "retry to connect to the AP");
        } else {
            xEventGroupSetBits(s_wifi_event_group, WIFI_FAIL_BIT);
        }
        ESP_LOGI(TAG,"connect to the AP fail");
    } else if (event_base == IP_EVENT && event_id == IP_EVENT_STA_GOT_IP) {
        ip_event_got_ip_t* event = (ip_event_got_ip_t*) event_data;
        ESP_LOGI(TAG, "got ip:" IPSTR, IP2STR(&event->ip_info.ip));
        reconnect_count = 0;
        xEventGroupSetBits(s_wifi_event_group, WIFI_CONNECTED_BIT);
    }
}
```

Initialize the WiFi network connection in the app_main function and enable the function of connecting to WiFi.

```
void app_main(void)
{
    printf("hello yahboom\n");
    ESP_LOGI(TAG, "Nice to meet you!");

    Wifi_Net_Init();
}
```

4. Compile, download and flash firmware

Use a Type-C data cable to connect the virtual machine/computer and the microROS control board. If the system pops up, choose to connect to the virtual machine.

Activate the ESP-IDF development environment. Note that every time you open a new terminal, you need to activate the ESP-IDF development environment before compiling the firmware.

```
source ~/esp/esp-idf/export.sh
```

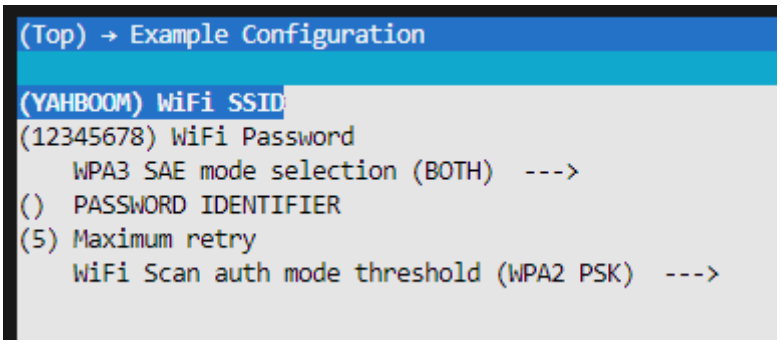
Enter the project directory

```
cd ~/esp/Samples/esp32_samples/wifi_sta
```

Open the ESP-IDF configuration tool.

```
idf.py menuconfig
```

Open Example Configuration and fill in your WiFi name and password in the WiFi SSID and WiFi Password fields.



After modification, press S to save, and then press Q to exit the configuration tool.

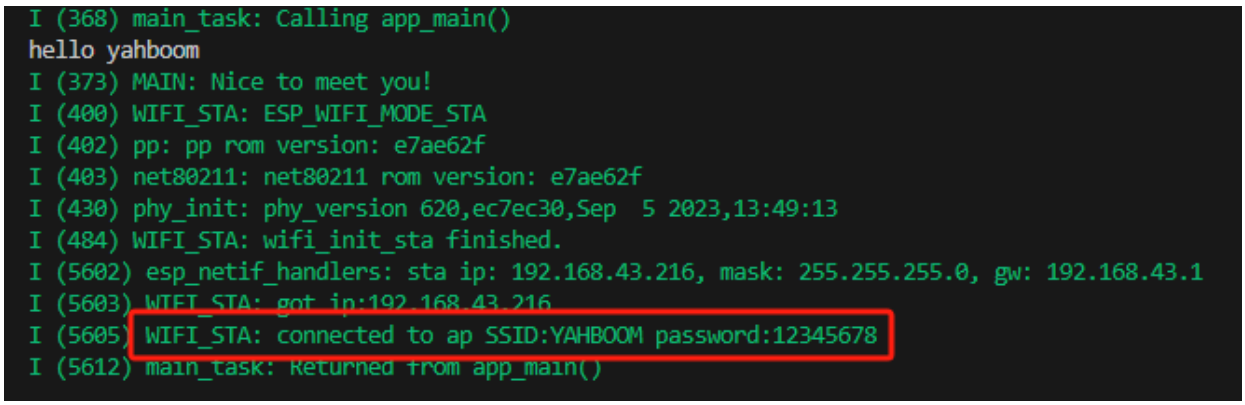
Compile, flash, and open the serial port simulator

```
idf.py build flash monitor
```

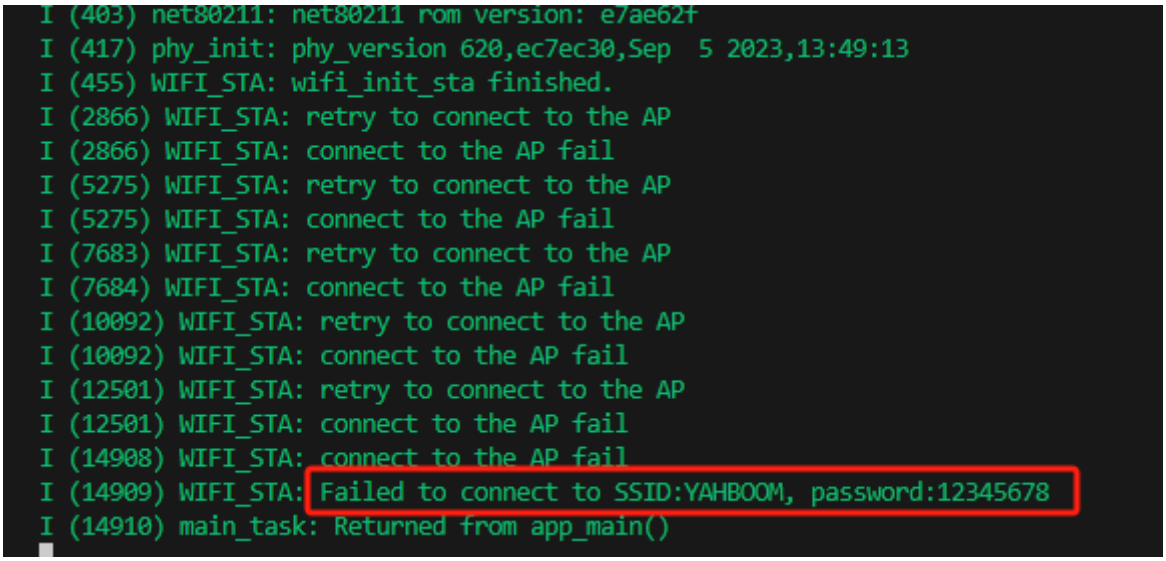
If you need to exit the serial port simulator, press **Ctrl+]**.

5. Experimental results

The serial port simulator prints the "hello yahboom" greeting and tries to connect to the WiFi hotspot. As shown in the figure below, if the connection is successful, the name and password of the hotspot with successful connection will be printed.



If the connection attempt fails more than a certain number of times, a failure will be reported and the connection will be terminated.


I (403) net80211: net80211 rom version: e7ae62f
I (417) phy_init: phy_version 620,ec7ec30,Sep 5 2023,13:49:13
I (455) WIFI_STA: wifi_init_sta finished.
I (2866) WIFI_STA: retry to connect to the AP
I (2866) WIFI_STA: connect to the AP fail
I (5275) WIFI_STA: retry to connect to the AP
I (5275) WIFI_STA: connect to the AP fail
I (7683) WIFI_STA: retry to connect to the AP
I (7684) WIFI_STA: connect to the AP fail
I (10092) WIFI_STA: retry to connect to the AP
I (10092) WIFI_STA: connect to the AP fail
I (12501) WIFI_STA: retry to connect to the AP
I (12501) WIFI_STA: connect to the AP fail
I (14908) WIFI_STA: connect to the AP fail
I (14909) WIFI_STA: Failed to connect to SSID:YAHBOOM, password:12345678
I (14910) main_task: Returned from app_main()