ESP-Moonlight

Nov 30, 2023

CONTENTS

1 ESP-Moonlight Support Policy **3**

2 Supported versions and chips **5**

3 Get Started **7**

3.1 □□ 7
3.2 □□□□ 9
3.3 □□□□ 14
3.4 Wi-Fi □□ 17
3.5 SoftAP □□□ Bluetooth Low Energy □□ 21
3.6 □□□□□□□□□□ 25
3.7 □□□□ 29
3.8 □□□□□□ 34

[??]

CHAPTER

ONE

ESP-MOONLIGHT SUPPORT POLICY

From December 2023, we will offer limited support on this project, but Pull Request is still welcomed!

CHAPTER

TWO

SUPPORTED VERSIONS AND CHIPS

ESP-Moonlight	Dependent ESP-IDF	Target	Support State
master	>=release/v4.4	ESP32 ESP32S3	Limited Support

CHAPTER THREE

GET STARTED

Building Products with ESP32 fast: MoonLight

Jumpstart Now

3.1 [illegible]

3.1.1 ESP-Moonlight[illegible] ESP32 [illegible]

ESP-Moonlight [illegible] ESP32 [illegible] LED [illegible]

[illegible]

- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible] OTA [illegible]
- [illegible]

Fig. 1: ESP-Moonlight

3.1.2 [illegible]

[illegible] ESP32-Moonlight V2.0 [illegible]

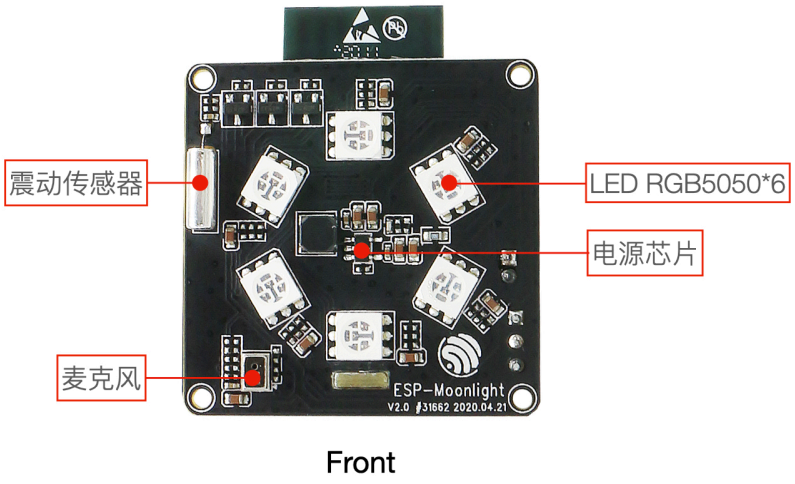

Fig. 2: ESP32-Moonlight [illegible]

3.2 [illegible]

[illegible] ESP32 [illegible] ESP32 [illegible]

3.2.1 [illegible]

[illegible] ESP32 [illegible]

[illegible] Linux、Windows [illegible] MacOS [illegible]ESP32 [illegible] USB [illegible] ESP-IDF ([illegible] SDK)[illegible]

3.2.2 [illegible]

- ESP32-Moonlight [illegible] ESP32 [illegible]
- USB [illegible]
- [illegible] PC、Windows、Linux [illegible] Mac OS、

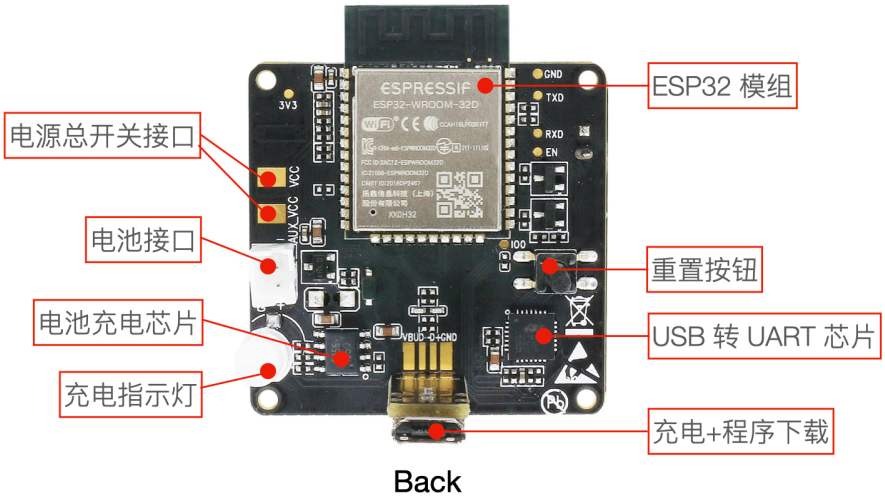

Fig. 3: ESP32-Moonlight [illegible]

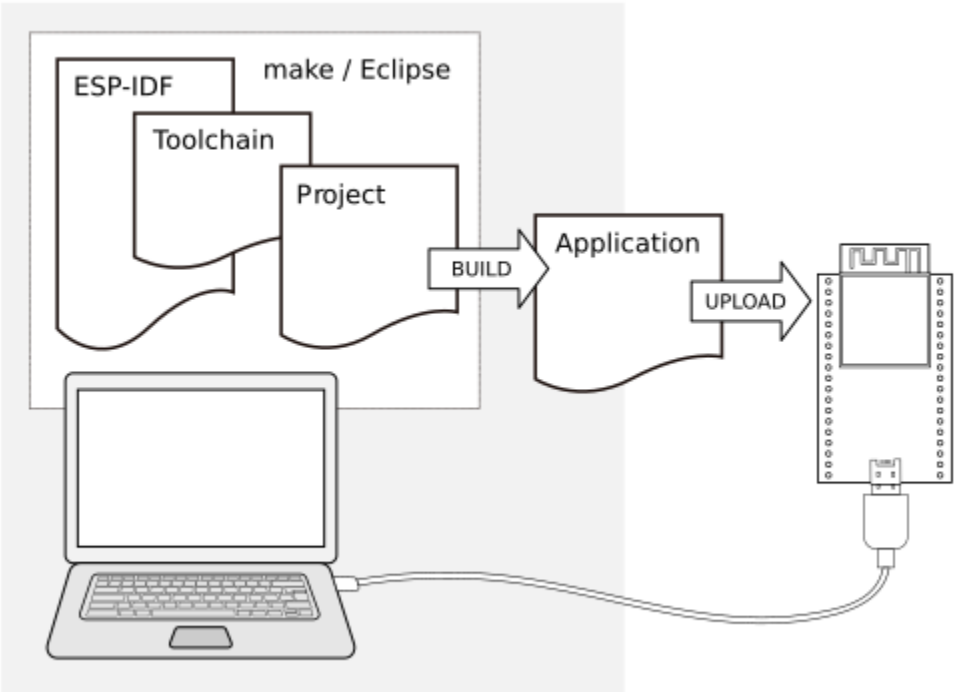

Fig. 4: ESP32 [illegible]

3.2.3 ESP-IDF ??

ESP-IDF ???? ESP32 ??????????

- ESP-IDF (Espressif IoT Development Framework) ?????????? ESP32 ?????????????????????????? ESP-ADF ??? MESH ????? ESP-MDF ?????????????? github ??
- ESP-IDF ?????????????????? ESP32 ??
- ESP-IDF ??? Windows?Mac?Linux ?????????? V4.0 ??

?? ESP-IDF

??? ESP-IDF ????????? ESP-IDF??????????????

??? ESP-IDF?

ESP-IDF ??

ESP-IDF ????????????

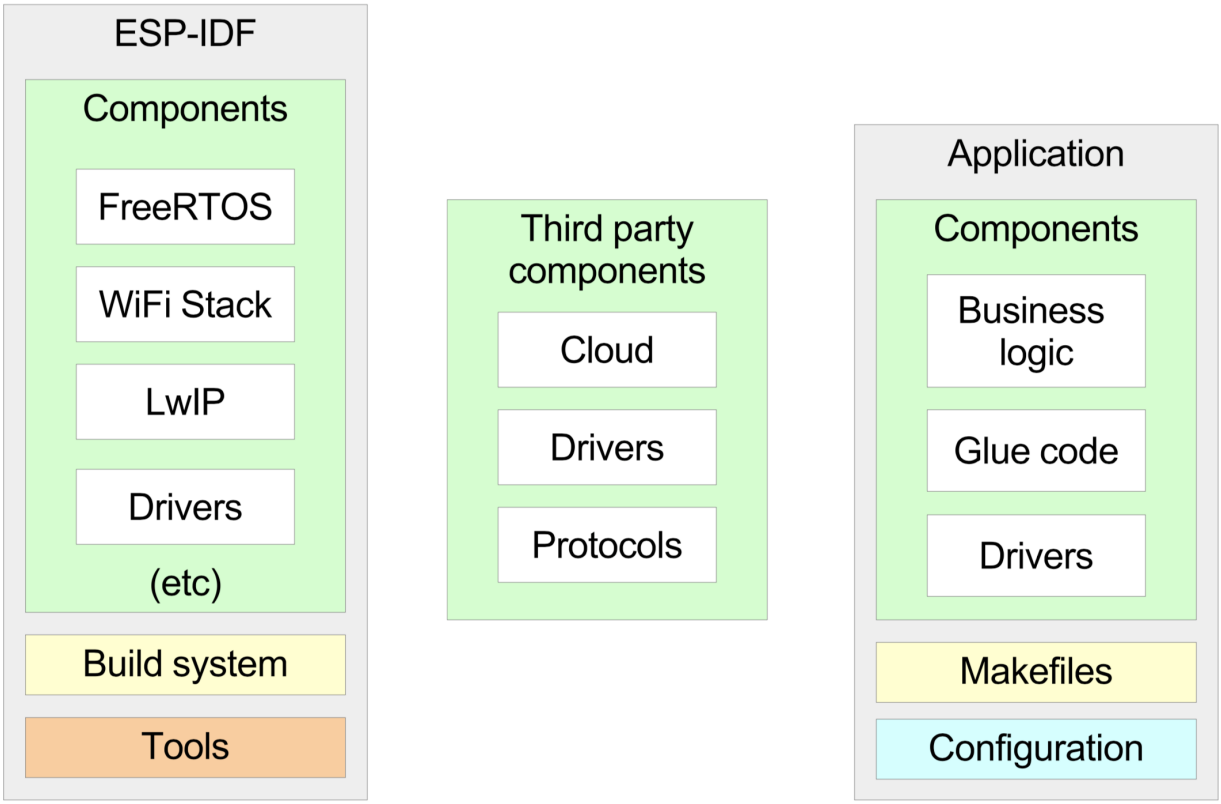

Fig. 5: ESP-IDF ????

ESP-IDF ?????????“??”??????????????????????Wi-Fi ???????? HTTP ?????????????“??”??????????????????????

??????

????????? ESP-IDF ?? *?????????????????????* SDK ???

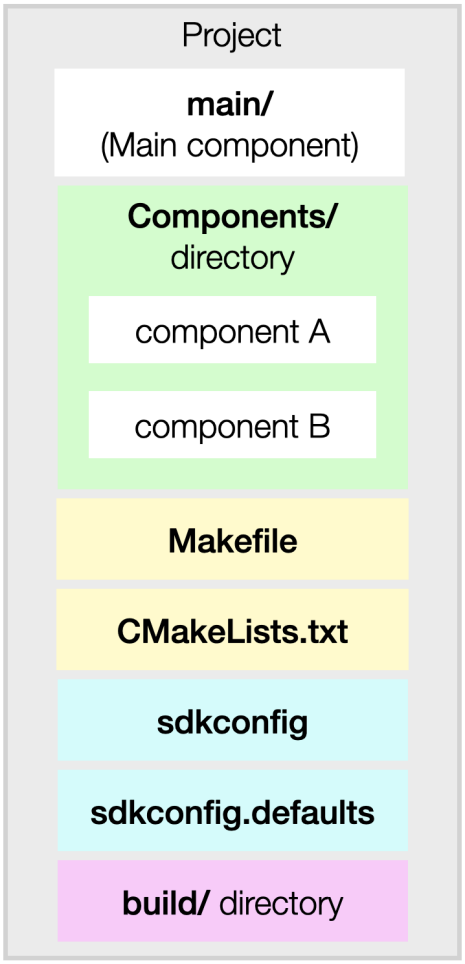

Fig. 6: ??????

- *CMakeLists.txt* ? *Makefile* ????????????????
- *components* ????????????????
- *main* ?????????????????????????????? **main** ???
- ????? *sdkconfig.defaults* ?????????? SDK ???

????????????:

- *build* ????????????
- *sdkconfig* ?????????????????????????????? menuconfig ?????????

Note: ???????????????????? ?????/?????

3.2.4 ?? ESP-Moonlight

ESP-Moonlight ????????? ESP-IDF ??? ESP-Moonlight ??

```
$ git clone --recursive https://github.com/espressif/esp-moonlight
```

???????????????????? ESP-IDF ???????????? ESP-Moonlight ???? ESP-IDF V4.0.1 ????????????????? esp-idf ? v4.0.1 ? tag?

```
$ cd esp-idf
$ git checkout -b v4.0.1 v4.0.1
$ git submodule update --recursive
```

Note: ????????????????????????????????????? IDF ?????? ESP-IDF ?????

???????? ESP-Jumpstart ?????????? *Hello World*???

```
$ cd esp-moonlight/1_hello_world
$ idf.py flash monitor
```

???

????????????????????????????????????

3.2.5 ??

?????????? Hello World ?????????? `examples/1_hello_world`?????????????????????:

```
void app_main()
{
    int i = 0;
    while (1) {
        printf("[%d] Hello world!\n", i);
        i++;
        vTaskDelay(5000 / portTICK_PERIOD_MS);
    }
}
```

????????????????

- `app_main()` ????????????FreeRTOS ???????????? ESP32 ??? `app_main()` ???????????????????? *main* ????????? idf ????????????????????????????????
- printf()?strlen()?time() ? C ????????????IDF ?? newlib C ????newlib ?????????? C ??????? stdio?stdlib????????????????/??????/?????? C ???????????? signal?locale?wchr ???????????????? printf() ??????????????
- `vTaskDelay()` ??? FreeRTOS ????????????????FreeRTOS ??? ESP32 ?????????FreeRTOS ??? vTaskDelay ???????? 5 ???? FreeRTOS API ?????????? FreeRTOS ???

3.2.6 [illegible]

[illegible]

3.3 [illegible]

[illegible] ESP32 [illegible] `examples/2_drivers` [illegible]

[illegible]

- [illegible] LED [illegible]
- [illegible] LED [illegible]
- [illegible]
- [illegible]

[illegible] `components` [illegible]

3.3.1 LED [illegible]

[illegible] LED [illegible] 6 [illegible] RGB LED [illegible]ESP32 [illegible] MOS [illegible] LED [illegible]

[illegible]

[illegible]	ESP32-S2-SOLO	ESP32-WROOM-32D
[illegible] (RED)	IO36	IO16
[illegible] (GREEN)	IO35	IO4
[illegible] (BLUE)	IO37	IO17

[illegible]

[illegible] LED [illegible]

```
/**< configure led driver */
led_rgb_config_t rgb_config = {0};
rgb_config.red_gpio_num   = BOARD_GPIO_LED_R;
rgb_config.green_gpio_num = BOARD_GPIO_LED_G;
rgb_config.blue_gpio_num  = BOARD_GPIO_LED_B;
rgb_config.red_ledc_ch    = LEDC_CHANNEL_0;
rgb_config.green_ledc_ch  = LEDC_CHANNEL_1;
rgb_config.blue_ledc_ch   = LEDC_CHANNEL_2;
rgb_config.speed_mode = LEDC_LOW_SPEED_MODE;
rgb_config.timer_sel  = LEDC_TIMER_0;
rgb_config.freq       = 20000;
rgb_config.resolution = LEDC_TIMER_8_BIT;
g_leds = led_rgb_create(&rgb_config);

if (!g_leds) {
    ESP_LOGE(TAG, "install LED driver failed");
}
```

LEDC [illegible] ESP32 [illegible] LED [illegible] PWM [illegible] `led_rgb_create()` [illegible] RGB [illegible] PWM [illegible] 20 KHz [illegible] 8 [illegible]

[illegible] LED [illegible]

```
/**< Write HSV values to LED */
ESP_ERROR_CHECK(g_leds->set_hsv(g_leds, a, b, c));

/**< Write RGB values to LED */
ESP_ERROR_CHECK(g_leds->set_rgb(g_leds, a, b, c));
```

[illegible] HSV [illegible] RGB [illegible] LED [illegible]

3.3.2 [illegible]

[illegible] ESP32-Moonlight [illegible] ESP32 [illegible] GPIO0[illegible]

[illegible]

[illegible]

```
static void button_press_cb(void *arg)
{
    if (g_led_mode) {
        g_led_mode = 0;
    } else {
        g_led_mode = 1;
    }

    ESP_LOGI(TAG, "Set the light mode to %d", g_led_mode);
}

static void configure_push_button(int gpio_num, void (*btn_cb)(void *))
{
    button_handle_t btn_handle = iot_button_create(gpio_num, 0);

    if (btn_handle) {
        iot_button_set_evt_cb(btn_handle, BUTTON_CB_TAP, button_press_cb, NULL);
    }
}
```

[illegible] `configure_push_button()` [illegible] button [illegible] GPIO [illegible] esp-timer [illegible] `button_press_cb()` [illegible] esp-timer [illegible]

3.3.3 [illegible]

[illegible]
ESP32 [illegible]

[illegible]ESP32 [illegible] IO
[illegible]

[illegible]

[illegible]

```
sensor_vibration_init(BOARD_GPIO_SENSOR_INT);
sensor_vibration_triggered_register(vibration_handle, NULL);
```

[illegible] `sensor_vibration_init()` [illegible] IO [illegible]

[illegible]

```
static void vibration_handle(void *arg)
{
    uint16_t h;
    uint8_t s;

    if (!g_led_mode) {
        return;
    }

    /**< Set a random color */
    h = esp_random() / 11930465;
    s = esp_random() / 42949673;
    s = s < 40 ? 40 : s;

    ESP_ERROR_CHECK(g_leds->set_hsv(g_leds, h, s, 100));
}
```

3.3.4 [illegible]

ESP32 [illegible] 12 [illegible] ADC[illegible] 18 [illegible] 1/2 [illegible] ESP32 ADC [illegible] ADC [illegible] 1100 mv [illegible] ADC [illegible]

[illegible]

[illegible]

```
esp_err_t sensor_adc_init(int32_t adc_channel)
{
    g_adc_ch_bat = adc_channel;
    /**< Check if Two Point or Vref are burned into eFuse */
    adc_check_efuse();

    /**< Configure ADC */
    adc1_config_width(ADC_WIDTH_BIT_12);
    adc1_config_channel_atten(g_adc_ch_bat, ADC_ATTEN_DB_12);
```

(continues on next page)

(continued from previous page)

```
    /**< Characterize ADC */
    g_adc_chars = calloc(1, sizeof(esp_adc_cal_characteristics_t));
    esp_adc_cal_value_t val_type = esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_
↪12, ADC_WIDTH_BIT_12, DEFAULT_VREF, g_adc_chars);
    print_char_val_type(val_type);

    xTaskCreatePinnedToCore(sensor_battery_task, "battery", 1024 * 2, NULL, 3, NULL,␣
↪1);

    return ESP_OK;
}
```

[illegible] `adc1_config_width()` [illegible] ADC [illegible] 12 [illegible] `adc1_config_channel_atten()` [illegible] 11 DB[illegible] 1/3.6 [illegible] ADC [illegible] [illegible] `xTaskCreatePinnedToCore()` [illegible] CPU1 [illegible]

3.3.5 [illegible]

[illegible]LED [illegible]ESP32 [illegible] monitor [illegible] ADC [illegible]

3.3.6 [illegible]

[illegible] [illegible] Wi-Fi [illegible]

3.4 Wi-Fi [illegible]

ESP32 [illegible] Wi-Fi [illegible] ESP32 [illegible] Wi-Fi [illegible] Wi-Fi [illegible] `examples/3_wifi_connection` [illegible]

3.4.1 Wi-Fi [illegible]

ESP32 [illegible] Wi-Fi [illegible]

- [illegible] STA [illegible] Wi-Fi [illegible] ESP32 [illegible] (AP)[illegible]
- AP [illegible] Soft-AP [illegible] ESP32[illegible]
- AP-STA [illegible]ESP32 [illegible]
- [illegible] IEEE802.11 Wi-Fi [illegible]

[illegible] STA [illegible] SSID [illegible] PASSWORD[illegible] ESP32 [illegible]

Note: ESP32 [illegible] 2.4G [illegible] Wi-Fi[illegible] 5G Wi-Fi[illegible]

3.4.2 ??

? `app_main.c` ??????????? Wi-Fi ???? sta ??????

```
esp_err_t wifi_init_sta(void)
{
    esp_err_t ret = ESP_OK;
    s_wifi_event_group = xEventGroupCreate();

    tcpip_adapter_init();

    ESP_ERROR_CHECK(esp_event_loop_create_default());

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

    ESP_ERROR_CHECK(esp_event_handler_register(WIFI_EVENT, ESP_EVENT_ANY_ID, &event_handler, NULL));
    ESP_ERROR_CHECK(esp_event_handler_register(IP_EVENT, IP_EVENT_STA_GOT_IP, &event_handler, NULL));

    wifi_config_t wifi_config = {
        .sta = {
            .ssid = EXAMPLE_ESP_WIFI_SSID,
            .password = EXAMPLE_ESP_WIFI_PASS
        },
    };
    ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_STA));
    ESP_ERROR_CHECK(esp_wifi_set_config(ESP_IF_WIFI_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) {
        ret = ESP_FAIL;
        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");
    }

    ESP_ERROR_CHECK(esp_event_handler_unregister(IP_EVENT, IP_EVENT_STA_GOT_IP, &event_handler));
```

(continues on next page)

(continued from previous page)

```
50     ESP_ERROR_CHECK(esp_event_handler_unregister(WIFI_EVENT, ESP_EVENT_ANY_ID, &event_handler));
51     vEventGroupDelete(s_wifi_event_group);
52     return ret;
53 }
```

- □□ xEventGroupCreate() □□□□□□□□□□□□□□□□□□□□□□□□ □□□□ Event Group API□
- □□ esp_event_handler_register() □ Wi-Fi □□□□□□□□□□□
- □□ tcpip_adapter_init() □□□□ TCP/IP □□□
- □□ WIFI_INIT_CONFIG_DEFAULT □□□□ Wi-Fi □□□□□□
- □□ esp_wifi_init() □esp_wifi_set_config() □esp_wifi_set_mode() □□□□ Wi-Fi □□□□□ station □□□□□□□ Wi-Fi □□□□□□□□ EXAMPLE_ESP_WIFI_SSID □ EXAMPLE_ESP_WIFI_PASS□
- 30 ~ 52 □□□□□□□□□□□□□□□□□□ WIFI_CONNECTED_BIT □ WIFI_FAIL_BIT □□□□□□□□□□□□□□□□□□□□□□□□□□□

Event loop □ idf □□ □□□□□□□□□□□ Wi-Fi □□□□□□□:

```
1  static void event_handler(void *arg, esp_event_base_t event_base,
2                            int32_t event_id, void *event_data)
3  {
4      static int32_t s_retry_num = 0;
5
6      if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_START) {
7          esp_wifi_connect();
8      } else if (event_base == WIFI_EVENT && event_id == WIFI_EVENT_STA_DISCONNECTED) {
9          if (s_retry_num < EXAMPLE_ESP_MAXIMUM_RETRY) {
10             esp_wifi_connect();
11             s_retry_num++;
12             ESP_LOGI(TAG, "retry to connect to the AP");
13         } else {
14             xEventGroupSetBits(s_wifi_event_group, WIFI_FAIL_BIT);
15         }
16
17         ESP_LOGI(TAG, "connect to the AP fail");
18     } else if (event_base == IP_EVENT && event_id == IP_EVENT_STA_GOT_IP) {
19         ip_event_got_ip_t *event = (ip_event_got_ip_t *) event_data;
20         ESP_LOGI(TAG, "got ip:" IPSTR, IP2STR(&event->ip_info.ip));
21         s_retry_num = 0;
22         xEventGroupSetBits(s_wifi_event_group, WIFI_CONNECTED_BIT);
23     }
24 }
```

- □□□ esp_wifi_start() □□□□□□ WIFI_EVENT_STA_START □□□□□□□ esp_wifi_connect() □□□□□□□□□□
- □□□□□□□□□ IP_EVENT_STA_GOT_IP □□□□□□□ event_data □□□ ip □□□□□□ xEventGroupSetBits() □□□□□□□□□ WIFI_CONNECTED_BIT □□
- □ Wi-Fi □□□□□□□□□ WIFI_EVENT_STA_DISCONNECTED □□□□□□□□ 9 ~ 15 □□□□□□□□□

□□□□□□

```c
void app_main(void)
{
    uint32_t hue = 0;
    /**< Initialize NVS */
    esp_err_t ret = nvs_flash_init();

    if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
        ESP_ERROR_CHECK(nvs_flash_erase());
        ret = nvs_flash_init();
    }

    ESP_ERROR_CHECK(ret);
    /**< install ws2812 driver */
    led_strip_config_t strip_config = LED_STRIP_DEFAULT_CONFIG(BOARD_GPIO_WS2812_DIN, BOARD_STRIP_LED_NUMBER, (led_strip_dev_t)RMT_CHANNEL_0);
    g_strip = led_strip_new_rmt_ws2812(&strip_config);

    if (!g_strip) {
        ESP_LOGE(TAG, "install WS2812 driver failed");
    }

    xTaskCreate(breath_light_task, "breath_light_task", 1024 * 3, NULL, 5, &g_breath_light_task_handle);
    ESP_LOGI(TAG, "Wait for connect");

    /**< Start the station */
    ret = wifi_init_sta();
    vTaskDelete(g_breath_light_task_handle);

    if (ESP_OK != ret) {
        /**< Set leds to red to indicate failure */
        ESP_ERROR_CHECK(g_strip->set_all_rgb(g_strip, 60, 0, 0));
        ESP_ERROR_CHECK(g_strip->refresh(g_strip, 10));
        ESP_LOGW(TAG, "Connect failed");
        return;
    }

    ESP_LOGI(TAG, "Color fade start");

    while (true) {

        /**< Write HSV values to strip driver */
        ESP_ERROR_CHECK(g_strip->set_all_hsv(g_strip, hue, 100, 100));
        /**< Flush to LEDs */
        ESP_ERROR_CHECK(g_strip->refresh(g_strip, 10));
        vTaskDelay(pdMS_TO_TICKS(30));
        hue++;

        if (hue > 360) {/**< The maximum value of hue in HSV color space is 360 */
            hue = 0;
        }
    }
}
```

- 5 ~ 10 [illegible] NVS (Non-volatile storage)[illegible]Wi-Fi [illegible] NVS [illegible]
- [illegible] 21 [illegible] `xTaskCreate()` [illegible] LED [illegible]
- `wifi_init_sta()` [illegible] Wi-Fi [illegible]

Note: [illegible] Wi-Fi [illegible] `idf.py menuconfig` [illegible] menuconfig [illegible] Example Configuration [illegible]

3.4.3 [illegible]

- [illegible] Wi-Fi [illegible]LED [illegible] ESP32 [illegible]
- [illegible] LED [illegible]
- [illegible]

3.4.4 [illegible]

[illegible]

3.5 SoftAP [illegible] Bluetooth Low Energy [illegible]

[illegible] Wi-Fi [illegible]SSID [illegible] PASSWORD[illegible] Wi-Fi[illegible] SoftAP [illegible]Bluetooth Low Energy [illegible]smartconfig [illegible] Wi-Fi [illegible]

[illegible] **[illegible]** [illegible] `examples/4_network_config` [illegible]

3.5.1 [illegible]

[illegible] Wi-Fi [illegible] Wi-Fi [illegible] Wi-Fi [illegible]

3.5.2 [illegible]

- **SoftAP [illegible]**[illegible]ESP32 [illegible] Wi-Fi [illegible] Wi-Fi [illegible] ESP32[illegible] ESP32 [illegible]
- **Bluetooth Low Energy [illegible]**[illegible]ESP32 [illegible] Bluetooth Low Energy [illegible] Bluetooth Low Energy [illegible] ESP32[illegible] Wi-Fi [illegible] SoftAP [illegible]
- **Smartconfig [illegible]**[illegible] UDP [illegible] Wi-Fi [illegible]ESP32 [illegible] Wi-Fi [illegible]
- **WEB [illegible]**[illegible] ESP32 [illegible]

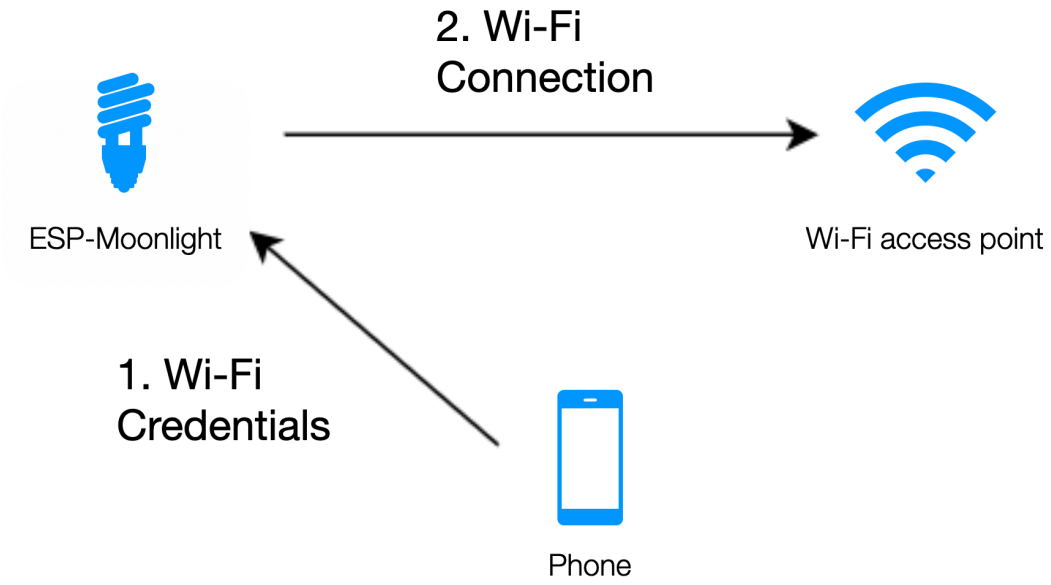

Fig. 7: ????

3.5.3 BluFi ??

??????????? Bluetooth Low Energy ???????ESP32 ????????????????——BluFi???????????????? Wi-Fi ???????????????? Wi-Fi ????????? ESP32???? ESP32 ????????????? AP ???? SoftAP??

Blufi ??????????????????????

- BluFi ??
- iOS ??
- Android ??

????? Android version ? iOS version ?????? APP ???????

Note: ???? IOS ? Android ? APP ??

[illegible]

BluFi [illegible]

```
/**< Initialize the BluFi */
blufi_init();

ESP_LOGI(TAG, "Wait for connect");
blufi_wait_connect();
```

`blufi_init()` [illegible] BluFi [illegible]`blufi_wait_connect()` [illegible] *components/blufi* [illegible] BluFi [illegible]

[illegible]

- `components/blufi/blufi.c`[illegible] BluFi [illegible]
- `components/blufi/blufi_security.c`[illegible] BluFi [illegible]
- `sdkconfig.default`[illegible]
- `partitions.csv`[illegible]ESP32 flash [illegible] factory [illegible] factory [illegible] 2 MB[illegible]

3.5.4 Wi-Fi [illegible]

[illegible] NVS [illegible]NVS(Non-volatile storage) [illegible]NVS [illegible] flash [illegible]

NVS [illegible] NVS [illegible] flash [illegible] NVS [illegible]

[illegible]Wi-Fi [illegible] NVS [illegible] Wi-Fi [illegible] *esp_wifi_set_storage(WIFI_STORAGE_RAM)* [illegible] Wi-Fi [illegible] RAM [illegible] NVS [illegible] Wi-Fi [illegible]

[illegible]

[illegible] *[illegible]* [illegible]

[illegible]

```
static void configure_push_button(int gpio_num)
{
    button_handle_t btn_handle = iot_button_create(gpio_num, 0);

    if (btn_handle) {
        iot_button_add_on_press_cb(btn_handle, 3, button_press_3sec_cb, NULL);
    }
}
```

[illegible] 3 [illegible] `button_press_3sec_cb()` [illegible]

[illegible]

```
static void button_press_3sec_cb(void *arg)
{
    ESP_LOGW(TAG, "Restore factory settings");
    nvs_flash_erase();
```

(continues on next page)

(continued from previous page)

```
    esp_restart();
}
```

□□□□□□□□□□ NVS □□□□□□□□□□□□□□□□□□ NVS □□□□□□□□□□□□□□□□□□□□□□□□□□

3.5.5 □□

□ ESP-MoonLight □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

Fig. 8: □□□□□□□□□

□□□□□□□□□□ https://github.com/EspressifApps/Moonlight □□□

- □□□□ 10 □□□□□□□□□□ Wi-Fi □□□□□□□□□□□ LED □□□□□□□□□□
- □□□□□□□□□□ LED □□□□□□□□□□□□□□□□□□□□□□□□□□□□ BluFi □□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□
  - □□□□ APP □□□□□□□□□□□□□□□□□□□□□□□□□□□□ ESP32 □□□□□□□□□□
  - □□□□□□□□□□□□□□□□□□□□□
- ESP32 □□□□□□□ Wi-Fi □□□□□□□□□□□ LED □□□□□□□□□□□□□□□□□□
- □□ Wi-Fi □□□□□□□□□□□□□□□
- □□□□□□□□□□□□□□□□□□□□□□□□□□□□□3 □□□□□□□□□□□□□□□□□□□□□□□□□□

3.5.6 [illegible]

[illegible] app [illegible] Wi-Fi [illegible]

3.6 [illegible]

[illegible]
[illegible] `examples/5_app_control` [illegible]

3.6.1 [illegible]

[illegible]

1. [illegible] API [illegible]

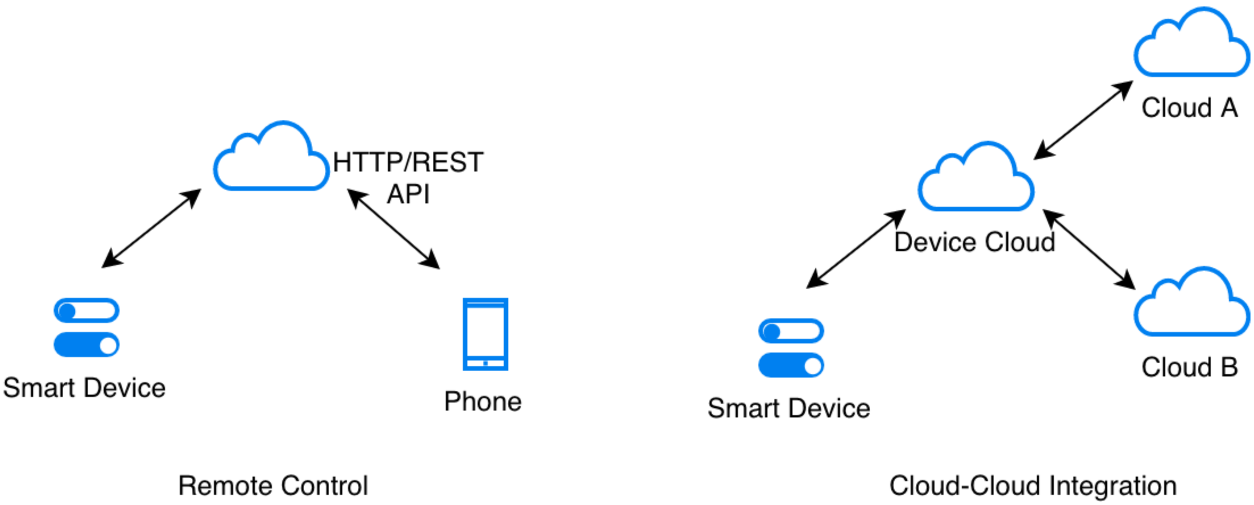

Fig. 9: [illegible]

[illegible]

2. [illegible]

[illegible] Wi-Fi [illegible]

3.6.2 [illegible]

[illegible] UDP [illegible] ESP32 [illegible] UDP Server[illegible] Client [illegible] UDP [illegible] JSON [illegible]ESP32 [illegible] LED [illegible]

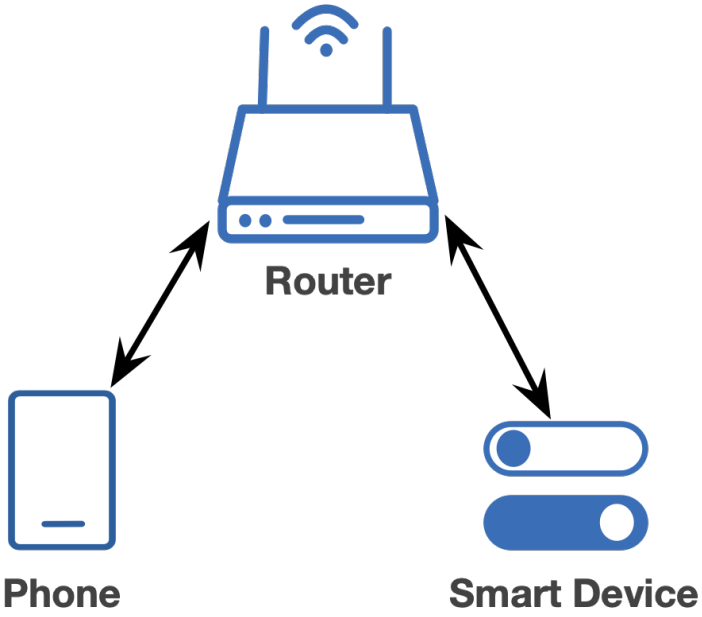
Router
Phone
Smart Device

Fig. 10:

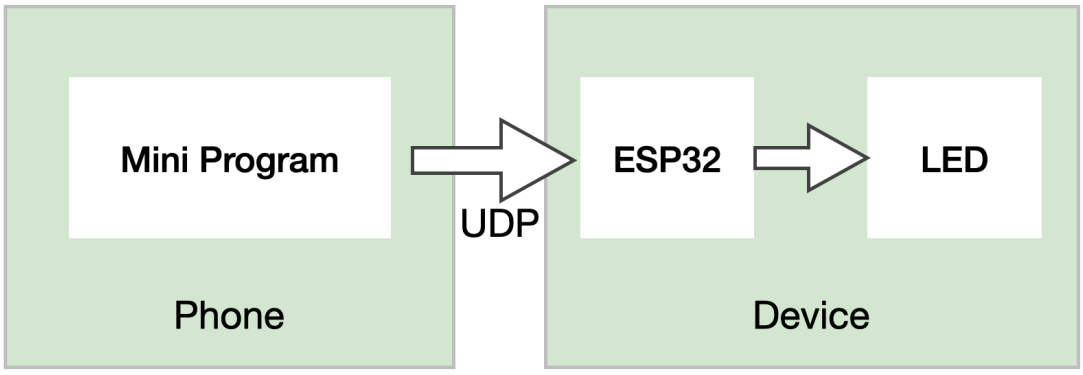
Mini Program
UDP
ESP32
LED
Phone
Device

Fig. 11:

????

???? JSON ????????:

```
{
    "led":{
        "red":255,
        "green":255,
        "blue":255
    }
}
```

?? red?green?blue ?????????????????????????? 0 ~ 255?

3.6.3 ??

????? UDP ?????????

```
if (addr_family == AF_INET) {
    struct sockaddr_in *dest_addr_ip4 = (struct sockaddr_in *)&dest_addr;
    dest_addr_ip4->sin_addr.s_addr = htonl(INADDR_ANY);
    dest_addr_ip4->sin_family = AF_INET;
    dest_addr_ip4->sin_port = htons(PORT);
    ip_protocol = IPPROTO_IP;
}

int sock = socket(addr_family, SOCK_DGRAM, ip_protocol);

if (sock < 0) {
    ESP_LOGE(TAG, "Unable to create socket: errno %d", errno);
    break;
}

ESP_LOGI(TAG, "Socket created");

int err = bind(sock, (struct sockaddr *)&dest_addr, sizeof(dest_addr));

if (err < 0) {
    ESP_LOGE(TAG, "Socket unable to bind: errno %d", errno);
}

ESP_LOGI(TAG, "Socket bound, port %d", PORT);

while (1) {

    ESP_LOGI(TAG, "Waiting for data");
    struct sockaddr_in6 source_addr; /**< Large enough for both IPv4 or IPv6 */
    socklen_t socklen = sizeof(source_addr);
    int len = recvfrom(sock, rx_buffer, sizeof(rx_buffer) - 1, 0, (struct sockaddr *)&
↪source_addr, &socklen);

    /**< Error occurred during receiving */
    if (len < 0) {
        ESP_LOGE(TAG, "recvfrom failed: errno %d", errno);
        break;
    }
```

(continues on next page)

(continued from previous page)

```
    /**< Data received */
    else {
        /**< Get the sender's ip address as string */
        if (source_addr.sin6_family == PF_INET) {
            inet_ntoa_r(((struct sockaddr_in *)&source_addr)->sin_addr.s_addr, addr_str, sizeof(addr_str) - 1);
        } else if (source_addr.sin6_family == PF_INET6) {
            inet6_ntoa_r(source_addr.sin6_addr, addr_str, sizeof(addr_str) - 1);
        }

        rx_buffer[len] = 0; /**< Null-terminate whatever we received and treat like a string... */
        ESP_LOGI(TAG, "Received %d bytes from %s:", len, addr_str);

        cJSON *root = cJSON_Parse(rx_buffer);

        if (!root) {
            printf("JSON format error:%s \r\n", cJSON_GetErrorPtr());
        } else {
            cJSON *item = cJSON_GetObjectItem(root, "led");
            int32_t red = cJSON_GetObjectItem(item, "red")->valueint;
            int32_t green = cJSON_GetObjectItem(item, "green")->valueint;
            int32_t blue = cJSON_GetObjectItem(item, "blue")->valueint;
            cJSON_Delete(root);

            if (red != g_red || green != g_green || blue != g_blue) {
                g_red = red;
                g_green = green;
                g_blue = blue;
                ESP_LOGI(TAG, "Light control: red = %d, green = %d, blue = %d", g_red, g_green, g_blue);
                ESP_ERROR_CHECK(g_leds->set_rgb(g_leds, g_red, g_green, g_blue));
            }
        }
    }
}
```

- 1 ? 26 ?? UDP ????????
- ????????? `recvfrom()` ?????
- ????????? `cJSON_Parse()` ?????? LED ?????
- ???????????????? LED ?

Note: ?????????????????????? UDP ???????????????

3.6.4 [illegible]

[illegible]

3.7 [illegible]

[illegible]
OTA (over-the-air) [illegible]

[illegible] ESP32 [illegible] URL [illegible] `examples/6_ota` [illegible]

3.7.1 Flash [illegible]

[illegible] ESP32 [illegible] flash [illegible]

[illegible] ESP32 [illegible] flash [illegible]

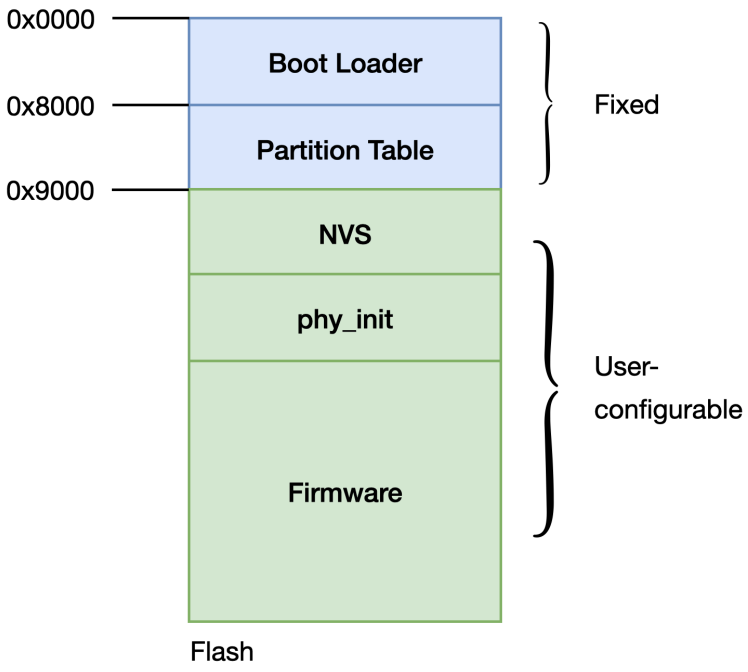

Fig. 12: Flash [illegible]

[illegible]flash [illegible] 0x9000 [illegible] Bootloader[illegible] flash [illegible] [illegible] [illegible]

?????

?? OTA ??????? OTA Data ????????????????????????????????????? CSV ???Comma Separated Values?????????

???????????????????

```
# Name,   Type, SubType, Offset,   Size, Flags
nvs,      data, nvs,     ,        0x4000,
otadata,  data, ota,     ,        0x2000,
phy_init, data, phy,     ,        0x1000,
ota_0,    app,  ota_0,   ,        1600K,
ota_1,    app,  ota_1,   ,        1600K,
```

????????????? 1600 KB ?????????????????????????

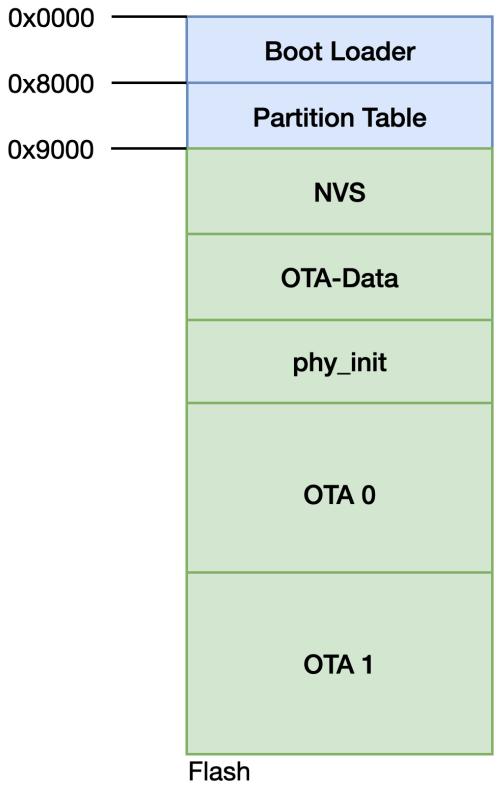

Fig. 13: OTA Flash ??

????????????? menuconfig ??????????????????????? `Partition Table ---> Partition Table` ??? `Custom partition table CSV`????????????? ???????? `sdkconfig.defaults` ??????????????????????????

3.7.2 ??????

??OTA Data ????????????????

OTA ??????????????????????

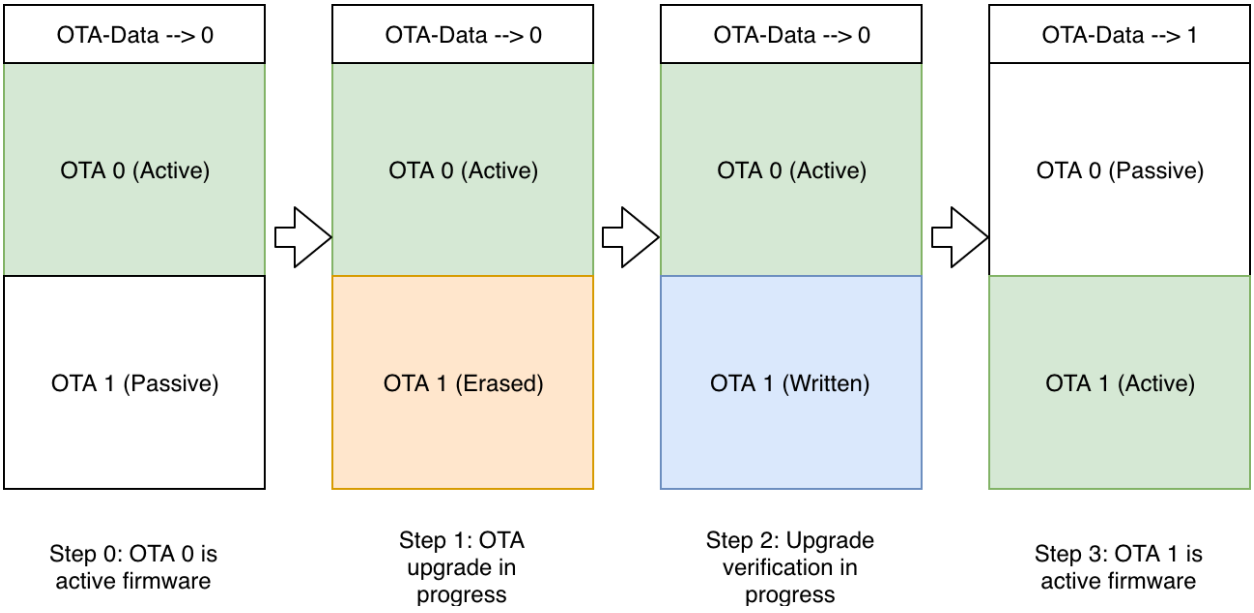

Fig. 14: ??????

- ?? 0?OTA 0 ???????????? OTA Data ???
- ?? 1?????????????????????????????? OTA 1 ???
- ?? 2??????????????????
- ?? 3??????????OTA Data ?????????? OTA 1 ????????????????????????????

3.7.3 ??

??????????????????????

```
esp_http_client_config_t config = {
    .url = url,
    .cert_pem = (char *)server_cert_pem_start,
    .event_handler = _http_event_handler,
};

esp_err_t ret = esp_https_ota(&config);
if (ret == ESP_OK) {
    esp_restart();
} else {
    ESP_LOGE(TAG, "Firmware upgrade failed");
}
return ret;
```

- ?? `esp_http_client_config_t` ?? OTA ?????????? URL????????? CA ????????????????
- ???? `esp_https_ota()` API ????????????????????

???? URL

??????????????? URL ???? menuconfig ?? `Example Configuration ---> firmware upgrade url endpoint` ?????

??????????? http server?????? IP ??????????

3.7.4 ??

???????????????????

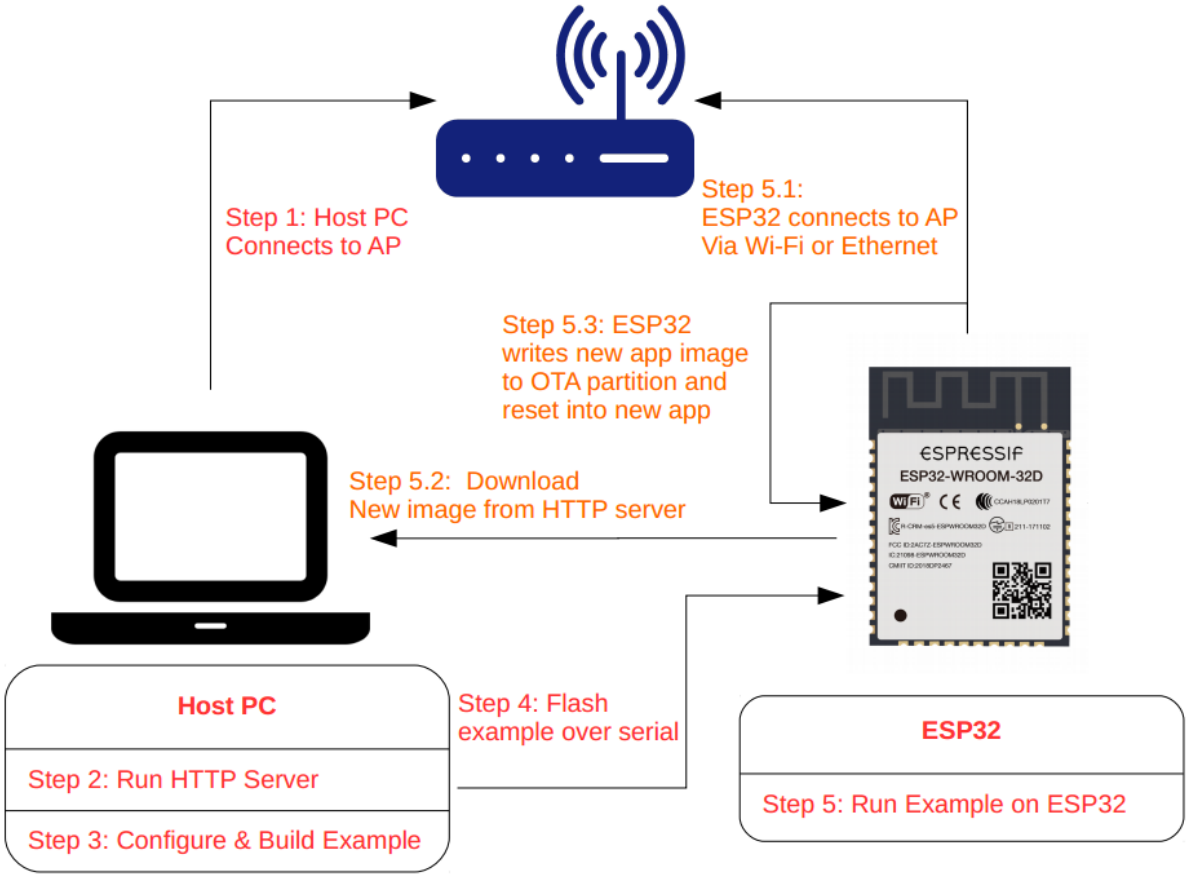

Fig. 15: OTA ????

[illegible] HTTPS Server

- [illegible] `cd https_server`[illegible]
- [illegible]`openssl req -x509 -newkey rsa:2048 -keyout ca_key.pem -out ca_cert.pem -days 365 -nodes`[illegible] KEY[illegible] [illegible][illegible] *.pem* [illegible]
- [illegible] HTTPS server[illegible]`openssl s_server -WWW -key ca_key.pem -cert ca_cert.pem -port 8070`[illegible]
- [illegible] *2_drivers* [illegible] *moonlight.bin*[illegible] *firmware upgrade url endpoint*[illegible]

Note: [illegible] *8070* [illegible]

Note: Windows [illegible] *openssl* [illegible] *winpty*[illegible]

- `winpty openssl req -x509 -newkey rsa:2048 -keyout ca_key.pem -out ca_cert.pem -days 365 -nodes`
- `winpty openssl s_server -WWW -key ca_key.pem -cert ca_cert.pem -port 8070`

[illegible]

[illegible] `idf.py flash monitor` [illegible] [illegible] `ca_cert.pem` [illegible]

[illegible]

[illegible] OTA [illegible]

[illegible] HTTPS Server [illegible]

```
ACCEPT
FILE:moonlight.bin
```

3.7.5 [illegible]

[illegible] flash [illegible]

3.8 ??????

?????????????????????????????????? ESP-SR ??? LED ????????????????? `examples/7_recognition` ???

???????????

- ??????????
- ???????
- ????? LED ???????????

3.8.1 ESP-SR ??

??????????

- ?????????WakeNet
- ??????????MultiNet
- ???????????????????Mic-Array Speech Enhancement??? MASE????????Acoustic Echo Cancellation??? AEC???????????Voice Activity Detection??? VAD???????????Automatic Gain Control??? AGC????????Noise Suppression??? NS?

????? WakeNet ????????? “Hi???” ?????????????????????????????? MultiNet ??????????????????????????

???????? ESP-SR?

3.8.2 ?????

?????????????????????????????????? I2S ?????? MEMS ??????? ESP32 ? I2S ??????

????????

Pin Name	ESP32-S3-WROOM-1	ESP32-S2-SOLO	ESP32-WROOM-32D
DMIC_I2S_SCK	IO39	IO15	IO32
DMIC_I2S_WS	IO38	IO16	IO32
DMIC_I2S_SDO	IO40	IO17	IO25

??????????? Audio Codec ???????????

3.8.3 ?????

???????????????????? I2S ????????????????

Pin Name	ESP32-S3-WROOM-1	ESP32-S2-SOLO	ESP32-WROOM-32D
DSPEAKER_I2S_BCK	IO48	×	×
DSPEAKER_I2S_WS	IO45	×	×
DSPEAKER_I2S_SDO	IO47	×	×

Note: *ESP32S3* ???????????? IDF ?? v5.0 ????

3.8.4 ??

??????????????????????

```
i2s_read(1, buffer, size * 2 * sizeof(int), &read_len, portMAX_DELAY);

for (int x = 0; x < size * 2 / 4; x++) {
   int s1 = ((buffer[x * 4] + buffer[x * 4 + 1]) >> 13) & 0x0000FFFF;
   int s2 = ((buffer[x * 4 + 2] + buffer[x * 4 + 3]) << 3) & 0xFFFF0000;
   buffer[x] = s1 | s2;
}

if (enable_wn) {
   wakenet_state_t r = wakenet->detect(model_wn_data, (int16_t *)buffer);

   if (r == WAKENET_DETECTED) {
         ESP_LOGI(TAG, "%s DETECTED", wakenet->get_word_name(model_wn_data, r));

         if (NULL != g_sr_callback_func[SR_CB_TYPE_WAKE].fn) {
            g_sr_callback_func[SR_CB_TYPE_WAKE].fn(g_sr_callback_func[SR_CB_TYPE_
↪WAKE].args);
         }

         enable_wn = false;
   }
} else {
   esp_mn_state_t mn_state = multinet->detect(model_mn_data, (int16_t *)buffer);
   if (mn_state == ESP_MN_STATE_DETECTED) {
         esp_mn_results_t *mn_result = multinet->get_results(model_mn_data);
         ESP_LOGI(TAG, "MN test successfully, Commands ID: %d", mn_result->phrase_
↪id[0]);
         int command_id = mn_result->phrase_id[0];

         if (NULL != g_sr_callback_func[SR_CB_TYPE_CMD].fn) {
            if (NULL != g_sr_callback_func[SR_CB_TYPE_CMD].args) {
               g_sr_callback_func[SR_CB_TYPE_CMD].fn(g_sr_callback_func[SR_CB_TYPE_
↪CMD].args);
            } else {
               g_sr_callback_func[SR_CB_TYPE_CMD].fn((void *)command_id);
            }
         }

   } else if (mn_state == ESP_MN_STATE_TIMEOUT) {
         esp_mn_results_t *mn_result = multinet->get_results(model_mn_data);
         ESP_LOGI(TAG, "timeout, string:%s\n", mn_result->string);

         if (NULL != g_sr_callback_func[SR_CB_TYPE_CMD_EXIT].fn) {
            g_sr_callback_func[SR_CB_TYPE_CMD_EXIT].fn(g_sr_callback_func[SR_CB_TYPE_
↪CMD_EXIT].args);
         }

         enable_wn = true;
   } else {
         continue;
   }
}
```

- ???? `i2s_read()` ??????????????????????????????

- ?? `enable_wn` ??????????????????
- ?? `detect()` ??????????????????????
- ????????????????????????

3.8.5 ?????

`app_speech_rec.c` ??????? 11 ?????????

```
char *commands[] = {
   "da kai dian deng",
   "kai deng",
   "da kai xiao ye deng",
   "guan bi dian deng",
   "guan deng",
   "guan bi xiao ye deng",
   "huan yi ge yan se",
   "liang yi dian",
   "zeng da liang du",
   "an yi dian",
   "jian xiao liang du",
};

esp_mn_commands_clear();
for (int i = 0; i<COMMANDS_NUM ; i++) {
   esp_mn_commands_add(i, commands[i]);
}
esp_mn_commands_update();
```

???? *esp_mn_commands_add* ?????????????????????????????????? MultiNet ??? ????????????????????????????????

3.8.6 ??

- ???????“Hi????”????????? ESP32 ????????????? LED ???????????
- ???????“?????”?“?????”?“?????”????????????????????????????????
- ????????????????????????
- ??