Phân tích vòng lặp while (1/2)

```
    if(isreceive != 1){
        items_rx_ir = (rmt_item32_t*)xRingbufferReceive(ring_buffer_ir,
&number_rx_ir, 100);
        if (items_rx_ir){
            if(number_rx_ir > 100){
                if((isteaching == 3) || (isteaching == 5)){
                    if(isteaching == 3){
                        isok = add_raw_key(items_rx_ir, number_rx_ir/4,
tmprequest[0], tmprequest[1], tmprequest[2]);
                        ble_mesh_send(gateway, 0xE50211, 5, SEND_TEACHING_CMD,
tmprequest[0], tmprequest[1], tmprequest[2], isok, 0, 0, 0);
                        voicerequest[1] = 253;
                        uart_write_bytes(UART_NUM_0, voicerequest, 3);
                        gpio_set_level(LED, 0);
                        isreceive = check_params_remote();
                        if(isreceive != 1) rmt_rx_start(RMT_CHANNEL_0, true);
                        else rmt_rx_stop(RMT_CHANNEL_0);
                        isteaching = 1;
                    }
                    else{
                        fill_recv_ir(items_rx_ir, number_rx_ir/4);
                        if(irrecv.decode(&results)){
                            if(find_ac_teaching()){
                                voicerequest[1] = 253;
                                uart_write_bytes(UART_NUM_0, voicerequest, 3);
                                gpio_set_level(LED, 0);
                                isreceive = check_params_remote();
                                if(isreceive != 1) rmt_rx_start(RMT_CHANNEL_0,
true);
                                else rmt_rx_stop(RMT_CHANNEL_0);
                                isteaching = 1;
                            }
                        }
                        delete data_ir_recv;
                    }
                }
                else{
                    if(isdisplay == 2){
                        if(isdecode == 2){
                            if(decode_custom_ir(items_rx_ir, number_rx_ir/4))
uart_send_custom_ir(custom_ir);
                        }
                        else {
                            uart_send_raw_ir(items_rx_ir, number_rx_ir/4);
                        }
                        voicerequest[1] = 255;
                        uart_write_bytes(UART_NUM_0, voicerequest, 3);
                        gpio_set_level(LED, 1);
                        vTaskDelay(100);
                        gpio_set_level(LED, 0);
```

```
                    }
                    else{
                        statusdecode = decode_custom_ir(items_rx_ir,
number_rx_ir/4);
                        if(isreceive & 0x02){
                            if(find_memory_remote(items_rx_ir, number_rx_ir/4,
statusdecode)){
                                voicerequest[1] = 255;
                                uart_write_bytes(UART_NUM_0, voicerequest, 3);
                                gpio_set_level(LED, 1);
                                vTaskDelay(100);
                                gpio_set_level(LED, 0);
                            }
                        }
                        if(isreceive & 0x04){
                            fill_recv_ir(items_rx_ir, number_rx_ir/4);
                            if(irrecv.decode(&results)){
                                if(find_ac_remote()){
                                    voicerequest[1] = 255;
                                    uart_write_bytes(UART_NUM_0, voicerequest, 3);
                                    gpio_set_level(LED, 1);
                                    vTaskDelay(100);
                                    gpio_set_level(LED, 0);
                                }
                            }
                            delete data_ir_recv;
                            if(statusdecode){
                                if(find_ac_custom_remote()){
                                    voicerequest[1] = 255;
                                    uart_write_bytes(UART_NUM_0, voicerequest, 3);
                                    gpio_set_level(LED, 1);
                                    vTaskDelay(100);
                                    gpio_set_level(LED, 0);
                                }
                            }
                        }
                    }
                }
            }
            irrecv.resume();
            vRingbufferReturnItem(ring_buffer_ir, items_rx_ir);
        }
    }
    else {
        vTaskDelay(100);
    }
```

Đoạn mã cung cấp là một phần của một vòng lặp trong hàm `while` trong một tác vụ (task) của ESP32. Mục tiêu là kiểm tra dữ liệu IR (hồng ngoại) nhận được từ một vòng đệm (`ring_buffer_ir`) và xử lý các tình huống khác nhau dựa trên giá trị nhận được. Sau đây là một phân tích chi tiết và một số cải tiến cho mã:

Phân tích mã:

1. **Kiểm tra trạng thái `isreceive`:**
   - Đoạn mã bắt đầu với kiểm tra biến `isreceive`. Nếu `isreceive` không bằng 1 (có thể đây là trạng thái "đã nhận thành công"), nó sẽ tiếp tục xử lý các dữ liệu nhận được từ vòng đệm.

```
if (isreceive != 1) {
    items_rx_ir = (rmt_item32_t*)xRingbufferReceive(ring_buffer_ir,
&number_rx_ir, 100);
    if (items_rx_ir) {
```

2. **Đọc dữ liệu từ vòng đệm:**
   - Sử dụng `xRingbufferReceive` để nhận dữ liệu từ vòng đệm `ring_buffer_ir` và lưu trữ vào `items_rx_ir`.
   - Nếu nhận thành công (kết quả trả về là không null), tiếp tục xử lý dữ liệu.

```
if (number_rx_ir > 100) {
```

3. **Kiểm tra điều kiện `isteaching`:**
   - Kiểm tra các giá trị của `isteaching` để xử lý các hành động khác nhau. Có hai trường hợp: `isteaching == 3` và `isteaching == 5`.

Nếu `isteaching == 3`:

- Gọi hàm `add_raw_key` để thêm một khóa IR thô và gửi qua BLE Mesh.
- Tiến hành các thao tác như gửi yêu cầu qua UART và điều khiển trạng thái của LED.

Nếu `isteaching == 5`:

- Chuyển đến xử lý mã IR đã nhận bằng cách sử dụng `fill_recv_ir`.
- Tiến hành giải mã mã IR với `irrecv.decode`.
- Kiểm tra và tìm kiếm mã IR qua `find_ac_teaching()`, sau đó xử lý tiếp.

```
if (isteaching == 3) {
    isok = add_raw_key(items_rx_ir, number_rx_ir / 4, tmprequest[0],
tmprequest[1], tmprequest[2]);
    ble_mesh_send(gateway, 0xE50211, 5, SEND_TEACHING_CMD, tmprequest[0],
tmprequest[1], tmprequest[2], isok, 0, 0, 0);
    voicerequest[1] = 253;
    uart_write_bytes(UART_NUM_0, voicerequest, 3);
    gpio_set_level(LED, 0);
    isreceive = check_params_remote();
    if (isreceive != 1) rmt_rx_start(RMT_CHANNEL_0, true);
    else rmt_rx_stop(RMT_CHANNEL_0);
    isteaching = 1;
}
```

4. **Các tình huống khác khi không phải `isteaching == 3` hoặc `isteaching == 5`:**
   - **Hiển thị IR đã nhận:** Kiểm tra và giải mã các mã IR nhận được nếu chế độ hiển thị là `isdisplay == 2`.
   - **Gửi IR raw hoặc custom IR:** Tùy thuộc vào giá trị `isdecode`, sẽ gửi IR raw hoặc custom IR qua UART.
   - Điều khiển trạng thái LED và thực hiện một số thao tác với mã IR đã giải mã.

```
if (isdisplay == 2) {
    if (isdecode == 2) {
        if (decode_custom_ir(items_rx_ir, number_rx_ir / 4))
uart_send_custom_ir(custom_ir);
    } else uart_send_raw_ir(items_rx_ir, number_rx_ir / 4);
    voicerequest[1] = 255;
    uart_write_bytes(UART_NUM_0, voicerequest, 3);
    gpio_set_level(LED, 1);
    vTaskDelay(100);
    gpio_set_level(LED, 0);
}
```

5. **Kiểm tra và xử lý các trường hợp `isreceive & 0x02` và `isreceive & 0x04`:**
   - Nếu điều kiện này đúng, hệ thống sẽ thực hiện các hành động như tìm mã IR đã lưu trong bộ nhớ, giải mã và kiểm tra với các mã IR đã cài đặt.
   - Gửi yêu cầu qua UART nếu mã IR hợp lệ và điều khiển LED.

```
if (isreceive & 0x02) {
    if (find_memory_remote(items_rx_ir, number_rx_ir / 4, statusdecode)) {
        voicerequest[1] = 255;
        uart_write_bytes(UART_NUM_0, voicerequest, 3);
        gpio_set_level(LED, 1);
        vTaskDelay(100);
        gpio_set_level(LED, 0);
    }
}
```

6. **Giải mã và gửi phản hồi nếu `isreceive & 0x04`:**
   - Nếu `isreceive` có bit 0x04, dữ liệu sẽ được giải mã, kiểm tra, và phản hồi lại qua UART.

```
if (isreceive & 0x04) {
    fill_recv_ir(items_rx_ir, number_rx_ir / 4);
    if (irrecv.decode(&results)) {
        if (find_ac_remote()) {
            voicerequest[1] = 255;
            uart_write_bytes(UART_NUM_0, voicerequest, 3);
            gpio_set_level(LED, 1);
            vTaskDelay(100);
```

```
            gpio_set_level(LED, 0);
        }
    }
    delete data_ir_recv;
    if (statusdecode) {
        if (find_ac_custom_remote()) {
            voicerequest[1] = 255;
            uart_write_bytes(UART_NUM_0, voicerequest, 3);
            gpio_set_level(LED, 1);
            vTaskDelay(100);
            gpio_set_level(LED, 0);
        }
    }
}
```

7. **Cuối cùng, hoàn tất quá trình nhận và giải mã IR:**
    - Sau khi xử lý xong, các mục IR đã nhận được sẽ được trả lại cho vòng đệm.

```
irrecv.resume();
vRingbufferReturnItem(ring_buffer_ir, items_rx_ir);
```

Đề xuất cải tiến:

- **Kiểm tra `isreceive` mỗi lần:** Việc kiểm tra `isreceive != 1` trước khi tiếp tục có thể giúp tránh tình trạng dữ liệu không hợp lệ. Có thể điều chỉnh trạng thái `isreceive` trong các tình huống khác nhau để tránh việc lặp lại vô tận.
- **Tối ưu vòng lặp:** Trong một số tình huống, bạn có thể muốn thêm một `vTaskDelay` ở đầu hoặc giữa các phần mã để tránh tác động lớn đến hiệu suất của hệ thống và giảm tải cho bộ xử lý.
- **Quản lý bộ nhớ:** Lưu ý rằng việc sử dụng `delete` cho `data_ir_recv` trong trường hợp nhất định có thể cần thiết để giải phóng bộ nhớ, nhưng cần phải chắc chắn rằng bộ nhớ được cấp phát đúng cách trong những trường hợp này.
- **Tối ưu hóa mã IR:** Cần chú ý rằng các thao tác liên quan đến mã IR như `irrecv.decode` và `find_*` có thể cần được kiểm tra kỹ lưỡng để xử lý các tình huống mà mã IR không hợp lệ hoặc bị hỏng.

Phân tích vòng lặp while (2/2)

```
    uart_get_buffered_data_len(UART_NUM_0, &length_buffer_uart);

    if(length_buffer_uart >= 8){
        uart_read_bytes(UART_NUM_0, (void*)tmpqueue, 8, 100);
        if(lockdevice)
        {
            if(tmpqueue[0] == CONFIG_MEMORY_CMD) set_queue(tmpqueue);
        }
        else set_queue(tmpqueue);
```

```
    }

    get_queue();

    switch(request[0])
    {
        case CONFIG_MEMORY_CMD:
            gpio_set_level(LED, 1);
            gpio_set_level(PIN_RES, 0);
            if(request[1] == 1) clear_flash(request[2], request[3]);
            else
            {
                if(request[1] == 2) save_flash();
                else
                {
                    if(request[1] == 3) load_flash(request[2], request[3],
request[4]);
                }
            }
            uart_write_bytes(UART_NUM_0, "!", 1);
            vTaskDelay(100);
            gpio_set_level(PIN_RES, 1);
            gpio_set_level(LED, 0);
            request[0] = NONE_CMD;
            break;
        case RECV_KEY_CMD:
            if(isreceive != 1) rmt_rx_stop(RMT_CHANNEL_0);
            gpio_set_level(LED, 1);
            isok = send_memory_ir(request[1], request[2], request[3], request[4]);
            if(isok)
            {
                if(check_list_remote(request[1], request[2], request[3]))
ble_mesh_send(gateway, 0xE50211, 8, SEND_KEY_CMD, 8, request[1], request[2],
request[3], request[4], 2, 17);
                else ble_mesh_send(gateway, 0xE50211, 8, RECV_KEY_CMD, 1, 0, 0, 0,
0, 0, 0);
                send_buffer_ir();
            }
            else ble_mesh_send(gateway, 0xE50211, 8, RECV_KEY_CMD, 0, 0, 0, 0, 0,
0, 0);
            voicerequest[1] = 255;
            uart_write_bytes(UART_NUM_0, voicerequest, 3);
            vTaskDelay(100);
            gpio_set_level(LED, 0);
            if(isreceive != 1) rmt_rx_start(RMT_CHANNEL_0, true);
            irrecv.resume();
            request[0] = NONE_CMD;
            break;
        case RECV_AC_CMD:
            gpio_set_level(LED, 1);
            if(isreceive != 1) rmt_rx_stop(RMT_CHANNEL_0);
            isok = send_ac_ir(request[1], request[2], request[3], request[4],
request[5], request[6], request[7]);
            if(isok)
```

```
            {
                if(check_list_remote(request[1], request[2], request[3]))
ble_mesh_send(gateway, 0xE50211, 8, SEND_KEY_CMD, 8, request[1], request[2],
request[3], request[4], request[5], ((request[6]<<4)&0xF0)|(request[7]&0x0F));
                else ble_mesh_send(gateway, 0xE50211, 8, RECV_AC_CMD, 1, 0, 0, 0,
0, 0, 0);
                send_buffer_ir();
            }
            else ble_mesh_send(gateway, 0xE50211, 8, RECV_AC_CMD, 0, 0, 0, 0, 0,
0, 0);
            voicerequest[1] = 255;
            uart_write_bytes(UART_NUM_0, voicerequest, 3);
            vTaskDelay(100);
            gpio_set_level(LED, 0);
            if(isreceive != 1) rmt_rx_start(RMT_CHANNEL_0, true);
            irrecv.resume();
            request[0] = NONE_CMD;
            break;
        case RECV_PARAM_CMD:
            gpio_set_level(LED, 1);
            isok = 0;
            if(request[1] == 1) isok = delete_params_remote(request[3],
request[4], request[5]);
            else
            {
                if(request[1] == 2) isok = add_params_remote(request[2],
request[3], request[4], request[5]);
                else
                {
                    if(request[1] == 3) isok = delete_all_params_remote();
                }
            }
            ble_mesh_send(gateway, 0xE50211, 2, RECV_PARAM_CMD, isok, 0, 0, 0,
0, 0);
            save_memory_system();
            isreceive = check_params_remote();
            if(isreceive != 1) rmt_rx_start(RMT_CHANNEL_0, true);
            else rmt_rx_stop(RMT_CHANNEL_0);
            voicerequest[1] = 255;
            uart_write_bytes(UART_NUM_0, voicerequest, 3);
            vTaskDelay(100);
            gpio_set_level(LED, 0);
            request[0] = NONE_CMD;
            break;
        case RECV_MODE_CMD:
            gpio_set_level(LED, 1);
            isdecode = request[1]+1;
            isdisplay = request[2]+1;
            isteaching = request[3]+1;
            isok = 0;
            if(isteaching == 2)
            {
                isok = delete_raw_key(request[4], request[5], request[6]);
                isteaching = 1;
```

```
            }
            else
            {
                if(isteaching == 4)
                {
                    clear_flash(request[4], request[5]);
                    isok = 1;
                    isteaching = 1;
                }
                else
                {
                    if(isteaching == 3)
                    {
                        if((request[4]+256*request[5] > 1) &&
(request[4]+256*request[5] < 21) && request[6])
                        {
                            voicerequest[1] = 252;
                            uart_write_bytes(UART_NUM_0, voicerequest, 3);
                            tmprequest[0] = request[4];
                            tmprequest[1] = request[5];
                            tmprequest[2] = request[6];
                            timewaittingteaching = 0;
                            isok = 1;
                        }
                        else isteaching = 1;
                    }
                    else
                    {
                        if(isteaching == 5)
                        {
                            voicerequest[1] = 252;
                            uart_write_bytes(UART_NUM_0, voicerequest, 3);
                            tmprequest[0] = request[4];
                            tmprequest[1] = request[5];
                            tmprequest[2] = request[6];
                            timewaittingteaching = 0;
                            isok = 1;
                        }
                        else isteaching = 1;
                    }
                }
            }
            ble_mesh_send(gateway, 0xE50211, 2, RECV_MODE_CMD, isok, 0, 0, 0, 0,
0, 0);
            if((isdisplay == 2) || (isteaching == 3) || (isteaching == 5))
            {
                rmt_rx_start(RMT_CHANNEL_0, true);
                isreceive = 2;
            }
            if(isteaching == 1)
            {
                voicerequest[1] = 255;
                uart_write_bytes(UART_NUM_0, voicerequest, 3);
                vTaskDelay(100);
```

```
                gpio_set_level(LED, 0);
            }
            request[0] = NONE_CMD;
            break;
        case RECV_SCRIPT_CMD:
            gpio_set_level(LED, 1);
            isok = 0;
            if((request[3] & 0xC0) == 0x40) isok = delete_script(request[1],
request[2], request[3], request[4], request[5], request[6]);
            else
            {
                if((request[3] & 0xC0) == 0xC0) isok =
delete_all_script(request[1], request[2]);
                else
                {
                    if((request[3] & 0xC0) == 0x80) isok = add_script(request[1],
request[2],request[3], request[4], request[5], request[6],request[7]);
                }
            }
            ble_mesh_send(gateway, 0x8245, 8, 0, request[1], request[2], 0, 0, 0,
0, 0);
            voicerequest[1] = 255;
            uart_write_bytes(UART_NUM_0, voicerequest, 3);
            vTaskDelay(100);
            gpio_set_level(LED, 0);
            request[0] = NONE_CMD;
            break;
        case RECV_OTA_CMD:
            gpio_set_level(LED, 1);
            isok = save_ota(request[1],
65536*request[2]+256*request[3]+request[4],
65536*request[5]+256*request[6]+request[7]);
            ble_mesh_send(gateway, 0xE50211, 2, RECV_OTA_CMD, isok, 0, 0, 0, 0, 0,
0);
            vTaskDelay(100);
            gpio_set_level(LED, 0);
            if(isok) esp_restart();
            request[0] = NONE_CMD;
            break;
        case RECV_VOICE_CMD:
            if(request[1]+256*request[2] == wakeupvoice)
            {
                iswaittingvoice = 2;
                gpio_set_level(LED, 1);
                voicerequest[1] = 100;
                uart_write_bytes(UART_NUM_0, voicerequest, 3);
            }
            else
            {
                if(iswaittingvoice == 2)
                {
                    if((request[1]+256*request[2] > 0) &&
(request[1]+256*request[2] < 255))
                    {
```

```
                        if(isreceive != 1) rmt_rx_stop(RMT_CHANNEL_0);
                        voicerequest[1] = request[1];
                        uart_write_bytes(UART_NUM_0, voicerequest, 3);
                        if((request[1]+256*request[2] > 0) &&
(request[1]+256*request[2] < 20))
                        {
                            if(find_script(request[1], request[2]))
vTaskDelay(1000);
                            if(request[1]+256*request[2] == 19)
find_voice_remote(2, 1);
                            else find_voice_remote(2, request[1]+256*request[2]);
                            ble_mesh_send(gateway, 0xE50211, 8, RECV_VOICE_CMD,
request[1], request[2], 0, 0, 0, 0, 0);
                        }
                        else
                        {
                            if((request[1]+256*request[2] > 19) &&
(request[1]+256*request[2] < 60))
                            {
                                if(find_script(request[1], request[2]))
vTaskDelay(1000);
                                if(request[1]+256*request[2] == 45)
find_voice_remote(3, 31);
                                else find_voice_remote(3,
request[1]+256*request[2]);
                                ble_mesh_send(gateway, 0xE50211, 8,
RECV_VOICE_CMD, request[1], request[2], 0, 0, 0, 0, 0);
                            }
                            else
                            {
                                if((request[1]+256*request[2] > 59) &&
(request[1]+256*request[2] < 71))
                                {
                                    if(find_script(request[1], request[2]))
vTaskDelay(1000);
                                    if(request[1]+256*request[2] == 70)
find_voice_remote(4, 60);
                                    else
                                    {
                                        if(request[1]+256*request[2] == 69)
find_voice_remote(4, 61);
                                        else find_voice_remote(4,
request[1]+256*request[2]);
                                    }
                                    ble_mesh_send(gateway, 0xE50211, 8,
RECV_VOICE_CMD, request[1], request[2], 0, 0, 0, 0, 0);
                                }
                                else
                                {
                                    if(request[1]+256*request[2] == 244)
                                    {
                                        iswaittingvoice = 1;
                                        gpio_set_level(LED, 0);
                                    }
```

```
                                    else
                                    {
                                        if(request[1]+256*request[2] > 109)
                                        {
                                            find_script(request[1], request[2]);
                                            ble_mesh_send(gateway, 0xE50211, 8,
RECV_VOICE_CMD, request[1], request[2], 0, 0, 0, 0, 0);
                                        }
                                    }
                                }
                            }
                        }
                        if(isreceive != 1) rmt_rx_start(RMT_CHANNEL_0, true);
                        irrecv.resume();
                    }
                    else
                    {
                        if(request[1]+256*request[2] == 0)
                        {
                            voicerequest[1] = 244;
                            uart_write_bytes(UART_NUM_0, voicerequest, 3);
                        }
                    }
                    iswaittingvoice = 1;
                    gpio_set_level(LED, 0);
                }
            }
            request[0] = NONE_CMD;
            break;
        case RECV_LOCAL_CMD:
            if(isreceive != 1) rmt_rx_stop(RMT_CHANNEL_0);
            isok = run_script(request[1]+256*request[2]);
            if(isreceive != 1) rmt_rx_start(RMT_CHANNEL_0, true);
            irrecv.resume();
            if(count_relay_list() > 3 * CONFIG_BLE_MESH_CRPL / 5)
            {
                if(!isok) remove_relay_list(request[3]+256*request[4]);
            }
            request[0] = NONE_CMD;
            break;
        case RECV_WAKEUP_CMD:
            gpio_set_level(LED, 1);
            if((request[1] + 256*request[2] > 99) && (request[1] + 256*request[2]
< 110))
            {
                wakeupvoice = request[1] + 256*request[2];
                save_memory_system();
                isok = 1;
            }
            else isok = 0;
            ble_mesh_send(gateway, 0xE50211, 2, RECV_WAKEUP_CMD, isok, 0, 0, 0, 0,
0, 0);
            uart_write_bytes(UART_NUM_0, request, 3);
            vTaskDelay(100);
```

```
            gpio_set_level(LED, 0);
            request[0] = NONE_CMD;
            break;
        case RECV_RESET_CMD:
            kick_out();
            gpio_set_level(LED, 1);
            voicerequest[0] = RECV_WAKEUP_CMD;
            voicerequest[1] = 100;
            uart_write_bytes(UART_NUM_0, voicerequest, 3);
            vTaskDelay(100);
            voicerequest[0] = RECV_VOLUME_CMD;
            voicerequest[1] = 100;
            uart_write_bytes(UART_NUM_0, voicerequest, 3);
            vTaskDelay(100);
            voicerequest[0] = RECV_VOICE_CMD;
            voicerequest[1] = 251;
            uart_write_bytes(UART_NUM_0, voicerequest, 3);
            vTaskDelay(5000);
            esp_restart();
            request[0] = NONE_CMD;
            break;
        case RECV_VOLUME_CMD:
            gpio_set_level(LED, 1);
            ble_mesh_send(gateway, 0xE50211, 2, RECV_VOLUME_CMD, 1, 0, 0, 0, 0, 0,
0);
            uart_write_bytes(UART_NUM_0, request, 3);
            vTaskDelay(100);
            gpio_set_level(LED, 0);
            request[0] = NONE_CMD;
            break;
        case SEND_SCRIPT_CMD:
            gpio_set_level(LED, 1);
            send_list_script();
            vTaskDelay(100);
            gpio_set_level(LED, 0);
            request[0] = NONE_CMD;
            break;
        case SEND_REMOTE_CMD:
            gpio_set_level(LED, 1);
            send_list_remote();
            vTaskDelay(100);
            gpio_set_level(LED, 0);
            request[0] = NONE_CMD;
            break;
    }

    if((isteaching == 3) || (isteaching == 5)){
        timewaittingteaching++;
        if(timewaittingteaching >= 150)
        {
            isteaching = 1;
            ble_mesh_send(gateway, 0xE50211, 5, SEND_TEACHING_CMD, tmprequest[0],
tmprequest[1], tmprequest[2], 2, 0, 0, 0);
            voicerequest[1] = 254;
```

```
            uart_write_bytes(UART_NUM_0, voicerequest, 3);
            gpio_set_level(LED, 0);
            isreceive = check_params_remote();
            if(isreceive != 1) rmt_rx_start(RMT_CHANNEL_0, true);
            else rmt_rx_stop(RMT_CHANNEL_0);
        }
    }

    if(timewaittingresponse){
        timewaittingresponse--;
        if(!timewaittingresponse) ble_mesh_send(gateway, 0x8204, 3, 0xF1, 0, 1, 0,
0, 0, 0, 0);
    }

    if(timeresetvoice > 36000){
        gpio_set_level(PIN_RES, 0);
        vTaskDelay(500);
        gpio_set_level(PIN_RES, 1);
        gpio_set_level(LED, 0);
        iswaittingvoice = 1;
        timeresetvoice = 0;
    }
    else {
        timeresetvoice++;
    }

    //Đọc giá trị nút nhấn để reset
    if(!gpio_get_level(BUTTON))
```