ESP32 Bluetooth Networking User Guide

Version 1.0

Copyright © 2017

About This Guide

This document provides a guide to using ESP32 in Bluetooth networking for IoT devices with examples.

This user guide is structured as follows:

Chapter	Title	Content
Chapter 1	Introduction	A brief introduction to Bluetooth networking for IoT devices and Espressif's app – EspBlufi app.
Chapter 2	APIs for Networking Development	Introduction to APIs related to Bluetooth networking on ESP32 end and the EspBlufi APK end.
Chapter 3	Examples of Bluetooth Networking with ESP32	Examples of Bluetooth networking for IoT devices via EspBlufi.

Release Note

Date	Version	Release notes
2017.03	V1.0	First release.

Content

1. Introduction....1
   1.1. Overview....1
   1.2. EspBlufi....1
2. APIs for Networking Development....3
   2.1. APIs on ESP32 End....3
   2.2. APIs on the EspBlufi APK End....4
3. ESP32 Bluetooth Networking Examples....5
   3.1. Hardware and Software Preparation....5
   3.2. Setting ESP32 to Station Mode....5
   3.3. Setting ESP32 as a SoftAP....9

1. Introduction

1.1. Overview

The ESP32, as a single 2.4 GHz Wi-Fi and Bluetooth combo chip, supports Wi-Fi setups via both SmartConfig and Bluetooth. Users can use ESP32 for secure configuration of Wi-Fi networking for IoT devices.

Using Bluetooth for configuring Wi-Fi network presents the following advantages:

- The Bluetooth protocol is open and scalable.
- By using Bluetooth protocol, users can easily discover nearby devices via Bluetooth beacons.
- The Bluetooth protocol is secure, because the authentication of the device is done over a secure Bluetooth connection before the password is sent to the device.
- Users can also transmit data over Bluetooth to a smartphone, even when the router is not working. The phone can then uploads the data to the Internet.
- The phone can now also connect to the Bluetooth device and send commands directly to control the device if the Wi-Fi network is down.

1.2. EspBlufi

ESP32 is compliant with Bluetooth v4.2 BR/EDR and BLE specifications. Espressif has developed the app EspBlufi specifically for Bluetooth networking of IoT devices. The app is supported on Android 4.3 and above. For smartphones of Android 6.0 or above, because of Google's modification to the Android API, users must give their permission for access to their current location and enable the location information module to start the Bluetooth scanning. EspBlufi is not supported on iOS for the time being.

- EspBlufi can be downloaded via: *https://github.com/EspressifApp/EspBlufi/tree/master/releases*
- Open EspBlufi after downloading it. In the user interface as shown in Figure 1-1, click on the icon in the upper right corner.

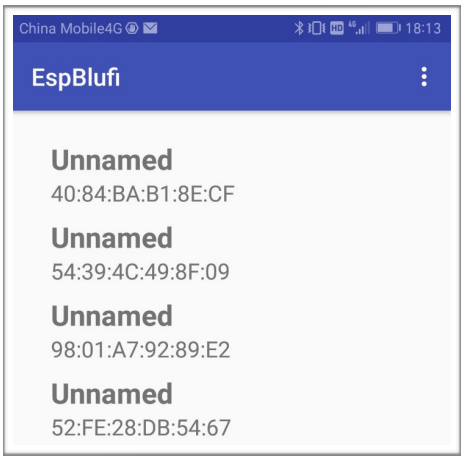

Figure 1-1. EspBlufi User Interface

- The “About” button will be shown in the interface.

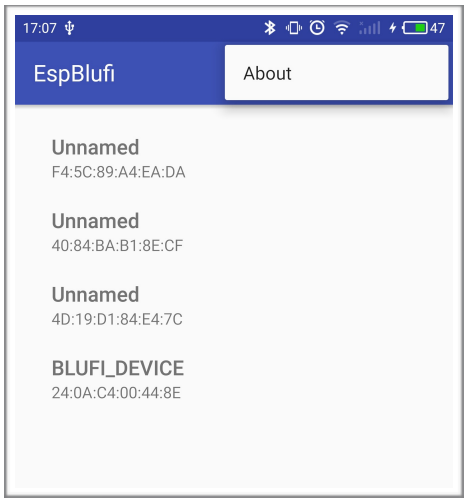

Figure 1-2. About Button

- Click on “About” to check the version of the app. The “Support protocol version” shown in the interface indicates the BluFi protocol version supported by the app.

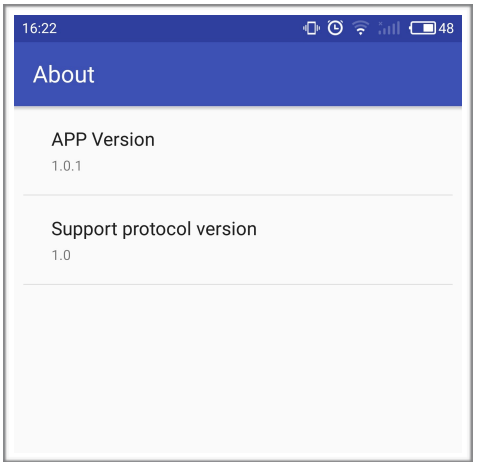

Figure 1-3. EspBlufi App Version Information

2. APIs for Networking Development

2.1. APIs on ESP32 End

In the program of ESP32, security-related processes such as key establishment are determined and developed by the users. The app sends the “negotiate data” to ESP32 and the data packets will be sent to the application layer. If the data is not handled by the application layer, DH algorithm provided by BluFi can be used to exchange the key. The application layer needs to register relevant callback functions to BluFi. Table 2-1 provides description of those functions.

Table 2-1. Security-Related Functions

Function	Description
`typedef void (*esp_blufi_negotiate_data_handler_t) (uint8_t *data, int len, uint8_t **output_data, int *output_len, bool *need_free);`	• This function is used to receive the “negotiate data” during key exchange. The data to be sent should be output by passing the parameters `output_data` and `output_len` to BluFi. • After BluFi calls `negotiate_data_handler`, `output_data` output by `negotiate_data_handler` is sent. • Since the length of data to be sent is not definite, `**output_data` has to `malloc` a part of memory or use global variables to release memory via `need_free`.
`typedef int (* esp_blufi_encrypt_func_t)(uint8_t iv8, uint8_t *crypt_data, int cyprt_len);`	The length of the encoded and decoded data must be the same. `iv8` in this function means the 8-bit sequence of the packets. It can be used for eight bits of the initialization vector.
`typedef int (* esp_blufi_decrypt_func_t)(uint8_t iv8, uint8_t *crypt_data, int crypt_len);`	The length of the encoded and decoded data must be the same. `iv8` in this function means the 8-bit sequence of the packets. It can be used for eight bits of the initialization vector.
`typedef uint16_t (*esp_blufi_checksum_func_t)(uint8_t iv8, uint8_t *data, int len);`	The function is used to calculate the checksum. Calling this function returns the value of checksum. The returned value will be compared with the checksum at the end of the packet by BluFi.

2.2. APIs on the EspBlufi APK End

For SDK classes and related APIs, please refer to Table 2-2.

Table 2-2. SDK Classes for Bluetooth Networking

SDK class	Description	API
`BleScanner`	Used for scanning BLE-enabled devices.	`startLeScan(final BluetoothAdapter.LeScanCallback callback)` • Start scanning for Bluetooth-enabled devices. `LeScanCallback` is the callback for having spotted devices.
		`stopLeScan(BluetoothAdapter.LeScanCallback callback)` • Stops the scanning.
`BleGattClientProxyImpl`	Used for connecting BLE-enabled devices, and acquiring Service and Characteristics.	`connect(@NonNull String bleAddr, long timeout)` • Connects to the BLE devices.
		`discoverCharacteristic(@NonNull BluetoothGattService gattService, @NonNull UUID uuid)` • Acquires the Bluetooth GATT Characteristics.
		`requestMtu(int mtu, long timeout)` • Sets the value of MTU (Maximum Transmission Unit).
		`close()` • Closes the connection.
`BlufiCommunicator`	Used for communicating with BLE-enabled devices.	`getVersion()` • Identifies the device's BluFi version.
		`negotiateSecurity()` • Negotiates the key with BLE devices.
		`getStatus()` • Gets the connection status of the devices.
		`configure(final BlufiConfigureParams params)` • Configures the networking mode for devices.

3. ESP32 Bluetooth Networking Examples

3.1. Hardware and Software Preparation

- 1 × ESP32 module
- 1 × PC, connected to the module to supply power and print log for ESP32.
- 1 × smartphone (Android 4.3 or above)
- Download and install EspBlufi on the smartphone. Turn on the Wi-Fi and Bluetooth capabilities on the phone. The link for downloading EspBlufi is *https://github.com/EspressifApp/EspBlufi/tree/master/releases*

3.2. Setting ESP32 to Station Mode

1. Power on the module. The following log will be output to the serial port tool:

```
E (3155) BT: Startup BTU

E (3165) BT: GATTS_CreateService:  handle of service handle1
E (3175) BT: bta_dm_co_ble_load_local_keys: func not ported

I (3185) BLUFI_DEMO: BD ADDR: 24:0a:c4:01:4d:be

I (3185) BLUFI_DEMO: BLUFI VERSION 0100

I (3185) BLUFI_DEMO: BLUFI init finish
```

2. Open the EspBlufi app, and refresh the interface by swiping down from the top of the screen. The nearby Bluetooth-enabled devices will be shown on the screen.

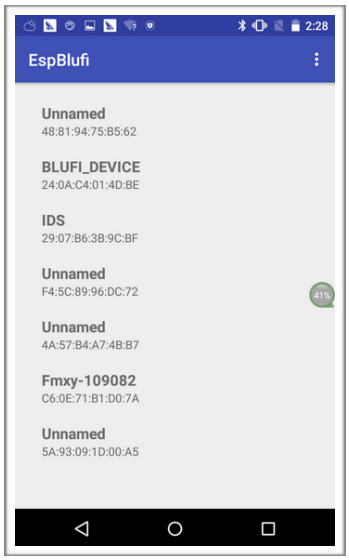

Figure 3-1. EspBlufi Interface

3. Click on the ESP32 device in the interface as Figure 3-1 shows, and establish Bluetooth connection to it. Successful connection will yield the interface shown in Figure 3-2. The duration marked off with a red box in Figure 3-2 indicates the time needed for the connection. The BluFi protocol version (which is "V1.0" in this example), and the Wi-Fi mode (Station/SoftAP/Station+SoftAP) will be shown. So is the device's connection status in Station mode (which is "disconnect" in this example), as well as the number of stations connected in SoftAP mode (which is "0" in the case shown in this example).

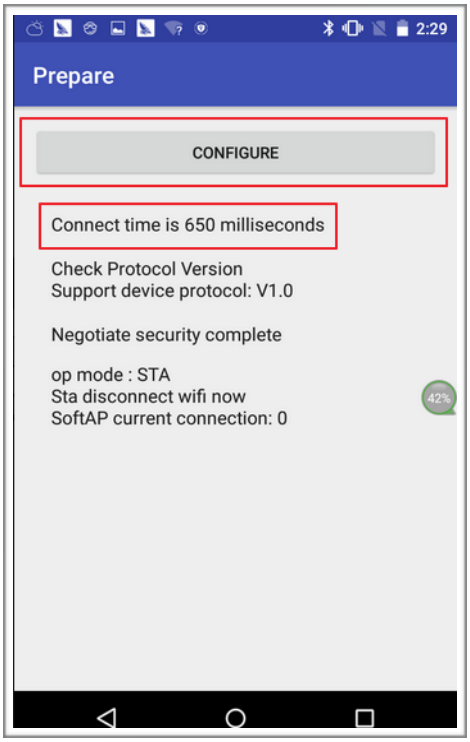

Figure 3-2. Interface Showing Successful Connection

Meanwhile, the following log will be output to the serial port tool:

```
E (37875) BT: btm_ble_resolve_random_addr_on_conn_cmpl unable to match and resol
ve random address
E (38145) BT:
device is connected 48:63:0f:47:72:86, server_if=4,reason=0x0,connect_id=4

E (38145) BT: smp_br_connect_callback is called on unexpected transport 2

I (38145) BLUFI_DEMO: BLUFI ble connect

E (38285) BT: MTU request PDU with MTU size 64

E (38285) BT: BTM_SetBleDataLength failed, peer does not support request
E (38285) BT: Call back not found for application conn_id=3
I (40185) BLUFI_DEMO: BLUFI get wifi status from AP
```

> 📖 ***Note:***
>
> *If the "CONFIGURE" button is not shown or can not be clicked in the interface of Figure 3-2, it means that Bluetooth connection has failed.*

4. Click on the "CONFIGURE" button to get the dropdown menu for network configuration, as Figure 3-3 shows:

Figure 3-3. Network Configuration Interface

5. Select the device mode from the dropdown menu, as Figure 3-4 shows. (BluFi networking supports the following three modes: Station, SoftAP, and Station+SoftAP.)

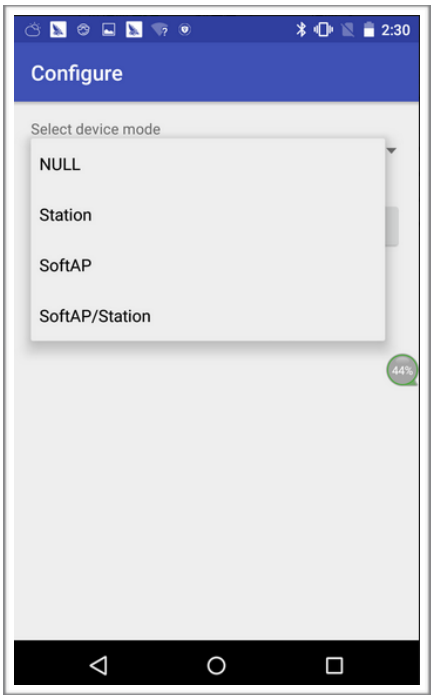

Figure 3-4. Selecting Device Mode

6. In the interface shown in Figure 3-5, click on the refresh button , and choose the Wi-FI SSID. Then, enter the password.

Figure 3-5. Configuring the Station Mode

7. Click on the 确定 button to complete the configuration. Successful configuration will lead to the interface as Figure 3-6 shows. The information marked off with the red box in Figure 3-6 shows the duration of time needed for connection. The Wi-Fi mode (which is Station mode in this example) and the connection information (such as the AP's BSSID and SSID, as well as the connection status) are also shown.

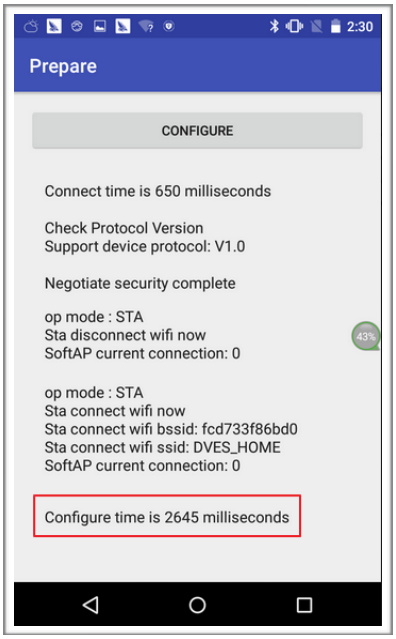

Figure 3-6. Station Connection Information

Meanwhile, the following log will be output to the serial port tool:

```
I (121745) BLUFI_DEMO: BLUFI Set WIFI opmode 1

I (121845) BLUFI_DEMO: Recv STA SSID DVES_HOME

I (121945) BLUFI_DEMO: Recv STA PASSWORD 12345678

I (122085) BLUFI_DEMO: BLUFI requset wifi connect to AP

I (122955) wifi: n:6 2, o:1 0, ap:255 255, sta:6 2, prof:1
I (123615) wifi: state: init -> auth (b0)
I (123625) wifi: state: auth -> assoc (0)
I (123645) wifi: state: assoc -> run (10)
I (123675) wifi: connected with DVES_HOME, channel 6
I (124305) event: ip: 192.168.1.105, mask: 255.255.255.0, gw: 192.168.1.1
I (133645) wifi: pm start, type:0
```

3.3. Setting ESP32 as a SoftAP

1. Power up the module. Connect the EspBlufi app to ESP32 via Bluetooth. Set the device mode as SoftAP in the network configuration interface, as Figure 3-7 shows. For details of doing so, please refer to Steps 1 to 4 in ***Section 3.2***.

Figure 3-7. Selecting SoftAP Mode

2. Configure the SoftAP mode, as Figure 3-8 shows. Select the security encryption mode, channel, and maximum number of Stations to be connected. Enter the SoftAP's SSID and password. Click on the 确定 button to complete the configuration.

Figure 3-8. Configuring the SoftAP Mode

3. Success of the SoftAP configuration will lead to the interface as Figure 3-9 shows. The current Wi-Fi mode and the connection status are shown in the interface.

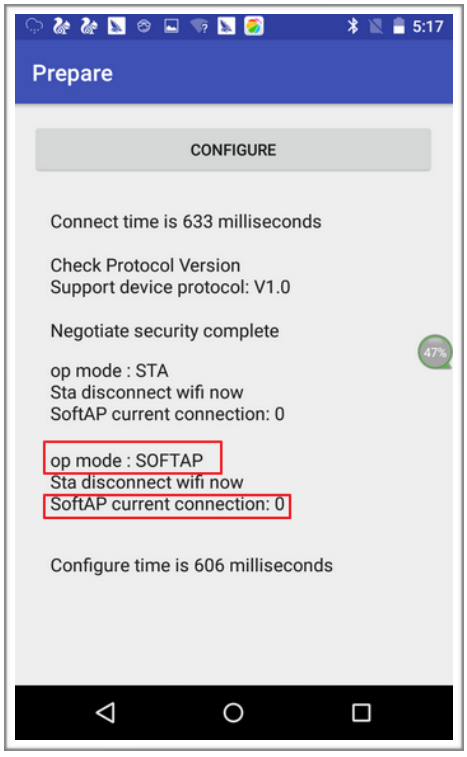

Figure 3-9. SoftAP Connection Information

Meanwhile, the following log will be output to the serial port tool:

```
I (141967) wifi: mode : softAP (24:0a:c4:01:4d:bd)
I (142067) BLUFI_DEMO: Recv SOFTAP SSID blufi_softap
, ssid len 13

I (142167) BLUFI_DEMO: Recv SOFTAP PASSWORD 12345678

I (142267) BLUFI_DEMO: Recv SOFTAP CHANNEL 10

I (142357) BLUFI_DEMO: Recv SOFTAP MAX CONN NUM 2

I (143107) BLUFI_DEMO: Recv SOFTAP AUTH MODE 4
```

4. Turn on the smartphone's Wi-Fi capability. The SoftAP connected can be found in the WLAN interface, as Figure 3-10 shows.

Figure 3-10. The Configured ESP32 SoftAP

5. Connect the smartphone to the ESP32 SoftAP. As shown in Figure 3-11, the prompt 网络已连接 will appear, indicating that the Wi-Fi is connected.

Figure 3-11. Wi Fi Connection Prompt

Meanwhile, the following log will be output to the serial port tool:

```
I (293557) wifi: n:10 0, o:10 0, ap:10 2, sta:255 255, prof:10
I (293557) wifi: station: 98:d6:f7:64:13:08 join, AID=1, n, 20
```

The ESP32 device is successfully connected to Wi-Fi via Bluetooth networking.

> 📖 ***Note:***
>
> *All values of connection time shown on the EspBlufi app screenshots in this document are provided only as examples. Connection time is subject to the actual situations.*

Espressif IOT Team

www.espressif.com

Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED AS IS WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

All liability, including liability for infringement of any proprietary rights, relating to the use of information in this document, is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The Wi-Fi Alliance Member logo is a trademark of the Wi-Fi Alliance. The Bluetooth logo is a registered trademark of Bluetooth SIG.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

Copyright © 2017 Espressif Inc. All rights reserved.