ESP8266

Mesh User Guide

Release notes of this guide

This guide is for the Version 1.4.0 of ESP IOT SDK.

Date	Version	Notes
Jul. 2015	V1.0	First released.
Sep. 2015	V1.1	Chapter 1 "Introduction" Updated; Chapter 3 "APIs" Updated.

Table of Contents

1. Introduction

1.1. Mesh network

With the development of the internet of things, there are an increasing number of nodes. However, due to the limited number of nodes that can be directly connected to one router (usually fewer than 64), it can be inconvenient to have all the nodes directly connected to the same router. There are currently two solutions:

- Super-Router / multiple routers: We can increase the capacity of the routers, so that more nodes can be directly connected to the router or routers.

- Mesh network: The nodes establish a network independently and forward the packets.

1.1.1. ESP-Mesh

It is the trend to have more and more IOT devices in the house that runs on Wi-Fi; however the router are usually limited to connect to only a couple of dozen of devices. Hence, to reduce the loading on the router, Espressif has developed ESP-Mesh to network the IOT devices amongst themselves to reduce the loading to the router. The nodes within the network can independently function as a standalone device or cooperatively, and the network is also self-healing and self organizing.

1.1.2. ESP-Touch

Espressif Systems developed the ESP-Touch protocol to seamlessly configure Wi-Fi devices to connect to the router. This is particularly important for headless systems, because of the lack of a user interface.

1.1.3. Network principle

Mesh network supports auto-networking. When ESP-Touch is used to configure the mesh network, the device automatically scans the Wi-Fi AP nearby.

The device will start with devices whose RSSI are higher than -45 dbm first. Among these devices, the device will always try to connect to the device with the fewest hop(s) away from the router. It will repeat this process until it has joined the mesh network.

If the device fails to do so, it will choose the devices whose RSSI are lower than -45 dbm, and among these devices, it will always try to connect to the device with the maximum RSSI. It will repeat this process until it has joined the mesh network.

1.2. Network diagram

The mesh network is shown as the figure below.
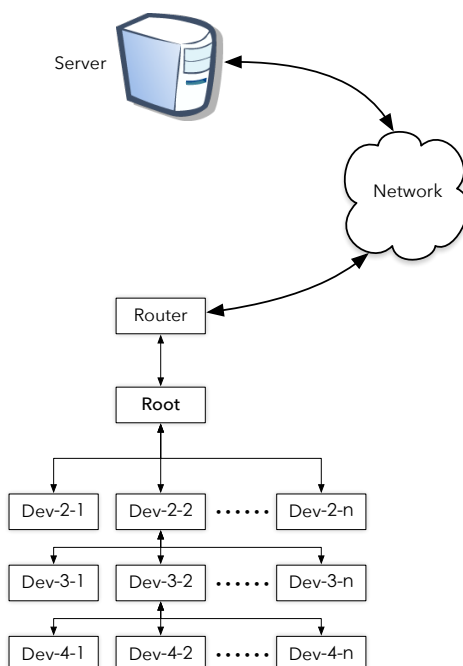
1.2.1.　Node Type

The node that is directly connected to the router is the root node; the others are non-root nodes.

Root node

It receives and sends packets and forwards packets from the server, smartphone apps and its child nodes.

Non-root node

- Non-leaf node: It receives and sends packets, and forwards packets from its parent node and its child nodes.

- Leaf node: It only receives and sends packets, but does not forward packets.

- The level of a device is defined by 1 + (the number of connections between the device and the root device). For root device, the level is 1. Currently the mesh network can support 4 level of the devices.

1.2.2.　Device type

When the router is connected to the server, you can use the smartphone to control the cloud devices in the server; otherwise, you can only control the local devices.

Local device

If a device is configured to the router by ESP-Touch, but not activated on the server-end, it is a local device, as the figure shown below. Such a device is accessible over Wi-Fi when the mobile App is on the Wi-Fi network, but not over the cloud platform.
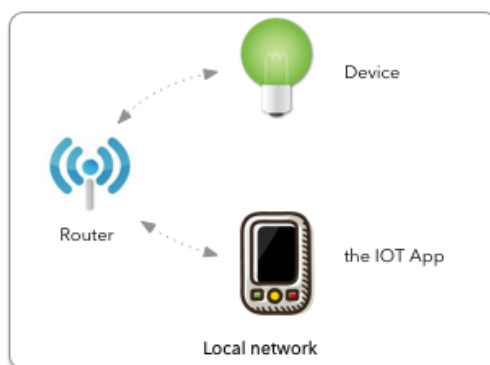
Cloud device

If a device is configured to the router by ESP-Touch, and activated on the server-end, it is now a cloud device. There are three possible connection statuses: on cloud, local Wi-Fi, or offline, as the figure shown below.
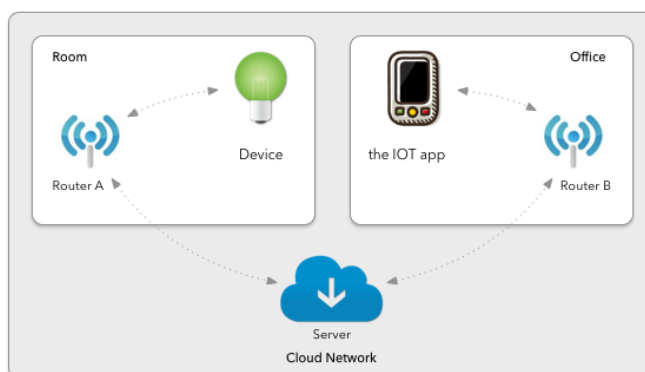
2.　　　Configure the mesh network

2.1.　　Overview

IOT Espressif App (hereinafter referred to as IOT app) is a smartphone application developed by Espressif. It is used to achieve local and remote control of Wi-Fi devices, including smart lights and smart plugs. The app is open source, and found on Github. You can use IOT app to configure the mesh device. The operation process is shown in the figure below.
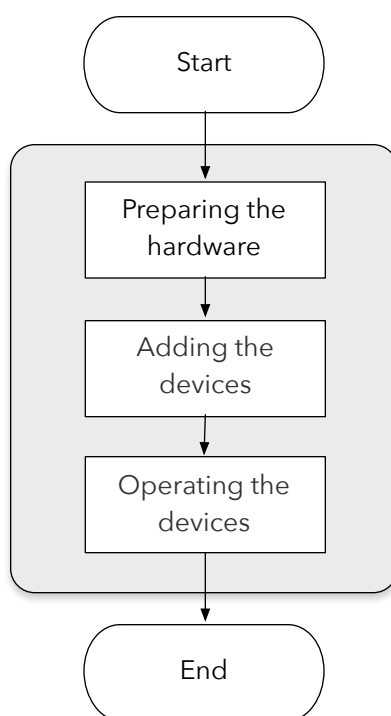
2.2.　　Preparing the hardware

The following hardware devices needed.

Device-end	• A router that can be connected to the internet (If you only need to operate the local devices, you don't need to connect to the internet.)
	• Devices with Wi-Fi modules and latest ESP IOT SDK, e.g. smart lights.
Smart phone-end	• A smartphone with the IOT app.

Note：

You may download the latest SDK and burn the firmware to the development board. For details of what and how to burn, refer to the *ESP8266__IOT_SDK_User Guide*.

2.3.   Adding the devices

Note：

For details of how to login into the system, refer to the *ESP8266_IOT Espressif User Guide*. This section will focus on how to add a new device to the mesh network with ESP-Touch.

You can use ESP-Touch to set the device as a local or cloud device after the firmwares have been burnt to the device.

Instruction

Add a device named ESP_A10666 to the cloud mesh network with the router whose SSID is ESP_IOE.

Note：

Currently, this system only supports smart lights. Other devices can be supported in the next version of the system.

Requirements

• You need a router whose SSID is ESP_IOE, and can be connected to the internet server.

• Related files have been burnt to device ESP_A10666, and the device supports IOT Espressif system and the mesh network.

Procedures

Note：

The actual interface may be different, depending on the version of IOT app.

1. Touch the IOT Espressif App on the main screen to enter the login page.
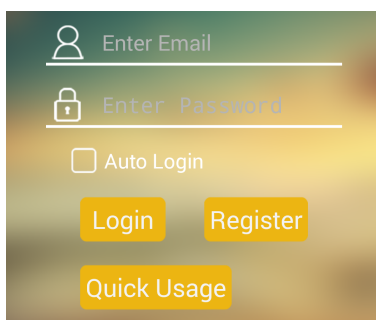
- If you are a new user, touch *Register* to register a new account.

- You can touch *Quick Usage* to operate the local devices.

2. Login into the system with your account and password. The system shows a cloud device list and a local device list.
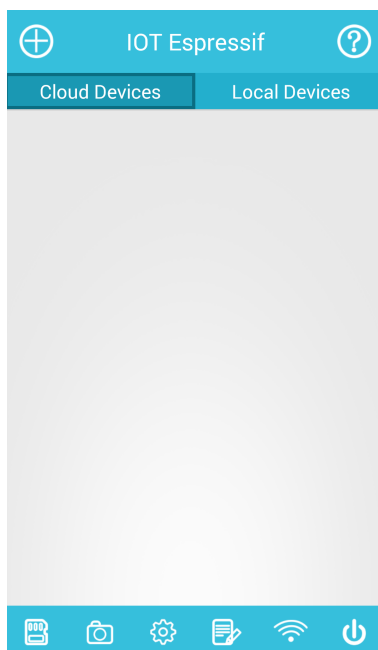
If you are a new user, the lists are empty.

3. Turn on the device. After the white light is on for 5 seconds, the green light will blink, which means the device can be configured by ESP-Touch.

Different colours of the smart light indicate different statuses of the device, as shown in the table below.

No.	Light colour	device status
1	White (constant on)	The device is on.
2	Green (blinking)	The device can be configured by ESP-Touch.
3	Blue (blinking)	A device is added by the IOT app, and is being configured by ESP-Touch.
4	White (blinking)	The device has been successfully configured by ESP-Touch and connected to the router.
5	Red (blinking)	The device hasn't been configured, or has failed to be configured.

4. Touch the button ✚ to enter the *Add devices* page, and then enter the SSID and the password.

The system will remember the password, so you don't need to enter the password again for the same Wi-Fi SSID if you have entered it before.

SSID	This is the Wi-Fi SSID of the smartphone. You can change it in settings of your smartphone.
Show password	Choose this item to check the password you have entered.
This Wi-Fi is hidden	Choose this item if the Wi-Fi is hidden. Most of the Wi-Fis are not hidden.
Activate device	Choose this item so that you can configure the device and activate it on the server-end. • If you want to configure a local device, don't choose *Activate device*. Configuration takes within 1 minute. • If you want to configure a cloud device, choose *Activate device*. Configuration takes about 1 to 2 minutes.

5. Touch *OK*, and the system shows *Configuring....* When the configuration is completed, the system shows the device lists.

> ⚠**Notice**：
>
> The device can only be connected to the mesh network when it's in the smart config mode for ESP-Touch. Don't touch *OK* until the green light is blinking; otherwise, the configuration might fail.
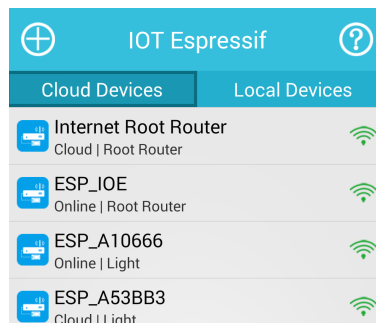
Tips：

- Hold and slide down the screen to refresh the device lists when the system can't auto-refresh.

- If the configuration is completed (at least one device is connected to the mesh network), the system shows *Configuration completed*. Your device will be shown in the cloud device list or local device list.

- If the configuration failed (no device is connected to the mesh network), the system shows *Configuration failed*. Your device will not be shown in the cloud device list or local device list. You will have to try again.

Internet Root Router	It shows all the cloud devices that are of cloud status or online status.
ESP_IOE	The router SSID that the smartphone is connected to. It shows all the cloud devices of online status.
ESP_A10666	The online device that has been successfully configured this time.
ESP_A53BB3	The cloud devices that have been successfully configured in other routers before.

6. Touch *Internet Root Router* in the device list, and you can see all the cloud devices, including cloud status devices and online status devices.
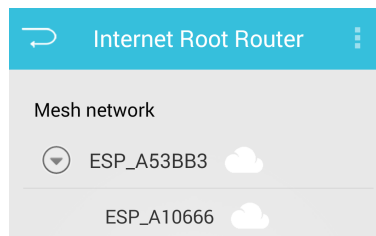
7. Touch *ESP_IOE* in the device list to see all the node devices that are connected to the router.
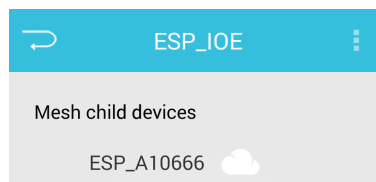
You can see and control the node devices. The device that is directly connected to the router is the root node device, the others are non-root devices.

8. Touch *ESP_A10666* to see the operation page of the device.
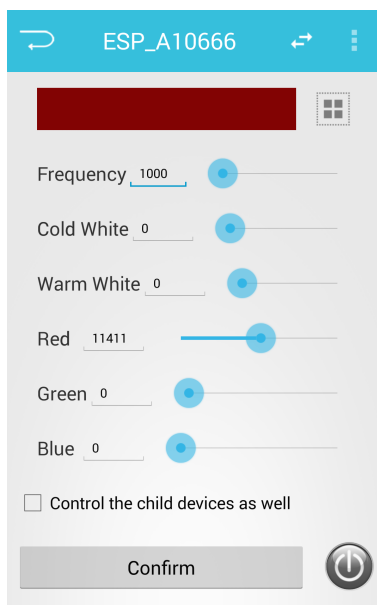
If you choose *Control the child devices as well*, you can control the device and its child devices at the same time.

9. Touch ⇄ in the ESP_A53BB3 operation page to see the network structure of the device.

- The device and all its child devices are shown by default.

- Touch ⇄ to go back to the ESP_A53BB3 operation page.

←
END

3.   APIs

3.1.   Data structure

Data structure of the mesh network is shown as below.

```
typedef void (*espconn_mesh_callback)();
enum mesh_type {
    Mesh_CLOSE = 0,
    Mesh_LOCAL,
    Mesh_ONLINE,
    Mesh_NONE = 0xFF
};
```
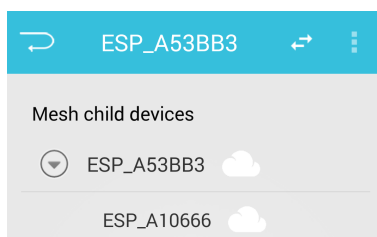
3.2.   Interfaces

3.2.1.   espconn_mesh_enable

Function

Enable mesh.

Functional definition

Definition of `espconn_mesh_enable`

```
void espconn_mesh_enable(espconn_mesh_callback enable_cb, enum mesh_type
type)
```

Definition of `espconn_mesh_callback`

```
typedef void (*espconn_mesh_callback)();
```

Parameters

enable_cb	The system will call `enable_cb` when mesh is enabled.
type	Types of mesh; currently, there are two types of mesh: • Mesh_LOCAL • Mesh_ONLINE

> **Note**:
>
> When `espconn_mesh_enable` is called, users should wait for the system to call `anable_cb`, and make subsequent requests in `enable_cb`.

Return

Null.

3.2.2. espconn_mesh_disable

Function

Disable mesh.

Functional definition

```
void espconn_mesh_disable(espconn_mesh_callback disable_cb)
```

Parameter

disable_cb	The system will call disable_cb when mesh is disabled.

Return

Null.

3.2.3. espconn_mesh_get_status

Function

Get the current status of the mesh network.

Functional definition

```
sint8_t espconn_mesh_get_status();
```

Parameter

Null.

Return

0	Succeed.	
	error code:	
	Mesh_DISABLE	Mesh is disabled.
	Mesh_Wi-Fi_CONN	The mesh node is trying to connect to the Wi-Fi.
Non-0	Mesh_NET_CONN	The mesh node has successfully connected to the Wi-Fi, and is trying to establish a TCP connection.
	Mesh_LOCAL_AVAIL	The node has joined the local mesh network.
	Mesh_ONLINE_AVAIL	The node has joined the cloud mesh network.

3.2.4.   espconn_mesh_connect

Function

Try to connect to mesh.

Functional definition

```
sint8 espconn_mesh_connect(struct espconn *usr_esp)
```

Parameter

usr_esp	User's connection parameter information.

Return

0	Succeed.	
	error code:	
	ESPCONN_MEM	Out of memory.
Non-0	ESPCONN_ARG	Invalid parameter.
	ESPCONN_RTE	Routing problem.
	ESPCONN_ISCONN	Already connected.

3.2.5. espconn_mesh_disconnect

Function

Disconnect mesh.

Functional definition

```
sint8 espconn_mesh_disconnect(struct espconn *usr_esp)
```

Parameter

usr_esp   User's connection parameter information.

Return

0	**Succeed.**	
Non-0	**error code:**	
	ESPCONN_ARG	Invalid parameter.

3.2.6. espconn_mesh_sent

Function

Use mesh connection to send packets.

Functional definition

```
sint8 espconn_mesh_sent(struct espconn *usr_esp, uint8_t *pdata, uint16_t len)
```

Parameters

usr_esp	User's connection parameter information.
pdata	Pointer of data packet.
len	Length of data packet.

Return

0	Succeed.		
Non-0	error code:		
	ESPCONN_MEM	Out of memory.	
	ESPCONN_ARG	Invalid parameter.	
	ESPCONN_MAXNUM	Buffer of sending data is full.	

3.2.7.   espconn_mesh_set_max_hop

Function

Set the maximum number of hop of mesh network.

Functional definition

```
bool espconn_mesh_set_max_hop(uint8_t max_hop)
```

Parameter

max_hop	The maximum max_hop supported by mesh network.

Return

True	Succeed to set.
False	Fail to set.

Note：

The maximum number of hop supported by mesh is 10. If the number is larger than 10, it will fail to set.

3.2.8.   espconn_mesh_get_max_hop

Function

Set the maximum number of hop of mesh network.

Functional definition

```
uint8_t espconn_mesh_get_max_hop()
```

Return

max_hop	The maximum number of hop of the current mesh network.

3.2.9.   espconn_mesh_get_node_info

Function

Set relevant information of the current node.

Functional definition

Definition of `espconn_mesh_get_node_info`

```
bool espconn_mesh_get_node_info(enum mesh_node_type type, uint8_t
**info,uint8_t*count)
```

Definition of `mesh_node_type`

```
enum mesh_node_type {
MESH_NODE_PARENT=0, MESH_NODE_CHILD, MESH_NODE_ALL
                  };
```

Parameters

type	Types of mesh node; currently, there are three types of mesh node: • MESH_NODE_PARENT=0: Indicate information of parent node. • MESH_NODE_CHILD: Indicate information of child node. • MESH_NODE_ALL: Indicate information of all nodes.
info	The information of the node.
count	Number of child nodes.

Return

True	Succeed.
False	Fail.

> ⚠ **Notice：**
>
> Do not use interfaces which are not described in this guide.