ESP-NOW
User Guide

Version 1.0

About This Guide

This document introduces ESP-NOW technology developed by Espressif. The document focuses on ESP-NOW features, how to use it and the demo code.

The structure is as below:

Chapter	Title	Subject
Chapter 1	ESP-NOW Introduction	Introduction on ESP-NOW technology and features.
Chapter 2	ESP-NOW Usage Guide	Description of the device information; guidance on how to use ESP-NOW.
Chapter 3	Sample Code	Provision of the sample code of ESP-NOW.

Release Notes

Date	Version	Release notes
2016.07	V1.0	First release.

Table of Contents

1. ESP-NOW Introduction

1.1. Overview

ESP-NOW is a fast, connectionless communication technology featuring short packet transmission. ESP-NOW is ideal for smart lights, remote control devices, sensors and other applications.

ESP-NOW applies IEEE802.11 Action Vendor frame technology, along with the IE function developed by Espressif and CCMP encryption technology, realizing a secure, connectionless communication solution.

1.2. Features

ESP-NOW supports the following features:

- Encrypted and unencrypted unicast communication.
- Mixed encrypted and unencrypted peer devices.
- Up to 250 byte payload can be carried.
- Send callback function can be set to inform the application layer of transmission success or failure.

ESP-NOW features also have the following limitations:

- Broadcast is not supported.
- Limited encrypted peers. 10 encrypted peers at most are supported in Station mode, 6 at most in SoftAP or SoftAP + Station mode. Multiple unencrypted peers are supported, however, together with encrypted peers, the sum should be less than 20.
- Payload is limited to 250 bytes.

2. ESP-NOW Usage Guide

2.1. Information Description

A linked list of the local device and the peer will be maintained in a low level layer of ESP-NOW. This information is used to send and receive data. ESP-NOW maintains the essential information such as MAC address and key in the lower layer for the peer. Besides, ESP-NOW also stores the frequently used data for the application layer to avoid the overhead of secondary maintenance of the linked list.

The information involved includes:

- In the local device:
  - PMK
  - Role
- In the peer (including frequently used information and other user-defined information):
  - Key
  - MAC Address
  - Role
  - Channel

For detailed description of the information, please refer to Table 2-1.

Table 2-1. Information Description

Device	Information	Value / length	Description	Note
Local device	PMK	Length: 16 bytes	Primary Master Key, i.e., KOK in API, used to encrypt the Key of the peer.	The system will maintain a default PMK, therefore, no configuration is required. If needed, please make sure it is the same as that of the local device.
	Role	IDLE CONTROLLER SLAVE COMBO	The device's role. IDLE: undefined role CONTROLLER: controller SLAVE: slave COMBO: double role as controller and slave	The local device's Role will define the transmitting interface (SoftAP interface or Station interface) of ESP-NOW. IDLE: data transmission is not allowed. CONTROLLER: priority is given to Sation interface SLAVE: priority is given to SoftAP interface COMBO: priority is given to SoftAP interface Station interface for Station only mode and SoftAP interface for SoftAP only mode.

Device	Information	Value / length	Description	Note
Peer	Key	Length: 16 bytes	Used to encrypt the payload Key during communication with the specified peer.	–
	Mac Address	Length: 6 bytes	MAC address of the peer.	MAC address must be the same as the sending address. For example, if the packet is sent from the Station interface, the MAC address should be the same as the Station address.
	Role	IDLE CONTROLLER SLAVE COMBO	The device's role. IDLE: undefined role CONTROLLER: controller SLAVE: slave COMBO: double role as controller and slave	The peer's Role does not affect any function, but only stores the Role information for the application layer.
	Channel	Value: 0 ~ 255	The channel in which the local device and the peer communicate.	Channel does not affect any function, but only stores the Channel information for the application layer. The value is defined by the application layer. For example, 0 can mean that the Channel is not defined; 1 ~ 14 valid Channel; and others specified functions.

2.2. Usage Process

1. Set sending callback function

Sending callback function can be used to tell transmission success or failure, e.g., if the information in the MAC sublayer is conveyed successfully.

Please note the following points carefully when using sending callback function:

- For unicast communication:
  - If the application layer does not receive the packet but the callback function returns success, the reasons may be:
    - attacks from rogue device
    - encrypted Key setting mistake
    - packet loss in the application layer

> 📖 ***Note:***
>
> *Handshake is must for guaranteed transmission success rate.*

  - If the application layer has received the packet but the callback function returns failure, the reasons may be:

- The channel is busy and ACK is not received.

> 📖 ***Note:***
> *The application layer may retransmit the packet and the receiving end needs to check the retransmitted packet.*

- For multicast communication (broadcast communication also included):
  - If the callback function returns success, it means the packet has been sent successfully.
  - If the callback function returns failure, it means the packet has not been sent successfully.

2. Set receiving callback function

Receiving callback function can be used to inform the application layer as the packet sent by the peer is received.

The receiving callback function will return the MAC address of the peer and the payload of the packet.

3. If the Key needs to be encrypted, the API that set PMK (KOK) can be called for configuration.

If PMK is not configured, then the default PMK will be used.

4. Select the communication interface for the devices.

Usually, Station interface is set for CONTROLLER, SoftAP interface for SLAVE AND COMBO.

> 📖 ***Note:***
> *It is not recommended to send packets to a device in Station only mode, for the device may be in sleep.*

5. Select the same Key for the devices. Call the function to add peers.

Please refer to Table 2-1 for details.

6. Call the sending function to return payload.

If the sending function returns the specified MAC address, then it will be sent to the specified device. If the sending function returns NULL, then it will be sent to all the peers, which may result in transmission failure or delay due to network congestion.

3. Sample Code

> 📖 ***Note:***
> *For more information on ESP-NOW APIs, please refer to ESP8266 Non-OS SDK API Guide.*

```
void ICACHE_FLASH_ATTR simple_cb(u8 *macaddr, u8 *data, u8 len)
{
	int i;
	u8 ack_buf[16];
	u8 recv_buf[17];

	os_printf("now from[");
	for (i = 0; i < 6; i++)
		os_printf("%02X, ", macaddr[i]);
	os_printf(" len: %d]:", len);

	os_bzero(recv_buf, 17);
	os_memcpy(recv_buf, data, len<17?len:16);

	if (os_strncmp(data, "ACK", 3) == 0)
		return;

	os_sprintf(ack_buf, "ACK[%08x]", ack_count++);
	esp_now_send(macaddr, ack_buf, os_strlen(ack_buf));
}

void user_init(void)
{
	u8 key[16]= {0x33, 0x44, 0x33, 0x44, 0x33, 0x44, 0x33, 0x44,
0x33, 0x44, 0x33, 0x44, 0x33, 0x44, 0x33, 0x44};
	u8 da1[6] = {0x18, 0xfe, 0x34, 0x97, 0xd5, 0xb1};
	u8 da2[6] = {0x1a, 0xfe, 0x34, 0x97, 0xd5, 0xb1};

	if (esp_now_init()==0) {
```

```
        os_printf("esp_now init ok\n");

        esp_now_register_recv_cb(simple_cb);
        esp_now_set_self_role(1);
        esp_now_add_peer(da1, 1, key, 16);
        esp_now_add_peer(da2, 2, key, 16)

    } else {
        os_printf("esp_now init failed\n");
    }
}

void ICACHE_FLASH_ATTR demo_send(u8 *mac_addr, u8 *data, u8 len)
{
    esp_now_send(NULL, data, len);/* the demo will send to two
devices which added by esp_now_add_peer() */
    //esp_now_send(mac_addr, data, len); /* send to the specified
mac_addr */
}
```

Espressif IOT Team

www.espressif.com