ESP-NOW Projekt Dokumentation

Übersicht - Sender (Farbe Rot)

Dieses Programm ermöglicht einem ESP32, Daten drahtlos an einen anderen ESP32 (Repeater) über **ESP-NOW** zu senden. ESP-NOW ist ein Protokoll von Espressif, das eine schnelle, verbindungslose Kommunikation zwischen ESP-Geräten erlaubt, ähnlich wie ein leichteres WLAN ohne Router.

Bibliotheken

```
1   #include <WiFi.h>
2   #include <esp_now.h>
```

- **WiFi.h** – wird benötigt, um den ESP32 in den WiFi-Station-Modus zu setzen und die MAC-Adresse auszulesen.

- **esp_now.h** – stellt die Funktionen für ESP-NOW bereit (initialisieren, Nachrichten senden/empfangen, Peers verwalten *(peers sind die Beteiligten an der Kommunikation also beim Wlan wär das Handy einer der Peers)*).

MAC-Adresse des Empfängers

```
4
5   uint8_t repeaterMac[] = {0xA0, 0xA3, 0xB3, 0x97, 0x81, 0x00};
6
```

- Hier wird die MAC-Adresse des Geräts eingetragen, an das die Daten gesendet werden sollen (Repeater).

- MAC-Adresse ist eindeutig für jedes WLAN-Gerät.

Datenstruktur

```
7   typedef struct msg {
8     int value;
9   } msg;
10
11  msg data;
```

- Definiert eine Struktur `msg` mit einem Integer-Feld `value`.
- Diese Struktur wird für den Datenaustausch verwendet.

Setup-Funktion

```
14  void setup() {
15    Serial.begin(115200);
16    WiFi.mode(WIFI_STA);
17    delay(1000);
18
19    Serial.print("Sender MAC: ");
20    Serial.println(WiFi.macAddress());
```

1. **Serielle Schnittstelle starten** – für Debug-Ausgaben.
2. **WiFi-Modus setzen** – `WIFI_STA` aktiviert den Station-Modus (notwendig für ESP-NOW).
3. **Sender MAC anzeigen** – MAC-Adresse des ESP32 ausgeben.

```
22
23    if (esp_now_init() != 0) {
24      Serial.println("ESP-NOW init failed");
25      return;
26    }
```

- **ESP-NOW initialisieren**. Falls es fehlschlägt, wird eine Fehlermeldung ausgegeben.

```
28    esp_now_peer_info_t peer;
29    memset(&peer, 0, sizeof(peer));
30    memcpy(peer.peer_addr, repeaterMac, 6);
31    peer.channel = 1;
32    peer.encrypt = false;
33    if (esp_now_add_peer(&peer) != ESP_OK) {
34      Serial.println("Fehler beim Hinzufügen des Peers");
35    }
```

Einen Peer für ESP-NOW definieren und hinzufügen:

- `peer_addr` → MAC-Adresse des Empfängers
- `channel` → WiFi-Kanal (hier 1)
- `encrypt` → Verschlüsselung aus (`false`)

`esp_now_add_peer` fügt den Peer hinzu, sonst Fehlermeldung.

Loop - Funktion

```
37  void loop() {
38    data.value = random(0, 100);
39    esp_now_send(repeaterMac, (uint8_t*)&data, sizeof(data));
40    Serial.print("Gesendet: ");
41    Serial.println(data.value);
42    delay(1000);
43  }
```

1. **Zufallswert erzeugen** – `data.value` bekommt einen Wert zwischen 0 und 99.
2. **Daten senden** – an die MAC-Adresse des Repeaters über ESP-NOW.
3. **Debug-Ausgabe** – zeigt den gesendeten Wert im Serial Monitor an.
4. **Pause von 1 Sekunde** – bevor der nächste Wert gesendet wird.

Zusammenfassung

Dieses Programm:

- Initialisiert ESP-NOW auf einem ESP32.
- Definiert einen Empfänger (Peer) über MAC-Adresse.
- Sendet jede Sekunde zufällige Integer-Werte an den Empfänger.
- Gibt den gesendeten Wert im Serial Monitor aus.

Anwendungsbeispiel: Sensorwerte drahtlos an ein zentrales Gerät (Repeater) senden, ohne WLAN-Router zu verwenden.

Übersicht - Repeater (Farbe Schwarz)

Dieses Programm erlaubt einem ESP32, als **Relay (Repeater)** zu fungieren: Es empfängt Daten von einem Sender über **ESP-NOW**, zeigt sie im Serial Monitor an und leitet sie dann an einen **Receiver** weiter.

Bibliotheken

```
#include <WiFi.h>
#include <esp_now.h>
```

- **WiFi.h** – benötigt für WiFi-Modus und MAC-Adresse.
- **esp_now.h** – stellt ESP-NOW Funktionen bereit (initialisieren, Daten senden/empfangen, Peers verwalten).

MAC-Adresse des Empfängers

```
uint8_t receiverMac[] = {0xD4, 0x8A, 0xFC, 0xA4, 0x21, 0xB8};
```

- MAC-Adresse des endgültigen Empfängers, an den der Repeater die Daten weiterleiten soll.

Datenstruktur

```
typedef struct msg {
  int value;
} msg;

msg data;
```

- Definiert die Struktur `msg` mit einem Integer-Feld `value`.

- Diese Struktur wird zum Speichern und Weiterleiten der empfangenen Daten genutzt.

Callback-Funktion bei Empfang

```
15  void onReceive(const uint8_t *mac_addr, const uint8_t *incomingData, int len) {
16    if(len == sizeof(msg)) {
17      memcpy(&data, incomingData, sizeof(data));
18      Serial.print("Relay empfing: ");
19      Serial.println(data.value);
20
21      esp_now_send(receiverMac, (uint8_t*)&data, sizeof(data));
22    }
23  }
```

1. **Parameter:**
   - `mac_addr` → MAC-Adresse des Senders
   - `incomingData` → empfangene Daten
   - `len` → Länge der Daten
2. Prüft, ob die Daten die erwartete Größe (`sizeof(msg)`) haben.
3. Kopiert die Daten in die lokale Variable `data`.
4. Zeigt den empfangenen Wert im Serial Monitor an.
5. Leitet die Daten sofort an den definierten Empfänger weiter (`esp_now_send`).

Setup-Funktion

```
25  void setup() {
26    Serial.begin(115200);
27    WiFi.mode(WIFI_STA);
28    delay(1000);
29
30    Serial.print("Repeater MAC: ");
31    Serial.println(WiFi.macAddress());
32
```

- Serielle Schnittstelle starten.
- ESP32 in WiFi-Station-Modus setzen.
- Repeater-MAC-Adresse ausgeben.

```
35    if (esp_now_init() != 0) {
36      Serial.println("ESP-NOW init failed");
37      return;
38    }
```

- Initialisiert ESP-NOW, Fehlermeldung bei Misserfolg.

```
41    esp_now_peer_info_t peer;
42    memset(&peer, 0, sizeof(peer));
43    memcpy(peer.peer_addr, receiverMac, 6);
44    peer.channel = 1;
45    peer.encrypt = false;
46    if (esp_now_add_peer(&peer) != ESP_OK) {
47      Serial.println("Fehler beim Hinzufügen des Peers");
48    }
```

- Fügt den Empfänger als Peer hinzu, damit Daten gesendet werden können.

```
49
50    esp_now_register_recv_cb(onReceive);
51
```

- Registriert die Callback-Funktion, die bei eingehenden ESP-NOW Nachrichten ausgeführt wird.

Loop-Funktion

```
53
54  void loop() {}
55
```

- Bleibt leer, da alle Aktionen **ereignisgesteuert** über die Callback-Funktion `onReceive` ablaufen.

Zusammenfassung

Dieses Programm:

- Initialisiert ESP-NOW auf einem ESP32.
- Registriert einen Peer (den finalen Empfänger) für Weiterleitung.
- Empfängt Daten von einem Sender.
- Zeigt die empfangenen Daten im Serial Monitor an.
- Leitet die Daten automatisch an den Receiver weiter.

Anwendungsbeispiel: Ein Sensornetzwerk, bei dem ein ESP32 Daten von mehreren Sendern sammelt und an einen zentralen Empfänger weiterleitet, ohne Router.

Übersicht - Empfänger (Farbe Blau)

Dieses Programm erlaubt einem ESP32, Daten über **ESP-NOW** zu empfangen. Es zeigt die empfangenen Werte im Serial Monitor an.

- Es gibt **keine Weiterleitung**, dieser ESP32 ist der finale Empfänger.

Bibliotheken

```
#include <WiFi.h>
#include <esp_now.h>
```

- **WiFi.h** – nötig für den WiFi-Modus und MAC-Adresse.
- **esp_now.h** – stellt ESP-NOW Funktionen bereit (initialisieren, empfangen).

Datenstruktur

```
typedef struct msg {
  int value;
} msg;

msg data;
```

- Definiert die Struktur `msg` mit einem Integer-Feld `value`.
- Zum Speichern der empfangenen Daten.

Callback-Funktion bei Empfang

```
void onReceive(const uint8_t *mac_addr, const uint8_t *incomingData, int len) {
  if(len == sizeof(msg)) {
    memcpy(&data, incomingData, sizeof(data));
    Serial.print("Empfangen am Receiver: ");
    Serial.println(data.value);
  }
}
```

1. **Parameter:**
   - `mac_addr` → MAC-Adresse des Senders
   - `incomingData` → empfangene Daten
   - `len` → Länge der Daten
2. Prüft, ob die Daten die erwartete Größe (`sizeof(msg)`) haben.
3. Kopiert die Daten in die lokale Variable `data`.
4. Gibt die empfangenen Werte im Serial Monitor aus.

Setup-Funktion

```
24  void setup() {
25    Serial.begin(115200);
26    WiFi.mode(WIFI_STA);
27    delay(1000);
28
29    Serial.print("Empfänger MAC: ");
30    Serial.println(WiFi.macAddress());
31
```

1. Serielle Schnittstelle starten.
2. WiFi-Station-Modus aktivieren.
3. MAC-Adresse des ESP32 ausgeben.

```
33    if (esp_now_init() != 0) {
34      Serial.println("ESP-NOW init failed");
35      return;
36    }
37
38    esp_now_register_recv_cb(onReceive);
39
```

- ESP-NOW initialisieren.
- Callback-Funktion registrieren, damit empfangene Nachrichten automatisch verarbeitet werden.

Loop-Funktion

```
41
42  void loop() {}
43
```

- Bleibt leer, da alle Aktionen ereignisgesteuert über `onReceive` ablaufen.

Zusammenfassung

Dieses Programm:

- Initialisiert ESP-NOW auf einem ESP32.
- Empfängt Daten von einem Sender oder Repeater.
- Zeigt die empfangenen Daten im Serial Monitor an.
- Fungiert als **finaler Empfänger** im ESP-NOW Netzwerk.

Anwendungsbeispiel: Ein Sensornetzwerk, bei dem ein Repeater Daten sammelt und der Receiver diese endgültig empfängt und auswertet.