ESP-NOW ist ein Protokoll, welches die Möglichkeit bietet, mehrere Geräte zu verbinden, ohne das lokale Netzwerk nutzen zu müssen. Hiermit sind ESP8266 oder ESP32 in der Lage autark zu Kommunizieren. In diesem Beitrag betrachten wir eine One-way Kommunikation zwischen zwei ESP8266

**ESP NOW in Kürze**

Esp now nutzt den im ESP verbauten Wlan-chip um zu Komunizieren. Hirbei ist die Besonderheit das kein Wlannetzt benötigt, da die ESPs selber direkt ohne netztwerk komoniziren. Zudem benötigen wir keine zusätzliche Bibiolothek, da ESP-Now zu den standart Bibiolotheken der ESP-Boards gehört. Wie Du ESPs mit der Arduino-IDE entwickeln kannst und somit auch alle benötikten Bibiolotheken hast findest du in unserem Beitrag:

**Funktionen im Überblick**

**esp\_now\_init()**: Zum Initialisieren von ESP-NOW. Liefert 0 bei erfolgreicher Initialisierung.

**esp\_now\_deinit()**: zur Deinitialisieren von ESP-NOW.

**esp\_now\_set\_self\_role()**: Stellt die jeweilige Rolle ein.

**esp\_now\_add\_peer()**: Hinzufügen eines Gerätes für die jeweilige Mac-Adresse.

**esp\_now\_send()**: Senden einer Nachricht.

**esp\_now\_register\_send\_cb()**: Registriert eine Funktion, welche nach dem senden einer Nachricht aufgerufen wird.

**esp\_now\_register\_recv\_cb()**: Registriert eine Funktion, welche nach dem empfangen einer Nachricht aufgerufen wird.

**ESP Rollen**

**ESP\_NOW\_ROLE\_IDLE**: keine Funktion.

**ESP\_NOW\_ROLE\_CONTROLLER**: Master, Priorität auf Station interface.

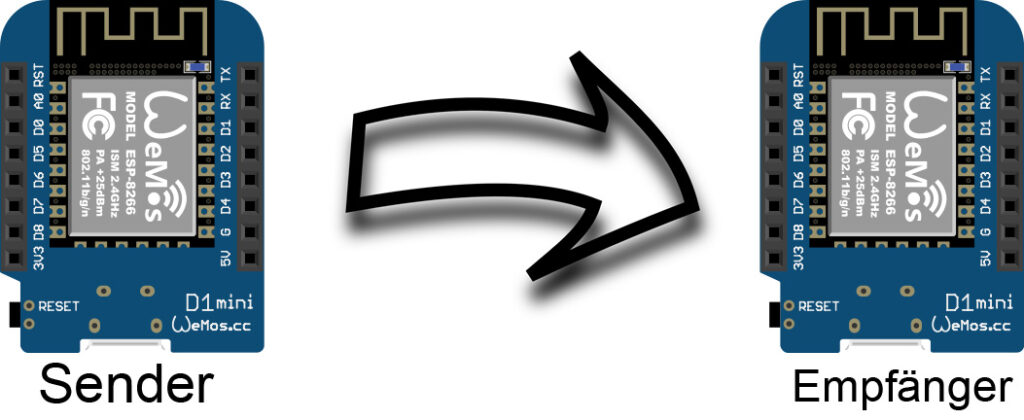
**ESP\_NOW\_ROLE\_SLAVE: Slave**, Priorität auf SoftAP interface.

**ESP\_NOW\_ROLE\_COMBO**: Master + Slave, Priorität auf SoftAP interface.

**ESP\_NOW\_ROLE\_MAX**: Master + Slave, Priorität auf Station interface.

Durch das setzen der Rolle, mit der Funktion **esp\_now\_set\_self\_role(),** werden die dazugehörigen Interfaces frei.

**One-way Kommunikation**

ESP-NOW One-way Kommunikation

In diesem Fall betrachten wir die Möglichkeit, Nachrichten von einem Gerät auf ein anderes zu übertragen.

Tipp

Der serielle Monitor der Arduino IDE kann nur einmal pro Instanz benutzt werden. Wenn man den seriellen Monitor parallel auf zwei COM-Ports nutzen möchte, muss die Arduino IDE mehrfach gestartet werden.

**Sender Code**

// Bnoetgte Bibliotheken

#include <ESP8266WiFi.h>

#include <espnow.h>

// Trage hier die MAC-Adresse des Empfaengers ein

uint8\_t macAddress[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

// struct welches übertragen wird

// Muss mit Empfaenger uebereinstimmen

typedef struct message {

long number;

// Speicher reserviert

char string[64];

} message;

// Erzeuge struct

message myMessage;

// Wird aufgerufen wenn die Nachricht gesendet wurde

void messageSend(uint8\_t \*mac\_addr, uint8\_t status) {

if (status == 0){

Serial.println("Erfolgreich gesendet");

} else {

Serial.println("Fehler beim senden");

}

}

void setup() {

// Starte seriellen Monitor

Serial.begin(115200);

// Setze Geraet in Station mode

WiFi.mode(WIFI\_STA);

// Init ESP-NOW

if (esp\_now\_init() != 0) {

Serial.println("Error initializing ESP-NOW");

}

// Setze Rolle das Geraetes

esp\_now\_set\_self\_role(ESP\_NOW\_ROLE\_CONTROLLER);

// Setze callback Funktion

esp\_now\_register\_send\_cb(messageSend);

// Koppeln auf Chanel 1

// null und 0 können durch Passwort und Passwortlenge ersetzt werden

// betrachten wir in diesem Fall aber nicht

esp\_now\_add\_peer(macAddress, ESP\_NOW\_ROLE\_SLAVE, 1, NULL, 0);

}

void loop() {

// setze Werte

myMessage.number = random(100); // zufaellige Zahl

char string[32] = "Das ist ein String";

memcpy(&myMessage.string, string, sizeof(string));

// Sende Nachricht ueber ESP-NOW

esp\_now\_send(macAddress, (uint8\_t \*) &myMessage, sizeof(myMessage));

// 2 Sekunden warten

delay(2000);

}

**Empfänger Code**

Dieser Code wird auf dem zweiten Gerät hochgeladen. Hierbei empfängt dieser Code eine Nachricht und gibt sie über den seriellen Monitor aus.

// Bnoetgte Bibliotheken

#include <ESP8266WiFi.h>

#include <espnow.h>

// struct welches übertragen wird

// Muss mit Sender uebereinstimmen

typedef struct message {

long number;

char string[64];

} message;

// Erzeuge struct

message myMessage;

// Diese Funktion wird aufgerufen wenn Daten empfangen werden

void recive(uint8\_t \* incommingMacAddress, uint8\_t \*incomingData, uint8\_t len) {

// Daten umkopierren um ggf. spaeter nutzen zu koennen

memcpy(&myMessage, incomingData, sizeof(myMessage));

// Daten ausgeben

Serial.print("Number: ");

Serial.println(myMessage.number);

Serial.print("String: ");

Serial.println(myMessage.string);

Serial.println();

}

void setup() {

// Starte seriellen Monitor

Serial.begin(115200);

delay(100);

// Gibt MAC-Adresse aus

Serial.print("MAC Address: ");

Serial.println(WiFi.macAddress());

// Setze Geraet in Station mode

WiFi.mode(WIFI\_STA);

// Init ESP-NOW

if (esp\_now\_init() != 0) {

Serial.println("Error initializing ESP-NOW");

}

// Setze Rolle

esp\_now\_set\_self\_role(ESP\_NOW\_ROLE\_SLAVE);

// Sezte Callback Funktion

esp\_now\_register\_recv\_cb(recive);

}

void loop() {

}

Hinweis

Strings mit variabler Länge sollten nicht im zu übertragenden **struct** verwendet werden. Der Grund hierfür ist, dass zur Übertragung die Größe der Daten benötigt wird. Bei einem Arduino-String ist diese Größe allerdings variabel und würde den Versand der Daten deutlich komplizierter machen. Aus diesem Grund verwenden wir ein **char[]** mit einer festen Länge von 64 Zeichen.

Wenn du die beiden Code Beispiele auf zwei ESP8266 spielst, können diese miteinander kommunizieren. Passe diesen Code an deine Bedürfnisse an und setze deine Ideen damit um. Wir freuen uns auf deine Kommentare!