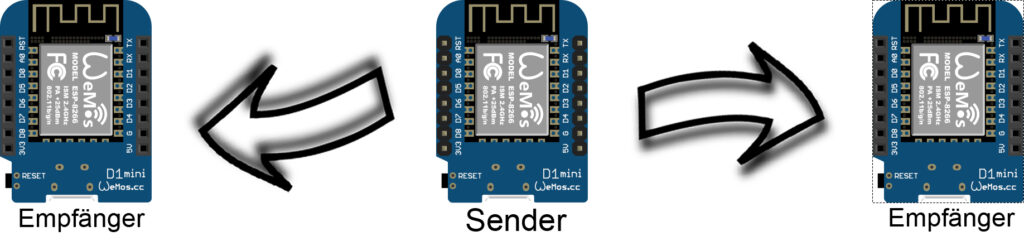
ESP-NOW ist ein Protokoll, welches die Möglichkeit bietet, mehrere Geräte zu verbinden, ohne das lokale Netzwerk nutzen zu müssen. Hiermit sind ESP8266 oder ESP32 in der Lage autark zu kommunizieren. In diesem Beitrag betrachten wir eine One-to-many Kommunikation zwischen zwei ESP8266.

ESP-NOW One-to-many Kommunikation

**Code Beispiel**

**Sender**

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.

**Gleiche Nachricht**

Wenn Du mehr Informationen zum Thema MAC-Adressen beim ESP wissen möchtest, schau bei unserem Beitrag ESP Basics – MAC-Adresse herausfinden vorbei.

#include <ESP8266WiFi.h>

#include <espnow.h>

// Trage hier die MAC-Adressen der Empfaenger ein

uint8\_t macAddress1[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

uint8\_t macAddress2[] = {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 \*reciverMacAddress, uint8\_t status) {

if (status == 0){

Serial.println("Erfolgreich gesendet");

} else {

Serial.println("Fehler beim senden");

}

}

// Callback when data is sent

void OnDataSent(uint8\_t \*mac\_addr, uint8\_t sendStatus) {

char macStr[18];

Serial.print("Packet to:");

snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",

mac\_addr[0], mac\_addr[1], mac\_addr[2], mac\_addr[3], mac\_addr[4], mac\_addr[5]);

Serial.print(macStr);

Serial.print(" send status: ");

if (sendStatus == 0){

Serial.println("Delivery success");

}

else{

Serial.println("Delivery fail");

}

}

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(macAddress1, ESP\_NOW\_ROLE\_SLAVE, 1, NULL, 0);

esp\_now\_add\_peer(macAddress2, 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(0, (uint8\_t \*) &myMessage, sizeof(myMessage));

// 2 Sekunden warten

delay(2000);

}

Wenn die gleiche Nachricht an mehere Geräte gesendet werden soll, kann dies mit dieser Funktion geschehen.

// Sende Nachricht ueber ESP-NOW

esp\_now\_send(0, (uint8\_t \*) &myMessage, sizeof(myMessage));

Wenn dies nicht benötigt wird, kann man auf klassische Weise (Siehe Abschnitt „Sender (verschiedene Nachrichten)“) zwei verschiedene Nachrichten an den jeweiligen ESP senden

**Verschiedene Nachrichten**

#include <ESP8266WiFi.h>

#include <espnow.h>

// Trage hier die MAC-Adressen der Empfaenger ein

uint8\_t macAddress1[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

uint8\_t macAddress2[] = {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 myMessage1;

message myMessage2;

// Wird aufgerufen wenn die Nachricht gesendet wurde

void messageSend(uint8\_t \*reciverMacAddress, uint8\_t status) {

if (status == 0){

Serial.println("Erfolgreich gesendet");

} else {

Serial.println("Fehler beim senden");

}

}

// Callback when data is sent

void OnDataSent(uint8\_t \*mac\_addr, uint8\_t sendStatus) {

char macStr[18];

Serial.print("Packet to:");

snprintf(macStr, sizeof(macStr), "%02x:%02x:%02x:%02x:%02x:%02x",

mac\_addr[0], mac\_addr[1], mac\_addr[2], mac\_addr[3], mac\_addr[4], mac\_addr[5]);

Serial.print(macStr);

Serial.print(" send status: ");

if (sendStatus == 0){

Serial.println("Delivery success");

}

else{

Serial.println("Delivery fail");

}

}

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(macAddress1, ESP\_NOW\_ROLE\_SLAVE, 1, NULL, 0);

esp\_now\_add\_peer(macAddress2, ESP\_NOW\_ROLE\_SLAVE, 1, NULL, 0);

}

void loop() {

// setze Werte

myMessage1.number = random(100); // zufaellige Zahl

myMessage2.number = random(100); // zufaellige Zahl

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

memcpy(&myMessage1.string, string, sizeof(string));

memcpy(&myMessage2.string, string, sizeof(string));

// Sende Nachricht ueber ESP-NOW

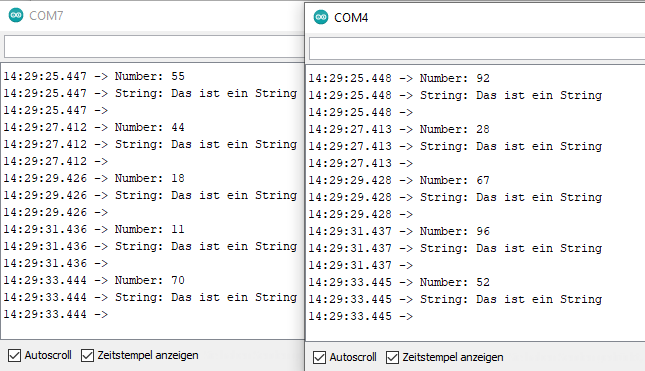
esp\_now\_send(macAddress1, (uint8\_t \*) &myMessage1, sizeof(myMessage1));

esp\_now\_send(macAddress2, (uint8\_t \*) &myMessage2, sizeof(myMessage2));

// 2 Sekunden warten

delay(2000);

}

ESP-NOW-one-to-many-eingehende-Nachrichten

Vom gleichen Sender verschiedene Nachrichten

**Empfänger**

**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 man die Arduino IDE mehrfach gestartet werden.

#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;

// Callback function that will be executed when data is received

void OnDataRecv(uint8\_t \* mac, uint8\_t \*incomingData, uint8\_t len) {

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

Serial.print("Bytes received: ");

Serial.println(len);

Serial.print("x: ");

Serial.println(myData.x);

Serial.print("y: ");

Serial.println(myData.y);

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() {

}