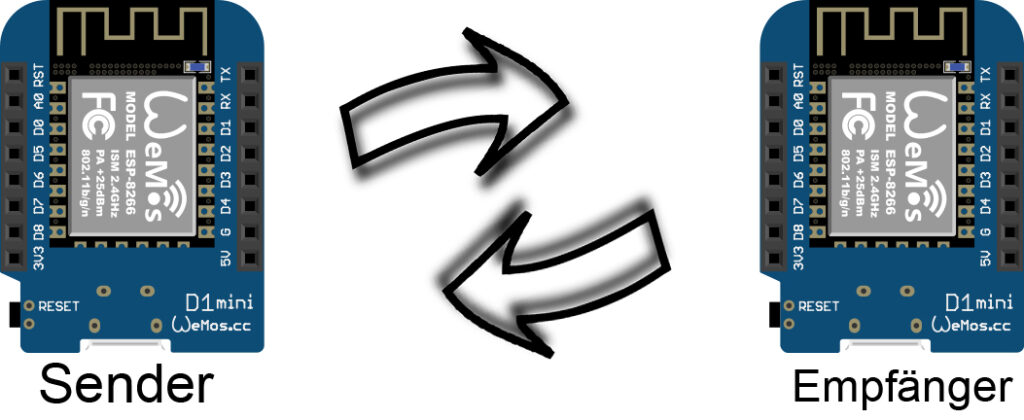
ESP-NOW ist ein Protokoll, welches die Möglichkeit bietet, mehrere Geräte zu verbinden, ohne das lokale Netzwerk nutzen zu müssen. Hiermit sind ESP8266 oder ESP32 in der Lage autark zu kommunizieren. In diesem Beitrag betrachten wir eine Two-way Kommunikation zwischen zwei ESP8266.

**ESP-NOW\_two-way**

ESP-NOW Two-way Kommunikation

In diesem Fall betrachten wir zwei ESPs, welche sich Nachrichten hin und her senden.

**Code Beispiel**

Da in diesem Beispiel gleichzeitig gesendet und empfangen wird, darf man nicht mit der „**delay()**“-Funktion arbeiten. Um das Programm „pausieren“ zu können, wird an dieser Stelle die „**millis()“-**Funktion benutzt. Auf diesem Weg kannst Du die Zeit, zwischen den Nachrichten bestimmen. Somit rechnest Du mit der vergangenen Zeit anstelle das Programm anzuhalten. Wenn Du dich zu diesem Thema mehr lesen möchtest, schau Dir unseren Beitrag **Arduino millis anstatt von delay** an.

Für die Two-Way Kommunikation benötigst Du nur ein Projekt, welches auf beiden Geräten aufgespielt werden kann.

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.

#include <ESP8266WiFi.h>

#include <espnow.h>

// Trage hier die MAC-Adresse des Empfaengers ein

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

// intervallzeit bestimmen

const long interval = 10000;

unsigned long previousMillis = 0;

// struct welches übertragen wird

typedef struct message {

long number;

char string[64];

} message;

// Nachricht welche empfangen wird

message messageIn;

// Nachricht welche gesendet wird

message messageOut;

// Wird aufgerufen wenn die Nachricht gesendet wurde

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

if (status == 0){

Serial.println("Erfolgreich gesendet");

} else {

Serial.println("Fehler beim senden");

}

}

// 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");

}

// Rolle wird auf COMBO gesetzt, da wir senden umd empfangen.

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

// 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\_COMBO, 1, NULL, 0);

// callback fuers senden

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

// callback fuers empfangen

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

}

void loop() {

unsigned long currentMillis = millis();

// Pruefe ob das Intervall vorbei ist

if (currentMillis - previousMillis >= interval) {

// save the last time you updated the DHT values

previousMillis = currentMillis;

// setze Werte

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

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

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

// Sende Nachricht ueber ESP-NOW

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

}

}

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 diese