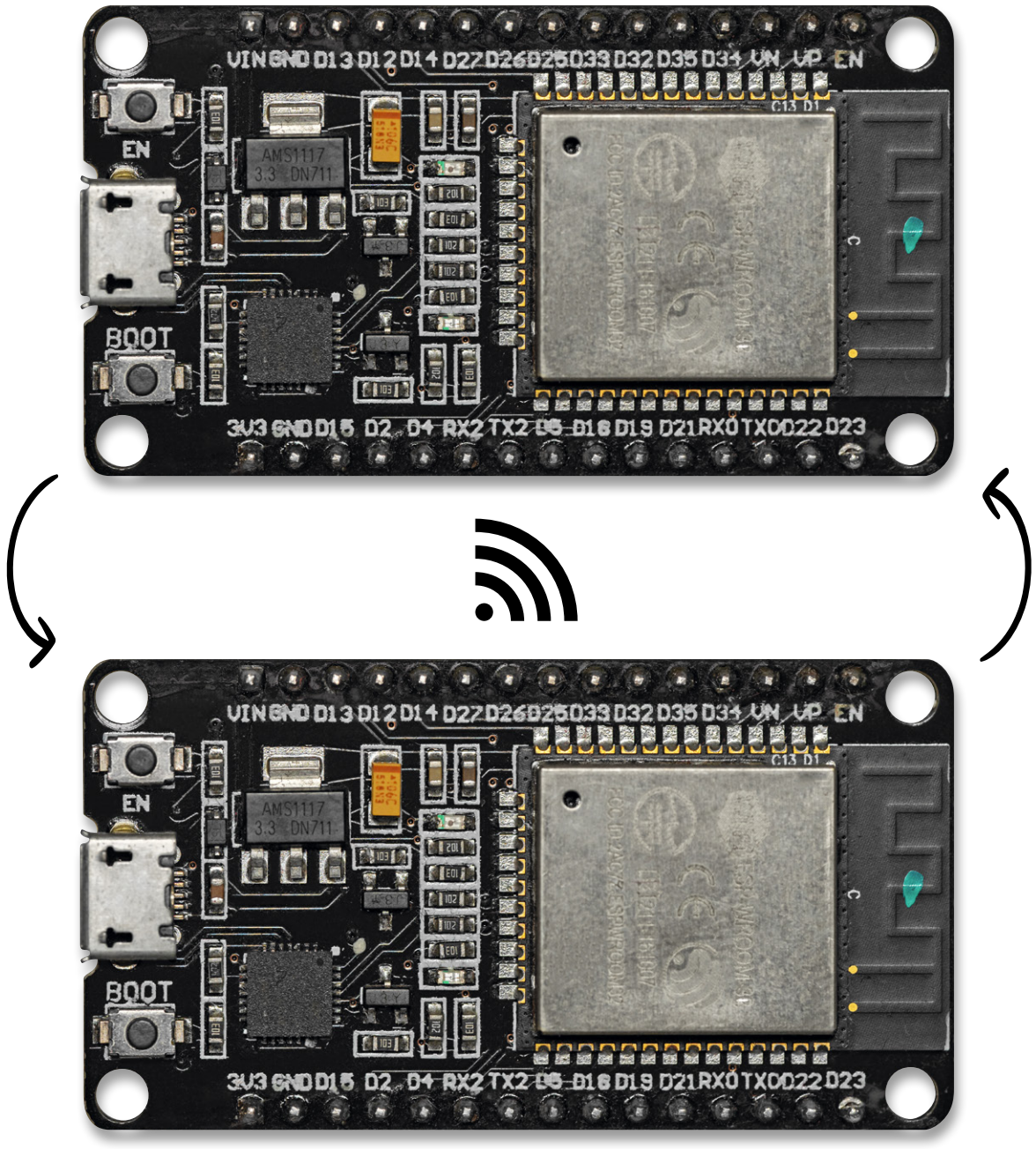
ESP-NOW

Teil 1

Drahtlose Datenübertragung ohne WLAN

Das drahtlose ESP-NOW-Kommunikationsprotokoll für die Kommunikation zwischen zwei oder mehreren ESP32-Controllern soll in dieser Artikelserie eingehender vorgestellt werden. ESP-NOW stellt ein proprietäres drahtloses Netzwerkprotokoll zur Verfügung, das von Espressif Systems, einem chinesischen Halbleiterunternehmen, entwickelt wurde. Als Hardwarebasis dienen die Controller der ESP8266- und ESP32-Familie.



ESP-NOW-Protokoll im Detail

Bei ESP-NOW handelt es sich um ein verbindungsloses Kommunikationsprotokoll, das die Übertragung kurzer Datenpakete erlaubt. Somit ermöglicht es mehreren Geräten, auf einfache Weise miteinander zu kommunizieren. Ein wichtiger Vorteil des Verfahrens ist, dass die einzelnen Teilnehmer ohne Verwendung von WiFi-Netzen miteinander kommunizieren können. Das Protokoll ähnelt zwar der klassischen WLAN-Kommunikation auf einer Frequenz von 2,4 GHz, es kommt jedoch ohne Router oder andere typische Netzwerkkomponenten aus.

Ähnlich wie bei Bluetooth ist vor dem Aufbau einer Kommunikation eine Kopplung der teilnehmenden Geräte erforderlich. Nachdem die Kopplung abgeschlossen ist, besteht eine sichere Verbindung zwischen den kommunizierenden Geräten.

Das System weist in Hinblick auf Reichweite und Stromverbrauch einige interessante Vorteile auf. Diese machen ESP-NOW für die verschiedensten Anwendungen unter anderem in den Bereichen

- Internet of Things (IoT)
- Heimautomatisierung
- Automation und Sensortechnik
- Modellbau und Robotik

zu einer spannenden und vor allem preisgünstigen Alternative zu anderen Systemen.

Mit ESP-NOW können sowohl unidirektionale als auch bidirektionale Verbindungen zwischen ESP32-basierten Geräten aufgebaut werden, auch wenn kein WiFi-Netzwerk zur Verfügung steht. Das System ermöglicht eine drahtlose Peer-to-Peer-Datenübertragung mit geringem Overhead und kleinen Datenpaketen. Es können maximal 250 Byte Daten pro Paket übertragen werden. Die Übertragung größerer Datenmengen liegt nicht im Hauptfokus dieses Protokolls.

ESP-NOW arbeitet mit einem vereinfachten Verbindungsprotokoll. Dieses ermöglicht eine deutliche Reduktion des Stromverbrauchs, da nun wesentlich weniger Zeit für die Datenübertragung benötigt wird. ESP-NOW verwendet dasselbe 2,4-GHz-Band wie die WLAN-Technologie, jedoch ist keine lokale Netzwerkanbindung erforderlich. So können auch weitab von verfügbaren WLAN-Netzen Daten drahtlos ausgetauscht werden. Umfangreiche Tests zeigten zudem, dass es nicht zu Störungen zwischen den beiden Verfahren kommt.

ESP-NOW ist ein proprietäres drahtloses Netzwerkprotokoll mit besonders stromsparender, sicherer und kostengünstiger Kommunikation. Dabei ermöglicht es nicht nur eine Punkt-zu-Punkt-Verbindung, sondern den kompletten Aufbau eines Kommunikationsnetzes.

Umfangreiche Funktionalität

ESP-NOW unterstützt die folgenden Funktionen:

- Bis zu 250 Byte Nutzdaten
- Verschlüsselte und unverschlüsselte Kommunikation
- Rückmeldung zum Erfolg oder Misserfolg der Übertragung
- Hohe Reichweite von über 100 m (Details dazu im Abschnitt Reichweitentest)

Die ESP-NOW-Technologie hat aber auch einige Einschränkungen:

- Im Normalmodus werden maximal zehn verschlüsselte Kommunikationsteilnehmer unterstützt.
- Die Gesamtzahl aller Teilnehmer sollte weniger als 20 betragen.

ESP-NOW ist damit ein schnelles Kommunikationsprotokoll, mit dem kleinere Datenmengen zwischen ESP32-Boards unkompliziert und über größere Entfernungen hinweg über Funk ausgetauscht werden können.

Dabei ist das System sehr vielseitig. So kann man in verschiedenen Set-ups sowohl eine Einweg- als auch eine Zwei-Wege-Kommunikation aufbauen. Die Kommunikation ist dabei nicht auf ein bestimmtes ESP-Board beschränkt. Praktisch alle Board-Varianten wie

- Pico Kit (V4)
- Joy-IT NodeMCU
- NodeESP
- MakePython ESP Development Board

und viele andere können problemlos miteinander kommunizieren (siehe Bild 1).

Auch alle anderen auf dem ESP32 basierenden Boards sollten ohne Probleme in ESP-NOW-Systeme eingebunden werden können.

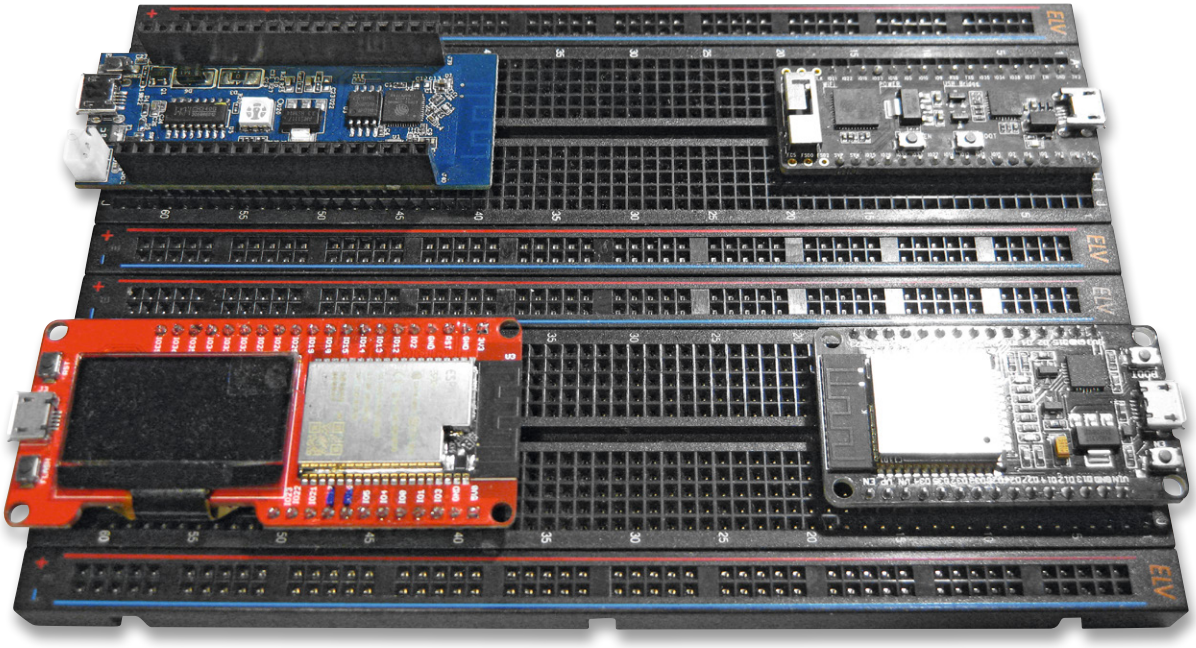


Bild 1: Verschiedene ESP32-Boards können drahtlos miteinander kommunizieren.

Programmierung mit der Arduino IDE

Prinzipiell ist ESP-NOW auch in Python verfügbar. Allerdings ist diese Variante noch im Aufbau begriffen, und die zugehörigen Tools sind noch nicht voll entwickelt. Die auf C basierenden Routinen sind dagegen weitverbreitet und erfreuen sich großer Beliebtheit.

Wenn man sich für die Arduino IDE entschieden hat, stellt sich aktuell die Frage, ob man die Version 1.x oder 2.x verwenden sollte. Es wird immer wieder berichtet, dass die Arduino IDE 2.0.3 sehr langsam läuft. Allein der Start-up kann mehrere Minuten in Anspruch nehmen. Zudem fehlen viele, inzwischen selbstverständliche Features wie etwa das Ändern der Font-Größe über das Mausrad etc. Für ein zügiges Arbeiten sollte also vorläufig noch die Version 1.x verwendet werden, bis diese „Kinderkrankheiten" behoben sind.

In der Arduino IDE sind zum Einbinden von ESP-Boards die folgenden Schritte erforderlich:

Über Datei → Voreinstellungen wird in das Feld „Zusätzliche Boardverwalter-URLs" der folgende Link eingegeben:

https://dl.espressif.com/dl/package_esp32_index.json

Wenn bereits andere URLs vorhanden sind, können diese URLs mit Kommas getrennt werden.

Danach wird über Werkzeuge → Boards → Boardverwalter die Board-Verwaltungsübersicht geöffnet. Hier kann man nach „ESP32" suchen und dann auf die Schaltfläche „Installieren" für „ESP32 by Espressif Systems" klicken. Das Board ist nach ein paar Sekunden installiert.

Zum Testen der Installation schließt man das ESP32-Board an den Host-Computer an. Dann sind bei geöffneter Arduino IDE die folgenden Schritte auszuführen:

1. Board im Menü Werkzeuge → Board auswählen (z. B. „ESP32 DEVKIT V4")
2. Port auswählen
3. Beispiel unter Datei → Beispiele → WiFi (ESP32) → WiFiScan öffnen
4. Der Sketch öffnet sich in der Arduino IDE. Dieser wird über „Hochladen" auf das Board übertragen.
5. Wenn alles wie erwartet gelaufen ist, wird „Hochladen abgeschlossen" angezeigt.
6. Im Serial Monitor (Baudrate von 115200) werden nun die verfügbaren Netzwerke in der Nähe des ESP32 angezeigt (Bild 2).

Damit steht das System für den Einsatz zur Verfügung. Zudem wurde auch bereits die Funktion des 2,4-GHz-Moduls auf dem Board überprüft. Die WLAN-Netze dienen hier lediglich zum Test des Boards und des 2,4-GHz-Funksystems. Für das ESP-NOW-System werden diese Netze nicht benötigt.

Kommunikation in allen Varianten

Das ESP-NOW-System zeichnet sich durch seine große Flexibilität in Hinblick auf die möglichen Kommunikationsarten aus. Die folgenden Varianten sind verfügbar:

- Einweg-Kommunikation:
  - Ein ESP32 sendet Daten an mehrere andere.
  - Ein ESP32 empfängt Daten von mehreren anderen.
- Zwei-Wege-Kommunikation
- Kommunikation zwischen mehreren ESP32-Boards
- Bidirektionale, voll vernetzte Kommunikation

Bei der Einweg-Kommunikation kann ein ESP32-Board beispielsweise Sensordaten drahtlos an ein anderes ESP-Board übermitteln. Diese Konfiguration ist vergleichsweise einfach zu implementieren und eignet sich bestens für einfache Einstiegsanwendungen. Typische Szenarien sind die Übertragung von Temperaturmesswerten oder auch von Schaltbefehlen für GPIO-Pins.

Hinweis: Da die Bezeichnungen „Master" und „Slave" auch in der Technik nicht mehr verwendet werden sollen, werden im Folgenden stattdessen die Begriffe „Primary" (für Master) und „Secondary" (für Slave) verwendet.

Zunächst einmal soll ein Primary-ESP32 Daten an einen Secondary-ESP senden. Diese Konfiguration ist bestens geeignet, um eine universelle Fernbedienung zu realisieren. Man kann so ESP32-Boards im Haus mit einem ESP32-Handsender steuern. Diese Variante eignet sich zudem bestens als Einstiegsprojekt, da man damit auf einfache Weise die Funktion des ESP-NOW-Systems kennenlernen kann.

Später kommen auch die anderen Versionen zum Einsatz, u. a.:

- **Mehrere Sender – ein Empfänger:**
  Mehrere ESP-NOW-Primary-Boards senden Daten an ein Secondary-Board.
  Diese Konfiguration ist ideal, wenn Daten von mehreren Sensorknoten auf einem ESP32-Board gesammelt werden sollen. Dieses kann dann z. B. auch als Webserver konfiguriert werden, um die Daten aller anderen Boards anzuzeigen.
- **Zwei-Wege-Kommunikation:**
  Bei ESP-NOW kann jedes Board gleichzeitig Sender und Empfänger sein. So kann eine bidirektionale Kommunikation zwischen mehreren Boards hergestellt werden.
- **Mehrere Empfänger – ein Sender:**
  Hier können z. B. mit einem zentralen Handsender mehrere Geräte simultan gesteuert werden.

Bild 2: Verfügbare WLAN-Netzwerke in der Nähe des ESP32-Boards

Schließlich kann man weitere Boards hinzufügen und ein komplettes Netzwerk aufbauen. Dabei können alle ESP32-Boards miteinander kommunizieren und Daten austauschen.

MAC-Adresse des Boards abrufen und ändern

Um über ESP-NOW zu kommunizieren, muss die MAC-Adresse (MAC: Media Access Control) des ESP32-Empfängers bekannt sein. So kann dem Sender mitgeteilt werden, an welches Gerät die Daten zu senden sind.

Jeder ESP32-Chip hat seine eigene eindeutige MAC-Adresse. Diese kann genutzt werden, um das Board, an welches Daten gesendet werden sollen, zu identifizieren. Die MAC-Adresse stellt eine eindeutige Hardwarekennung dar, die jedes Gerät in einem Netzwerk identifiziert. MAC-Adressen bestehen aus sechs Gruppen von zwei hexadezimalen Ziffern, die durch Doppelpunkte getrennt sind, zum Beispiel:

E2:EE:B7:3F:DE:A1

MAC-Adressen werden üblicherweise vom Hersteller vergeben. Man kann einem Controller-Board aber auch eine benutzerdefinierte Adresse zuweisen. Jedes Mal, wenn das Board zurückgesetzt wird, lädt es jedoch wieder seine ursprüngliche MAC-Adresse. Wenn die Möglichkeit benutzerdefinierter Adressen genutzt werden soll, muss man also die gewünschte MAC-Adresse in das Programm einbinden. In einigen Anwendungen kann es nützlich sein, einem Board eine benutzerdefinierte MAC-Adresse zuzuweisen. So kann man beispielsweise eine systematische MAC-Folge anlegen, die jedem Board einen bestimmten „sprechenden Code" zuweist.

Das folgende Programm legt eine benutzerdefinierte MAC-Adresse für das ESP32-Board fest:

```
// ESP_now_set_MAC.ino
// ESP32 @ IDE 1.8.16

#include <WiFi.h>
#include <esp_wifi.h>

uint8_t newMACAddress[] = {0x32, 0x32, 0x32, 0xFF, 0xFF, 0xFF};

void setup()
{ Serial.begin(115200);
  Serial.println();
  WiFi.mode(WIFI_STA);
  Serial.print("Old ESP32 Board MAC Address:  ");  Serial.println(WiFi.macAddress());
  esp_wifi_set_mac(WIFI_IF_STA, &newMACAddress[0]);
  Serial.print("New ESP32 Board MAC Address:  ");  Serial.println(WiFi.macAddress());
}

void loop(){}
```

Die gewünschte MAC-Adresse wird in der folgenden Zeile festgelegt:

uint8_t newMACAddress[] = {0x32, 0x32, 0x32, 0xFF, 0xFF, 0xFF};

Nach dem Hochladen des Codes wird nach einem Reboot (Drücken der „Boot"-Taste) die alte und die neue MAC im Serial Monitor ausgegeben (Bild 3).

Um die MAC-Adresse eines bestimmten Boards auszulesen, kann das folgende Programm genutzt werden:

```
// ESP_now_get_MAC.ino
// ESP32 @ IDE 1.8.16

#include "WiFi.h"

void setup()
{ Serial.begin(115200); WiFi.mode(WIFI_MODE_STA);
}

void loop()
{ Serial.print("MAC Address: \t"); Serial.println(WiFi.macAddress());
  delay(1000);
}
```

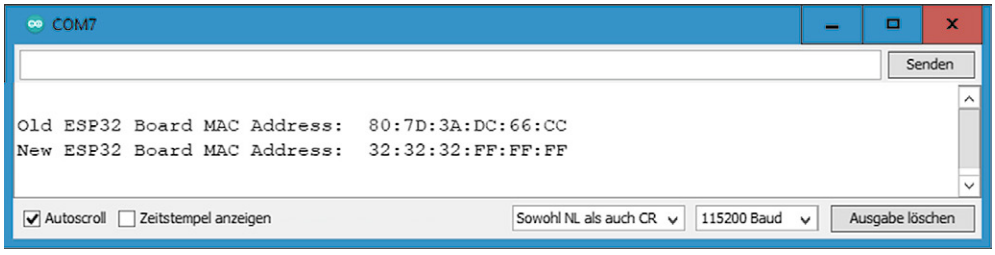


Bild 3: Alte und neue MAC-Adresse

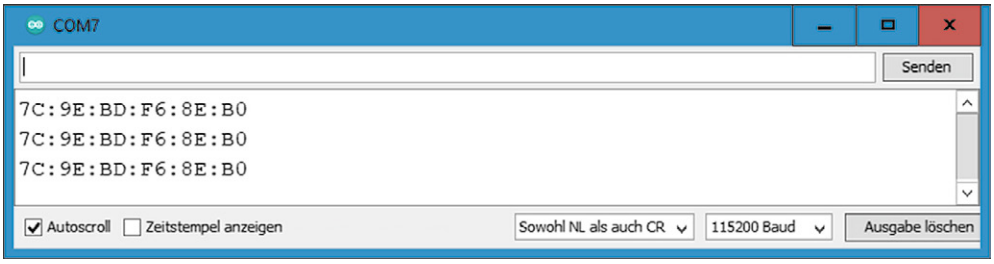


Bild 4: MAC-Adresse im seriellen Monitor

	A	B	C	D	E
1	**Version**	**No**	**MAC**	**COM**	**ESP now function**
2					
3	NodeESP	1	[illegible]	4	Rx01
4	NodeESP	2	[illegible]	5	Rx00
5	Franzis NodeESP	3	[illegible]	7	Tx01
6	MakePython ESP32	4	[illegible]	11	
7	NodeMCU	5			
8	NodeMCU	6	[illegible]		
9	ESP32-PICO-KIT_V4	7	[illegible]	16	
10	ESP32-PICO-KIT_V4	8			
11					
12					

Bild 5: Board-Übersicht

Nach dem Hochladen des Codes wird der Serial Monitor mit einer Baudrate von 115200 geöffnet. Anschließend wird die MAC-Adresse fortlaufend ausgegeben (Bild 4).

Wenn die MAC-Adresse eines Boards erfolgreich ausgelesen wurde, wird sie am besten in einer eignen Spreadsheet-Datei (z. B. LibreOffice oder Excel) abgespeichert. Idealerweise legt man gleich eine Übersichtsdatei mit den MAC-Adressen aller verfügbaren Boards an, da in späteren Anwendungen mit mehr als zwei Boards ansonsten schnell der Überblick verloren geht. Zudem werden die Boards am besten durchnummeriert, sodass jederzeit eine schnelle und eindeutige Zuordnung von MAC-Adresse und Board möglich ist. Auch die dem Board aktuell zugewiesene COM-Schnittstelle kann hier eingetragen werden. Alternativ kann die Adresse auch direkt auf den Aufklebern der Boards notiert werden (Bild 5).

Punkt-zu-Punkt-Kommunikation

Der einfachste Weg, zwei Geräte miteinander drahtlos zu verbinden, ist die Punkt-zu-Punkt-Kommunikation. In einem ersten Projekt soll daher lediglich eine Nachricht von einem ESP32 an einen anderen gesendet werden. Dabei ist ein ESP32 der Sender und der andere ESP32 der Empfänger. Gesendet werden die Daten in Form des Datentyps „Structure". Dieser Datentyp ist bei ESP-NOW-Anwendungen von zentraler Bedeutung. Da er andererseits wenig geläufig ist, soll er hier kurz erläutert werden.

Der Variablentyp „Structure" (struct) ist ein benutzerdefinierter, zusammengesetzter Typ. Er besteht aus Feldern, die selbst wiederum unterschiedliche Typen haben können. Strukturen sind also eine Sammlung und Beschreibung verschiedener Daten. Eine Struktur ist sozusagen die Vorlage für die Definition der konkreten Daten.

Ein klassisches Beispiel sind Informationen zu Städten, die mithilfe unterschiedlicher Datentypen zusammengefasst werden:

struct Stadt {String Name; String KFZ_Kennzeichen; float MioEinwohner }

Die Daten können dann so eingegeben werden:

Stadt Berlin = {„Berlin", „B", 3.64};
Stadt Hamburg = {„Hamburg", „HH", 1.84 };
Stadt Muenchen = { „München", „M", 1.82 };
Stadt Leer = { „Leer", „LER", 0.034 };

Die Zuordnung der verschiedenen Variablen einer structure erfolgt über die Punkt-Schreibweise:

- Stadtname.Name → String Name
- Stadtname.KFZ_Kennzeichen → String KFZ-Kennzeichen
- StadtName.MioEinwohner → float Einwohner

Über eine Funktion können die Daten formatiert ausgegeben werden (Bild 6).

Nachdem Aufbau und Anwendung des Datentyps „stucture" nun prinzipiell klar sein sollten, kann die structure in den ESP-NOW-Anwendungen entsprechend eingesetzt werden.

In der ersten ESP-NOW-Anwendung enthält die structure-Variable die Datentypen

- char
- int
- float
- boolean

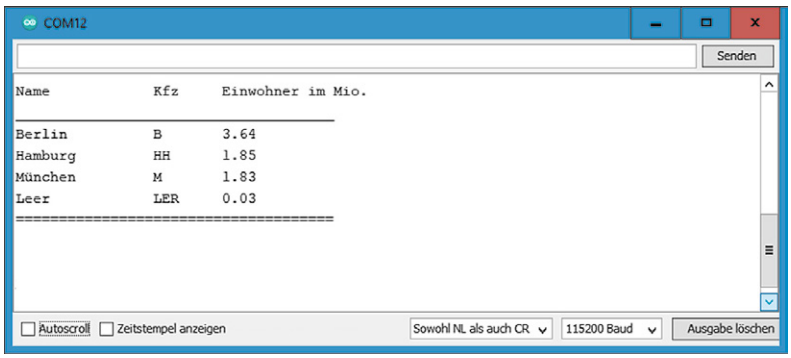


Bild 6: Ausgabe strukturierter Daten

In späteren Anwendungen kann diese „structure" dann leicht an die erforderlichen Projektaufgaben angepasst werden, um verschiedene Daten wie z. B. Sensormesswerte, boolesche Informationen oder Ein-/Ausschaltbefehle zu übermitteln.

Im Folgenden wird der Sender mit Tx01 und der Empfänger mit Rx01 angesprochen. Der Sketch auf der Senderseite soll die folgenden Aufgaben übernehmen:

- ESP-NOW initialisieren
- Registrieren einer „Callback"-Funktion beim Senden von Daten
- Hinzufügen eines „Peers" als Kommunikationspartner
- Versenden eines Datenpakets an den Peer

Auf der Empfängerseite müssen folgende Aufgaben erledigt werden:

- ESP-NOW initialisieren
- Ausführen einer Rückruffunktion, welche bestätigt, dass eine Nachricht empfangen wurde
- Speichern der Informationen, um die gewünschte Aufgabe damit auszuführen

Für die Kommunikation stehen diese Funktionen zur Verfügung:

- sp_now_init(): initialisiert ESP-NOW
- esp_now_add_peer(): Diese Funktion muss aufgerufen werden, um Geräte zu koppeln. Als Argument ist die MAC-Adresse des zu koppelnden Geräts („peer") zu übergeben.
- esp_now_send() : sendet Daten

ESP-NOW arbeitet mit Callback-Funktionen. Diese werden einerseits aufgerufen, wenn ein Gerät eine Nachricht versendet, andererseits wird über die Callback-Funktion auch eine Empfangsbestätigung zum Sender zurückgeschickt. Damit steht auf der Senderseite die Information zur Verfügung, ob eine Nachricht erfolgreich zugestellt wurde oder die Übertragung fehlgeschlagen ist. Die hierfür verwendeten Funktionen sind:

- esp_now_register_send_cb() : Registrieren einer Rückruffunktion. Diese wird beim Senden von Daten aufgerufen. Diese Funktion liefert eine Rückinformation darüber, ob eine Zustellung erfolgreich war oder nicht.
- esp_now_register_rcv_cb() : das Gegenstück auf der Empfängerseite. Beim erfolgreichen Empfang von Daten wird eine Bestätigung ausgegeben.

Daten senden

Das folgende Programm zeigt den Code für das ESP32-Sender-Board (Tx01). Vor dem Hochladen auf das ESP32-Board ist in der Zeile **uint8_t broadcastAddress[ ] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};** die oben ausgelesene MAC-Adresse des Empfänger-Boards einzutragen, z. B. **// uint8_t broadcastAddress[ ] = {0x0D, 0x3F, 0x12, 0x5E, 0x3F, 0xF1};**

```
// ESP_now_SEND.ino
// ESP32 @ IDE 1.8.16

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

// uint8_t broadcastAddress[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}; // RECEIVER MAC Address
uint8_t broadcastAddress[] = {0x01, 0x01, 0x03, 0xAB, 0xCD, 0xEF};

int MessageNumber=1;

typedef struct struct_message
{ char text[32];  int number;  float pi;  bool logic;} struct_message;
struct_message myData; esp_now_peer_info_t peerInfo;

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status)
{ Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery Success" : "Delivery Fail"");}

void setup()
{ Serial.begin(115200);  WiFi.mode(WIFI_STA);
  if (esp_now_init() != ESP_OK) { Serial.println("Error initializing ESP-NOW"); return; }
  esp_now_register_send_cb(OnDataSent);
  memcpy(peerInfo.peer_addr, broadcastAddress, 6); // six bytes for MAC
  peerInfo.channel = 0; peerInfo.encrypt = false;
  if (esp_now_add_peer(&peerInfo) != ESP_OK) { Serial.println("Failed to add peer"); return; }
}

void loop()
{ strcpy(myData.text, "Hello from TX01");
          myData.number=MessageNumber;
          myData.pi=3.14;
          myData.logic=false;
  esp_err_t result=esp_now_send(broadcastAddress,(uint8_t *) &myData, sizeof(myData));
  Serial.print(result==ESP_OK ? "Data Sent \t" : "Send Fail \t");
  delay(1000);
  MessageNumber++;
}
```

Das Programm lädt zunächst die beiden erforderlichen Bibliotheken:

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

Nach dem Zuweisen der Empfänger-MAC-Adresse wird die Datenstruktur für die zu sendenden Daten erstellt. Diese soll vier verschiedene Variablentypen enthalten:

```
typedef struct struct_message
{ char text[32];  int number;  float pi;  bool logic;} struct_message;
```

Bei Bedarf können die Variablen beliebig verändert oder ergänzt werden. Dann wird eine neue Variable „myData" vom Typ struct_message erstellt. Diese nimmt die später zu übertragenden Werte auf. In der Variablen „peerInfo" werden die Informationen über den Peer gespeichert.

Die „OnDataSent()"-Funktion ist die Callback-Rountine. Sie wird ausgeführt, sobald eine Nachricht gesendet wird. Die Funktion meldet, ob die Nachricht erfolgreich zugestellt wurde oder nicht.

Im „setup()" werden der serielle Monitor und die WiFi-Funktion initialisiert. Anschließend wird das ESP-NOW-System gestartet. Danach erfolgt die Kopplung mit dem Empfänger („peer")

In der Hauptschleife sollen einmal pro Sekunde die Beispieldaten

```
strcpy(myData.text, "Hello from TX01");
        myData.number=MessageNumber;
        myData.pi=3.14159;
        myData.logic=false;
    myData.d = false;
```

über ESP-NOW versendet werden.

Abschließend wird überprüft, ob der Sendevorgang erfolgreich war. In diesem Fall wird die Nachricht

„Data Sent Delivery Success"

im seriellen Monitor ausgegeben. Andernfalls erscheint die Fehlermeldung

„Data Sent Delivery Fail"

Im Programm wird der sogenannte tertiäre Operator verwendet. Dieser ist lediglich eine abgekürzte Schreibweise für die if/else-Anweisung. So kann man statt

if (a>b) then Serial.println („a ist größer als b")
else Serial.println („a ist kleiner oder gleich b")

kürzer schreiben:

Serial.println((a > b)?„a ist größer als b":„a ist kleiner oder gleich b");

In den Klammern erfolgt der Vergleich, darauf folgt ein Fragezeichen („?") und der Teil, der ausgeführt wird, wenn die Bedingung wahr ist. Mit dem Doppelpunkt („:") wird der Alternativ-Teil („else") eingeleitet.

Datenempfang

Der Empfänger-Sketch fällt deutlich kürzer aus:

```
// ESP_now_RECEIVE.ino
// ESP32 @ IDE 1.8.16

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

typedef struct struct_message
{ char text[32];  int number;  float pi;  bool logic;} struct_message;
struct_message myData;

void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int len)
{ memcpy(&myData, incomingData, sizeof(myData));
  Serial.print("Bytes received: ");  Serial.println(len);
  Serial.print("Text: ");            Serial.println(myData.text);
  Serial.print("Message Number: ");  Serial.println(myData.number);
  Serial.print("Pi: ");              Serial.println(myData.pi, 2);
  Serial.print("Bool: ");            Serial.println(myData.logic);
  Serial.println();
}

void setup()
{ Serial.begin(115200); WiFi.mode(WIFI_STA);
  if (esp_now_init() != ESP_OK) { Serial.println("Error initializing ESP-NOW"); return; }
  esp_now_register_recv_cb(OnDataRecv);
}

void loop() {}
```

Die Datenstruktur muss dem Aufbau der Sender-Daten entsprechen:

```
typedef struct struct_message
{ char a[32]; int b; float c; bool d;
} struct_message;
```

Die Callback-Funktion wird aufgerufen, sobald der Empfänger Daten über ESP-NOW erhält. Der Inhalt von „incomingData“ wird umgehend in die Variable „myData“ kopiert.

Damit enthält die myData-Struktur die vom ESP32-Sender gesendeten Werte.

Um beispielsweise auf die Variable a zuzugreifen, kann nun einfach myData.a ausgewertet werden.

Im Beispiel werden die empfangenen Daten auf die serielle Schnittstelle ausgegeben. Da das Programm über die Empfangsroutine gesteuert wird, kann die Hauptschleife leer bleiben. Der serielle Monitor auf der Empfängerseite zeigt die eingehenden Daten (Bild 7).

Hinweis: Alle hier vorgestellten Programme und Sketche sind als Download-Paket zu diesem Beitrag verfügbar (siehe Kasten „Weitere Infos“).

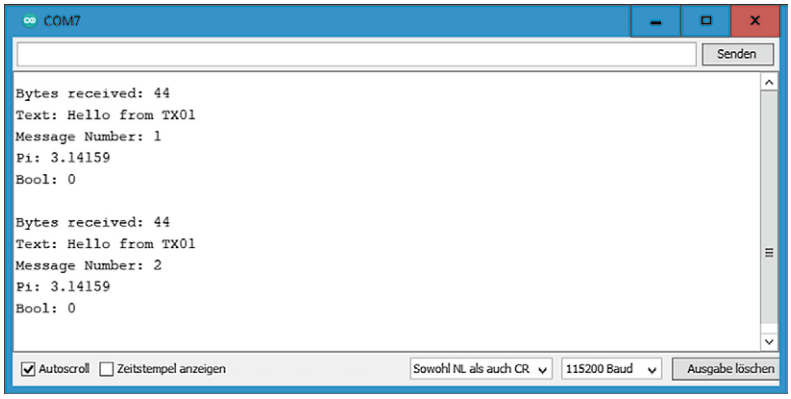


Bild 7: Datenempfang im seriellen Monitor

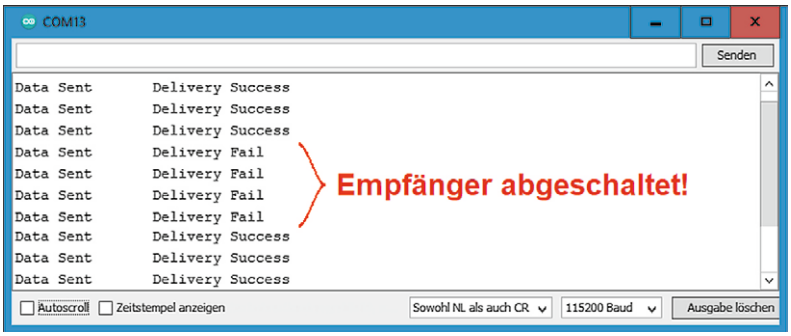


Bild 8: Das Sendeprotokoll zeigt den Ausfall des Empfängers.

Verbindungs- und Reichweitentests

Nachdem nun eine erste Verbindung erfolgreich aufgebaut wurde, kann man verschiedene Tests vornehmen. Zunächst kann man die Funktion der Callback-Routine prüfen. Hierzu trennt man den Empfänger von der Stromversorgung. Durch die Rückmeldefunktion wird der Sender darüber informiert, dass die Daten nicht zugestellt werden konnten. Im Protokoll des Senders wird dies entsprechend dokumentiert (Bild 8).

Sobald der Empfänger wieder in Betrieb genommen wird, können die Daten wieder korrekt zugestellt werden. Dies wird im Senderprotokoll durch die „Delivery Success“-Meldung bestätigt.

Als Nächstes kann man die Reichweite des Systems testen. Mit der Angabe von Reichweiten bei Funkverbindungen verhält es sich ähnlich wie bei der Fahrleistung von Elektroautos. Sofern die Zahlen überhaupt irgendeinen Bezug zur Realität haben, wurden sie unter idealsten Bedingungen gemessen. Bei den Reichweiten-Angaben zu ESP-NOW-Systemen verhält es sich ähnlich. Im Internet kursieren Angaben von bis zu über 300 m.

Um die tatsächliche Reichweite zu überprüfen, kann man den Empfänger auf Batteriebetrieb umschalten. Eine handelsübliche Powerbank ist bestens für diese Aufgabe geeignet (Bild 9).

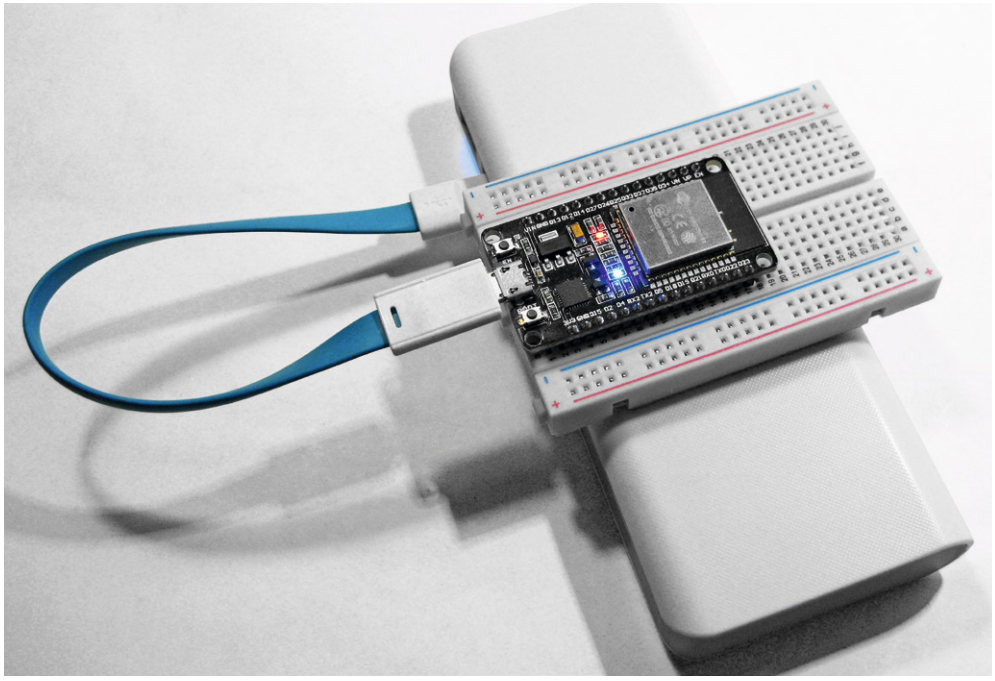


Bild 9: ESP-NOW-Empfänger mit Powerbank

Nachdem man sich von der korrekten Funktion der Übertragung überzeugt hat, kann man die Entfernung zwischen Sender und Empfänger vergrößern. Der mit dem PC verbundene Sender bleibt an Ort und Stelle. Der serielle Monitor zeigt fortlaufend die erfolgreiche Zustellung von Daten an. Erst ab einer bestimmten Distanz treten erste „Delivery Fail"-Meldungen auf. Wird die Entfernung weiter vergrößert, bricht die Verbindung schließlich komplett ab. Wird die Distanz erneut verkleinert, nimmt die Übertragungsstrecke automatisch wieder ihre Arbeit auf. Umfangreiche Versuche ergaben diese Reichweiten:

im Gebäude	>10 m, über bis zu zwei Mauern oder Betondecken hinweg
im Freien	erste „Delivery Fail"-Meldungen ab etwa 100 m
Komplettausfall der Verbindung	ab ca. 150 m

Auch wenn diese Werte deutlich geringer ausfallen als die im Internet kursierenden Zahlen zeigt sich dennoch, dass das ESP-NOW-System erstaunliche Reichweiten erzielt. Dies gilt besonders, wenn man die geringe Größe der On-Board-Antennen mit in Betracht zieht.

Die Reichweite von ESP-NOW hängt von einer Reihe von Faktoren wie dem Antennendesign oder der Betriebsumgebung ab. Im Allgemeinen hat ESP-NOW eine zuverlässige Reichweite von bis zu 100 Metern. Diese Reichweiten sind jedoch ungefähre Angaben und können je nach den spezifischen Umständen erheblich variieren. Zu den Faktoren, die die Reichweite von ESP-NOW beeinflussen können, gehören:

Hochfrequenzstörungen: ESP-NOW arbeitet im 2,4-GHz-Frequenzband, das mit anderen drahtlosen Technologien geteilt wird. Dies bedeutet, dass möglicherweise Reichweiten zurückgehen, auch wenn keine direkten Störungen durch andere Geräte auftreten.

Betriebsumgebung: Die Reichweite von ESP-NOW kann durch physische Hindernisse wie Wände, Decken und Böden sowie durch Umweltfaktoren wie Temperatur und Luftfeuchtigkeit beeinträchtigt werden.

Antennendesign: Die Reichweite von ESP-NOW kann durch Art und Design der verwendeten Antenne beeinflusst werden. Beispielsweise kann die Verwendung einer Richtantenne die Reichweite von ESP-NOW erhöhen. Einige ESP32-Boards erlauben den Anschluss externer Antennen, welche die Reichweite weiter verbessern.

Bei der Auswahl eines drahtlosen Kommunikationsprotokolls wie ESP-NOW ist darauf zu achten, dass die spezifischen Reichweiten-Anforderungen der Anwendung entsprechen. Im Bedarfsfall sind externe Antennensysteme eine nützliche Erweiterung.

Fazit und Ausblick

In diesem Beitrag wurden die Grundlagen des ESP-NOW-Kommunikationsprotokolls und seine verschiedenen Funktionen vorgestellt. In einem ersten Anwendungsbeispiel wurden Datenpakete von einem ESP32-Board zu einem anderen gesendet.

Dasselbe Verfahren ist auch verwendbar, um Anwendungsdaten wie etwa Sensormesswerte zu übertragen. Die drahtlose Steuerung von Ausgangspins eines ESP32-Boards ist so ebenfalls möglich. Im nächsten Beitrag werden diese Varianten näher betrachtet.

Neben einer bidirektionalen Messwertübertragung werden Praxisprojekte wie etwa eine drahtlose Türklingel oder eine ESP-NOW-basierte Fernsteuerung für Modellbauzwecke vorgestellt. ELV

Material	Artikel-Nr.
2x ESP-32-Board, z. B. Joy-IT Entwicklungsplatine NodeMCU mit ESP32	145164

Weitere Infos

Download-Paket: Artikel-Nr. 253474