ESP-NOW

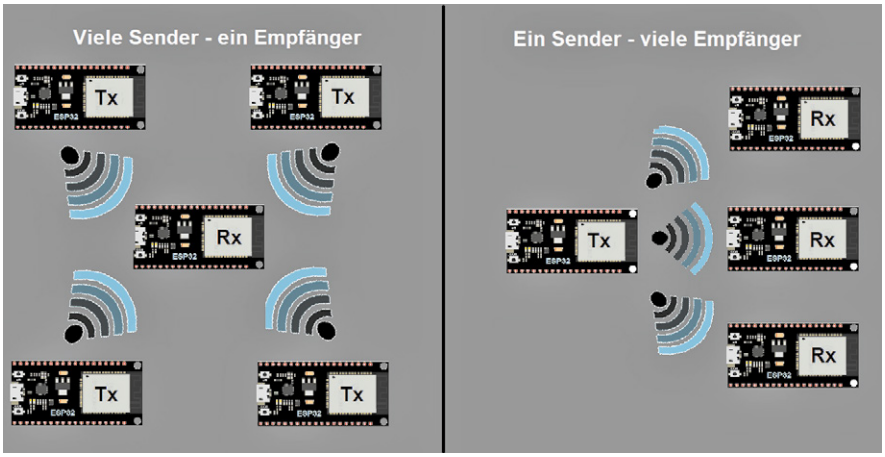
Teil 3

Drahtlose Netzwerke

Nachdem im letzten Beitrag ausführlich die Möglichkeiten des drahtlosen Datenaustauschs zwischen zwei ESP32-Boards behandelt wurden, soll dieser Artikel nun zeigen, wie das ESP-NOW-System verwendet werden kann, um Datennetze mit mehreren Stationen zu realisieren. Mit einem zentralen Handsender können auf diese Weise beispielsweise mehrere Geräte gesteuert werden. Es ist auch möglich, Daten von vielen Sendern zu empfangen, was das Sammeln von Daten aus einer Vielzahl von Messstationen ermöglicht.

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

Bild 1: Projektüberblick



Hardware

Um die Projekte und Anwendungen dieses Artikels komplett durchzuarbeiten, sind drei ESP32-Boards erforderlich. Dazu werden die folgenden Komponenten benötigt (siehe „Material"):

- zwei LEDs mit Vorwiderständen (ca. 150 Ohm)
- zwei Taster
- zwei Potenziometer

Das Diagramm in Bild 1 zeigt einen allgemeinen Überblick über die Projekte, die erstellt werden sollen.

ESP32 als Datensammler: Viele Sender – ein Empfänger

Bislang wurden klassische Zwei-Punkt-Verbindungen aufgebaut, die ähnlich wie eine serielle Schnittstelle Daten in beide Richtungen senden können. Das ESP-Protokoll ist jedoch deutlich flexibler und ermöglicht ähnlich wie Server und Router im Internet den Aufbau eines echten Datennetzwerks.

In einem ersten Schritt soll ein einfaches Netzwerk aufgebaut werden, in dem mehrere Sender ihre Daten an einen Empfänger übermitteln. Diese Konfiguration eignet sich bestens, wenn eine Vielzahl von Daten von mehreren Sensorknoten auf einem „Zentralknoten" gesammelt werden sollen. Das vorgestellte Beispiel nutzt zunächst nur zwei Sender und einen Empfänger. Eine Erweiterung auf größere Systeme ist jedoch problemlos möglich.

Die Anwendungsmöglichkeiten sind zahlreich und reichen von Klimasensoren in verschiedenen Räumen eines Hauses über optische Überwachungssysteme bis hin zu Energieeinsparungssystemen wie z. B.:

- Erfassung von Raumklimadaten in verschiedenen Räumen (Wohn-, Ess-, Schlafzimmer, Bad oder Küche zum Zwecke der Energieeinsparung)
- Durch die große Reichweite können auch Messwerte aus nahegelegenen Gebäuden wie Garagen, Gewächs- oder Gartenhäusern erfasst werden
- Überprüfung der Beleuchtungsverhältnisse in Häusern und Wohnungen

Bild 2 zeigt den prinzipiellen Aufbau.

Ein ESP32-Board fungiert als Empfänger/Secondary (früher auch als „Slave" bezeichnet), während zwei oder mehrere ESP32-Boards als Sender/Primary (ehemals „Master") dienen. Der Aufbau wurde erfolgreich mit bis zu acht ESP32-Senderboards getestet, ohne dass dabei Probleme auftraten. Es ist daher ohne Weiteres möglich, bei Bedarf zusätzliche Boards hinzuzufügen.

Analog zu den früheren Projekten empfängt die Senderplatine wieder eine Bestätigungsnachricht, die anzeigt, ob die Daten erfolgreich zugestellt wurden oder nicht. Zudem erhält das Empfängerboard von allen Absendern einen Code, über den das entsprechende Board identifiziert werden kann, das die Nachricht gesendet hat.

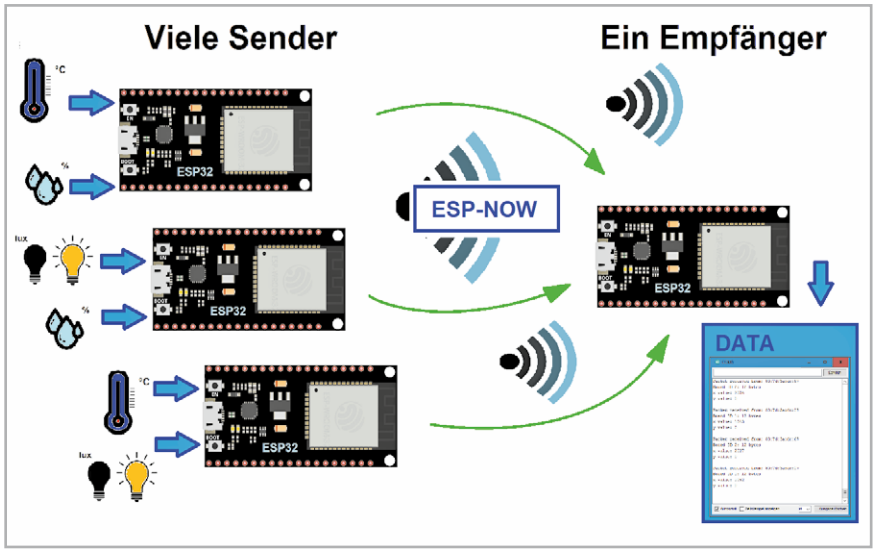


Bild 2: Viele ESP-Sender übertragen Daten an einen Empfänger.

Im folgenden Beispiel sollen beispielhaft Helligkeitswerte zwischen den Boards ausgetauscht werden. Alternativ können auch Potenziometerspannungen verwendet werden. Der Aufbau kann auch problemlos für die Erfassung anderer analoger Sensormesswerte wie zum Beispiel NTCs für Temperaturmessungen umgerüstet werden.

Hinweis: In der ESP-NOW-Dokumentation gibt es so etwas wie „Sender/Master" und „Empfänger/Slave" nicht. Jedes Board kann sowohl Sender als auch Empfänger sein. Der Übersichtlichkeit halber werden im Folgenden jedoch die Begriffe „Sender" und „Empfänger" mit ihrer üblichen Bedeutung verwendet.

Anhand der übermittelten Daten kann der Empfänger jeden aktiven Sender über seine eindeutige MAC-Adresse identifizieren. Der Umgang mit unterschiedlichen MAC-Adressen auf der Empfängerseite kann jedoch schnell aufwendig werden. Der Einfachheit halber wird daher jedes Board über eine eindeutige Nummer (ID) identifiziert. Wenn drei Sender-Boards vorhanden sind, sind also die ID-Nummern 1 bis 3 im Spiel. Die ID wird zusammen mit den anderen Messdaten an den Empfänger gesendet. Im Beispiel wird eine Strukturevariable definiert, die die Board-ID-Nummer und den aktuellen Messwert enthält.

Zunächst werden die Programme auf die Sender-Controller geladen. Dabei muss unbedingt darauf geachtet werden, dass den verschiedenen Boards unterschiedliche ID-Nummern (myData.id) zugeordnet werden.

Der entsprechende Code für den ersten Sender sieht so aus:

```
// Tx_01.ino
// ESP32 @ IDE 1.8.16

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


uint8_t broadcastAddress[]={0x00,0x00,0x00,0xFF,0xFF,0xFF};// RECEIVER MAC
byte analogInPin=34;

typedef struct struct_message
{   int id; // must be unique for each sender board
    int x; int y;
} 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.print(status==ESP_NOW_SEND_SUCCESS?"Data Sent ":"Send 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);
  peerInfo.channel = 0;   peerInfo.encrypt = false;
  if (esp_now_add_peer(&peerInfo) != ESP_OK){ Serial.println("Failed to add peer"); return; }
}

void loop()
{ myData.id=1; myData.x=analogRead(analogInPin);  myData.y=0;
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData, sizeof(myData));
  Serial.println(result==ESP_OK?"Delivery Success":"Delivery Fail");
  delay(1000);
}
```

Wichtig ist hier auch wieder, die MAC-Adresse des Empfängers einzutragen:

uint8_t broadcastAddress[ ] = {0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF};

Es folgt die „structure" für die Senderdaten. Sie enthält drei Integer-Variablen: die Board-ID, x und y. Falls diese für ein eigenes Projekt angepasst werden, muss diese Veränderung auch auf der Empfängerseite durchgeführt werden.

Im setup() wird der serielle Monitor für Debugging-Zwecke initialisiert. Dann wird der Controller als WLAN-Station aktiviert. Die Callback-Funktion sorgt für das Versenden der Daten.

In der Hauptschleife werden einmal pro Sekunde die gesammelten Daten über ESP-NOW versendet. Dabei wird jeder Variablen ein aktueller Wert zugewiesen:

myData.id=1; myData.x=analogRead(analogInPin); myData.y=0;

Nach der Board-ID wird der aktuelle Analogwert übernommen. Die Variable myData.y steht für weitere Kanäle zur Verfügung. In anderen Anwendungen können hier beispielsweise auch Steuerbefehle oder Digitalwerte etc. verwendet werden. Abschließend wird die Nachricht via ESP-NOW versendet. Details dazu finden sich im zweiten Teil zu dieser Artikelserie.

Sind alle Sender mit ihren Programmen versorgt, kann man daran gehen, den Empfänger zu programmieren. Der Code ist bereits darauf vorbereitet, Daten von drei verschiedenen Boards zu empfangen. Die Anpassung an eine größere Anzahl von Sendern ist problemlos möglich.

```
// Rx_00.ino
// ESP32 @ IDE 1.8.16

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

typedef struct struct_message
{ int id;  int x;  int y;
}struct_message;
struct_message myData;

struct_message board1;
struct_message board2;
struct_message board3;

struct_message boardsStruct[3] = {board1, board2, board3};

void OnDataRecv(const uint8_t * mac_addr, const uint8_t *incomingData, int len)
{ char macStr[18];  Serial.print("Packet received from: ");
  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.println(macStr);
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.printf("Board ID %u: %u bytes\n", myData.id, len);
  // Update the structures with the new incoming data
  boardsStruct[myData.id-1].x = myData.x;
  boardsStruct[myData.id-1].y = myData.y;
  Serial.printf("x value: %d \n", boardsStruct[myData.id-1].x);
  Serial.printf("y value: %d \n", boardsStruct[myData.id-1].y);
  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()
{ // Acess the variables for each board
  int board1X = boardsStruct[0].x;
  int board1Y = boardsStruct[0].y;
  int board2X = boardsStruct[1].x;
  int board2Y = boardsStruct[1].y;
  /*
  int board3X = boardsStruct[2].x;
  int board3Y = boardsStruct[2].y;
  */

  delay(1000);
}
```

Die Daten werden über die übliche „structure“ empfangen. Im Unterschied zu den früheren Projekten muss nun für jedes Sender-Board eine eigene „struct_message“-Variable definiert werden, damit die empfangenen Daten dem entsprechenden Board zugeordnet werden können:

```
struct_message board1;
struct_message board2;
struct_message board3;
...
```

Ein Array enthält alle beteiligten Senderboards:

```
struct_message boardsStruct[3] = {board1, board2, board3};
```

Nach dem Empfang der Daten wird der Inhalt der Datenvariable „incomingData“ in die Variable „myData“ kopiert und steht so zur weiteren Verwendung zur Verfügung.

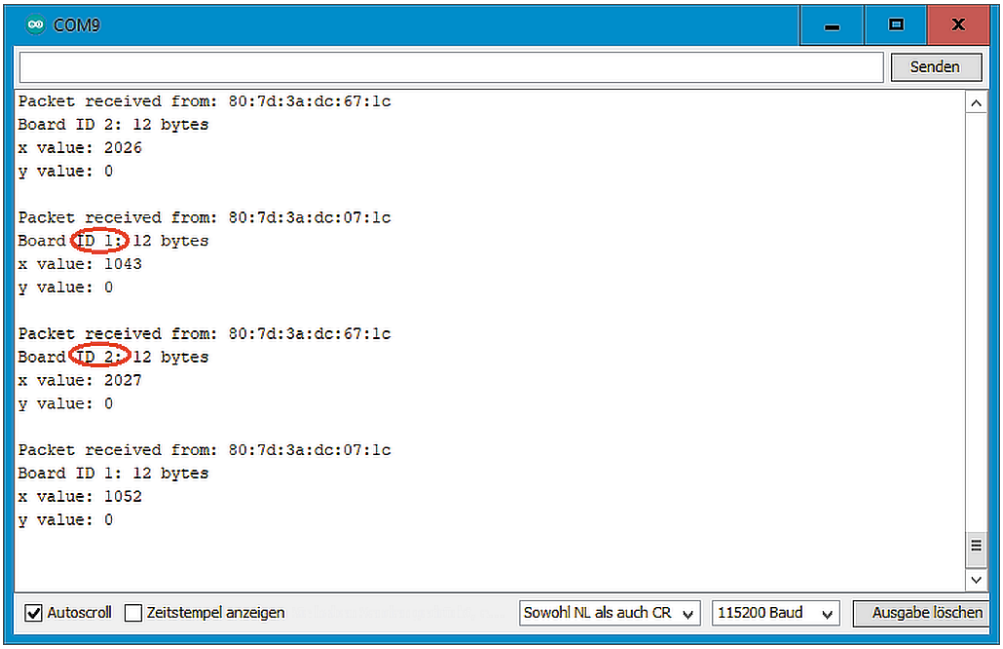


Bild 3: Empfang von Daten mehrerer Sender

Auf diese Weise können die empfangenen Werte den entsprechenden Boards im Array „boardsStruct" zugewiesen werden:

```
boardsStruct[meineDaten.id-1].x=meineDaten.x;
boardsStruct[meineDaten.id-1].y=meineDaten.y;
```

Da Arrays in C 0-indiziert sind, muss von der ID jeweils der Wert 1 abgezogen werden. Der Empfang der Daten kann im seriellen Monitor überprüft werden (Bild 3).

Man erkennt, dass die Board ID (1 oder 2) sowie die Anzahl der übertragenen Bytes (hier 12) korrekt übertragen und angezeigt werden. Es folgt die Ausgabe des Messwerts (0 ... 4095) sowie der Reservevariablen („0").

Die Programme zu diesem Abschnitt finden sich im Ordner „Tx_XX-Rx_00", also „mehrere Sender (XX) – ein Empfänger (00).

Privatrundfunk: Ein Sender – viele Empfänger

Das Einsammeln von Daten verschiedener Boards ist ein wichtiger Anwendungsfall mit vielen praktischen Einsatzmöglichkeiten. Oftmals ist aber auch der umgekehrte Vorgang erwünscht. Bei der Steuerung von Geräten wie Lampen oder Jalousien etc. soll ein einziger (Hand-)Sender viele Empfänger ansprechen. Auch diese Variante kann mit ESP-NOW realisiert werden (Bild 4). Hierbei sind zwei Fälle zu unterscheiden:

- Senden der gleichen Nachricht an alle Boards
- Senden einer jeweils anderen Nachricht an jedes Board

In einem ersten Beispiel sollen identische Daten von einem zentralen Sender an viele Empfänger übertragen werden. Eine Anwendung ist z. B. das zentrale Schalten verschiedener Verbraucher in einem Haushalt über einen einzelnen Handsender. Interessante Anwendungen hierzu wären:

- Das Schalten mehrere Gartenlampen per Knopfdruck auf einem einzelnen Sender. Verschiedene Lampen können so synchron ein- oder ausgeschaltet werden, auch wenn diese weiter (bis zu ca. 100 m) voneinander entfernt stehen.
- Simultane Steuerung mehrerer elektrischer Heizgeräte. Die Geräte können so alle drahtlos auf eine identische Solltemperatur eingestellt werden.

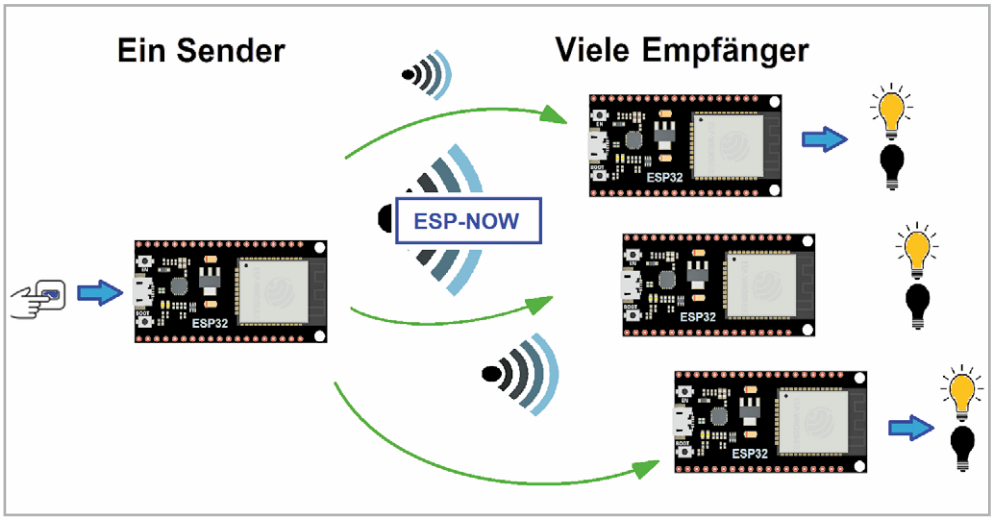


Bild 4: Mehre Empfänger werden von einem Sender gesteuert.

Zunächst soll dazu der Code für den („zentralen") Sender betrachtet werden. Der folgende Code (Tx_00_ident.ino im Ordner „Tx_00-Rx_XX") sendet Daten an mehrere, in diesem Beispiel drei ESP-Boards. Vor dem Hochladen müssen die MAC-Adressen der Empfängerplatinen eingetragen werden. Je nach Anzahl der Empfänger sind Codezeilen hinzuzufügen oder zu löschen.

```
// Tx_00_ident.ino
// ESP32 @ IED 1.8.16
// sender for multiple receivers

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

// REPLACE WITH YOUR ESP RECEIVER'S MAC ADDRESS
uint8_t broadcastAddress1[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x01};
uint8_t broadcastAddress1[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x02};
// uint8_t broadcastAddress3[] = {0xFF, , , , , }; // add more recievers

typedef struct test_struct {int x; int y;} test_struct;

test_struct test; esp_now_peer_info_t peerInfo;

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status)
{ 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:\t");
  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);
  peerInfo.channel = 0;    peerInfo.encrypt = false;
  // register first peer
  memcpy(peerInfo.peer_addr, broadcastAddress1, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){Serial.println("Failed to add peer"); return; }
  // register second peer
  memcpy(peerInfo.peer_addr, broadcastAddress2, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){Serial.println("Failed to add peer"); return; }
  // further peer
  memcpy(peerInfo.peer_addr, broadcastAddress3, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){Serial.println("Failed to add peer"); return; }
}

void loop()
{ test.x=123;  test.y=456;
  esp_err_t result = esp_now_send(0, (uint8_t *) &test, sizeof(test_struct));
  if (result == ESP_OK) Serial.println("Packsge Sent"); else Serial.println("Send Fail");
  delay(2000);
}
```

Die MAC-Adressen der Empfänger werden in den folgenden Zeilen eingetragen:

uint8_t BroadcastAddress1[ ] = {0x00, 0x00, 0x00, 0xff, 0xff, 0xb00};
uint8_t BroadcastAddress2[ ] = {0x00, 0x00, 0x00, 0xff, 0xff, 0xb02};
uint8_t BroadcastAddress3[ ] = {0x00, 0x00, 0x00, 0xff, 0xff, 0xb03};

Es wird wieder eine Structure „test_struct" erstellt, die die zu sendenden Daten enthält. Diese umfasst zwei Integer-Variablen:

typedef struct test_struct { int x; int y;} test_struct;

Die Funktion ontDataSent() ist wieder eine Callback-Funktion, die ausgeführt wird, sobald eine Nachricht gesendet wird. In diesem Fall gibt die Funktion auch die MAC-Adresse des jeweils angesprochenen Empfängers aus. So kann man auf der Senderseite in Erfahrung bringen, welche Empfänger-Boards die Nachricht erhalten haben bzw. welche Empfänger nicht aktiv sind.

Nach dem Registrieren der gültigen Empfänger („peers") können diese direkt adressiert werden. Falls zusätzliche Empfänger erforderlich sind, können diese durch Duplizieren der Zeilen

memcpy(peerInfo.peer_addr, BroadcastAddress, 6);
if(esp_now_add_peer(& peerInfo)! = ESP_OK){ Serial.println („Peer nicht hinzugefügt"); return;}

eingebunden werden.

Die Übertragung der Daten erfolgt in der Hauptschleife in einem Zeitabstand von zwei Sekunden.

Im Beispiel erhalten Variablen numerisch Testwerte:
test.x=123; test.y=456;

In realen Anwendungen können hier Schaltbefehle oder Sensorwerte stehen. Anwendungsbeispiele für das Schalten von Verbrauchern finden sich in den nächsten Abschnitten.

Der passende Empfänger-Code zum zentralen Sender sieht so aus:

```
...

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

//Structure example to receive data
//Must match the sender structure
typedef struct test_struct {  int x;  int y;} test_struct;

//Create a struct_message called myData
test_struct myData;

//callback function that will be executed when data is received
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(„x: ");  Serial.println(myData.x);
  Serial.print("y: ");  Serial.println(myData.y);
  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() {

}
```

Dieser Code kann für alle einzusetzenden Empfänger verwendet werden. Die „structure" zum Empfangen der Daten muss wieder mit der Sender-Structure übereinstimmen:

```
typedef struct test_struct { int x; int y;} test_struct;
```

Die empfangenen Daten sind wieder am seriellen Monitor überprüfbar. Alle Empfänger-Boards zeigen die identischen Zahlen des Senders (Bild 5).

Anstelle der Testdaten kann man nun wieder beliebige Nutzdaten einsetzen. Im einfachsten Fall sind das die binären Daten eines Schalters oder Tasters (0/1 für ein/aus) oder auch quasi-analoge Werte, um beispielsweise eine LED zu dimmen. Die Details dazu wurden bereits im zweiten Artikel der Serie behandelt.

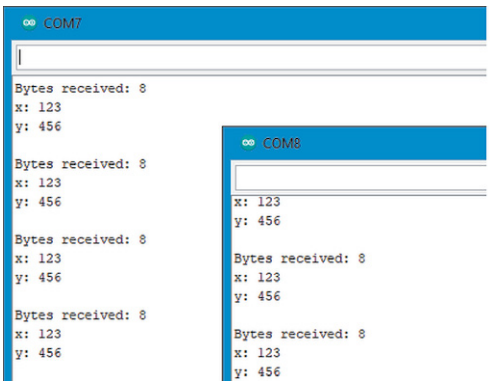


Bild 5: Alle Empfänger erhalten dieselben Daten.

Senden individueller Daten

Der Code zum Senden verschiedener Daten an jedes Empfänger-Board ist leicht aus dem letzten Beispiel ableitbar. Im Folgenden werden daher nur die Unterschiede zum Senden identischer Daten aufgezeigt.

Wenn individuelle Nachrichten an jedes Board gesendet werden sollen, muss eine eigene Datenstruktur für jedes Empfänger-Board erstellt werden:

```
Test_Struct Test1;
test_struct test2;
...
```

In diesem Fall wird derselbe Strukturtyp (test_struct) für alle Empfänger verwendet. Man kann auch unterschiedliche Strukturtypen senden, solange der Empfänger-Code auf die jeweilige Struktur angepasst ist.

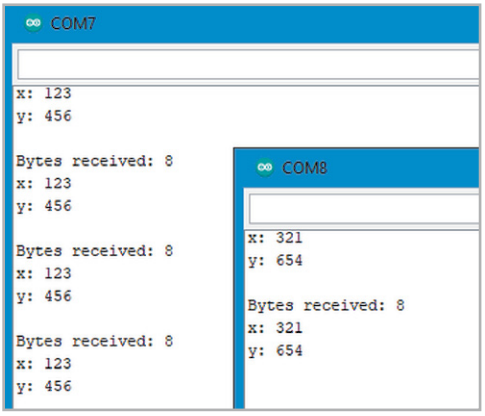


Bild 6: Jeder Empfänger erhält sein eigenes Datenpaket.

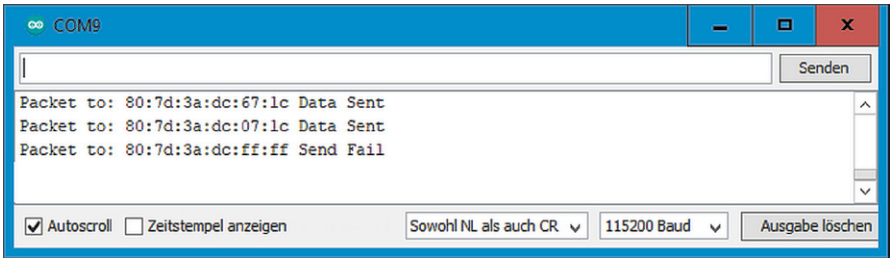


Bild 7: Nur zwei Empfänger sind aktiv!

Zunächst sollen wieder feste, aber jetzt eben unterschiedliche Daten übertragen werden:

```
test1.x=123;        test1.y=456;
test2.x=321;        test2.y=654;
```

Für jeden Empfänger ist die Funktion esp_now_send () einzeln aufzurufen.

Die vollständigen Programme zum Senden individueller Daten finden sich im Download-Paket (Tx_00_diff). Der Empfang – der nun unterschiedlichen Daten - kann wiederum im seriellen Monitor überprüft werden (Bild 6).

Durch die Callback-Funktion steht am Sender die Information über die erfolgreiche Übertragung zur Verfügung. Beim Betrieb von nur zwei Empfängern wird die dritte Informationseinheit als unzustellbar markiert (Bild 7).

Alles aus einer Hand: Viele Geräte mit einem Sender steuern

Eine wichtige Anwendung des Senders mit vielen Empfängern ist die Steuerung von Geräten. Bei entsprechender Programmierung können so etwa

- verschiedene Jalousien
- ein Garagentor
- Innen- und Außenbeleuchtungen

drahtlos geschaltet werden. Wie in den letzten Abschnitten klar wurde, muss dazu nur jeder eingesetzte Empfänger separat beim Sender angemeldet werden. Zum Schalten genügt eine boolesche Variable (Ein = True / Aus = False). Im folgenden Projekt werden zwei LED-Beleuchtungen über einen Sender individuell geschaltet (Bild 7 und Bild 8).

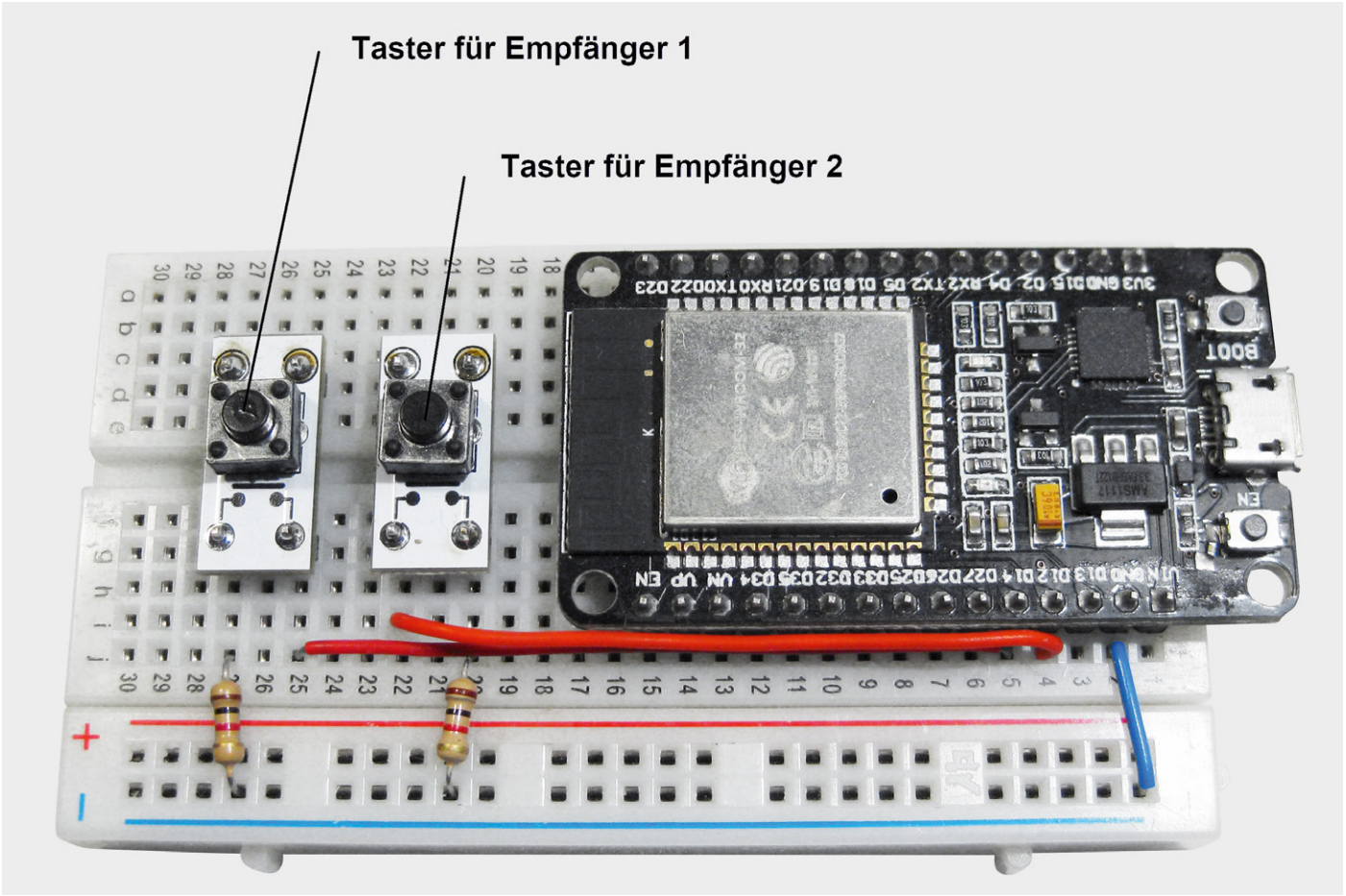


Bild 8: Handsender für das drahtlose Schalten von zwei LED-Beleuchtungen

Die LEDs auf Empfängerseite werden dabei über einen Vorwiderstand jeweils am Port 16 angeschlossen. Die Taster schalten die „low-aktiven" Ports 12 bzw. 14. Der Sketch für den „universellen" Sender sieht so aus (Tx_00_diff_LED_control.ino):

```
// Tx_00_diff_LED_control.ino
// ESP32 @ IED 1.8.16
// sender for multiple receivers - different data for controlling LEDs

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

// REPLACE WITH YOUR ESP RECEIVER'S MAC ADDRESS
uint8_t broadcastAddress1[] = {0x00, 0x00, 0x00, 0xFF, 0xFF, 0x01};
uint8_t broadcastAddress2[] = {0x00, 0x00, 0x00, 0xFF, 0xFF, 0x02};
uint8_t broadcastAddress3[] = {0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF};

byte LED1pin=12, LED2pin=14;

typedef struct test_struct
{ bool x;  bool y;
} test_struct;

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status)
{ 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(": ");
  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);
  esp_now_peer_info_t peerInfo;
  peerInfo.channel = 0;    peerInfo.encrypt = false;

  memcpy(peerInfo.peer_addr, broadcastAddress1, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){ Serial.println("Failed to add peer"); return;}
  memcpy(peerInfo.peer_addr, broadcastAddress2, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){ Serial.println("Failed to add peer"); return;}
  memcpy(peerInfo.peer_addr, broadcastAddress3, 6);
  if (esp_now_add_peer(&peerInfo) != ESP_OK){ Serial.println("Failed to add peer"); return;}

  pinMode(LED1pin,INPUT_PULLUP); pinMode(LED2pin,INPUT_PULLUP);
}

void loop()
{ test_struct LED1;  test_struct LED2;  test_struct LED3;
  if (digitalRead(LED1pin)==false)  LED1.x=true;  else LED1.x=false;
  if (digitalRead(LED2pin)==false)  LED2.x=true;  else LED2.x=false;

  Serial.println(LED1.x); Serial.println(LED2.x);

  esp_err_t result1 = esp_now_send( broadcastAddress1, (uint8_t *) &LED1, sizeof(test_struct));
  if (result1 == ESP_OK) { Serial.print("Sent with success ");} else {Serial.println("Error sending data ");
  }
  delay(100);
  esp_err_t result2 = esp_now_send( broadcastAddress2,  (uint8_t *) &LED2, sizeof(test_struct));
  if (result2 == ESP_OK) {Serial.print(„Sent with success „);}  else {Serial.println(„Error sending the data ");}
  delay(100);
  esp_err_t result3 = esp_now_send(broadcastAddress3, (uint8_t *) &LED3, sizeof(test_struct));

  if (result3 == ESP_OK) { Serial.print("Sent with success "); } else {Serial.println("Error sending the data ");}
  delay(100);
}
```

Wie üblich werden die Empfänger wieder über ihre MAC-Adressen identifiziert. Die zentrale „structure" enthält zwei boolesche Variablen. Diese erhalten ihre Werte über angeschlossene Taster:

```
  if (digitalRead(LED1pin)==false)  LED1.x=true;  else LED1.x=false;
  if (digitalRead(LED2pin)==false)  LED2.x=true;  else LED2.x=false;
```

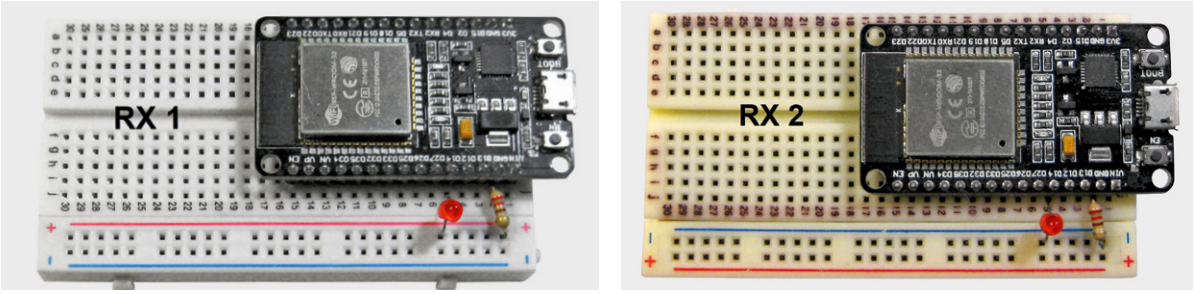


Bild 9: Die beiden identischen Empfänger können individuell geschaltet werden.

Die Taster-Information wird an die einzelnen Empfänger übertragen. Dort wird sie ausgewertet und steuert eine angeschlossene LED. So kann mit jedem Taster auf dem Sender eine einzelne LED individuell geschaltet werden.

Für die Empfänger kann immer der gleiche Sketch (Rx_01_LED_control.ino) verwendet werden, da die Zuordnung der Tasten zum jeweiligen Empfänger im Sender-Sketch erfolgt:

```
// Rx_01_LED_control.ino
// ESP32 @ IED 1.8.16
// receiver for one sender for controlling LEDs

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

byte LEDpin=16;

//Structure to receive data must match sender structure
typedef struct test_struct
{ bool x;  bool y;} test_struct;
test_struct 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("x: ");  Serial.println(myData.x);
  Serial.print("y: ");  Serial.println(myData.y);
  Serial.println();
  Serial.print("Switch State: "); Serial.println(myData.x);
  digitalWrite(LEDpin,myData.x);
}

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

void loop() {}
```

Sobald die Programme auf die Boards geladen sind, kann man nach Belieben „schalten und walten“: Jede Empfänger-LED kann per zugeordnetem Knopfdruck auf dem zentralen Sender ein- und ausgeschaltet werden.

Ausblick

Nachdem in diesem Beitrag die Anwendung des ESP-NOW-Systems als Netzwerkstruktur ausführlich erläutert wurde, soll es im nächsten Artikel um die Einbindung der ESP32-Boards ins Internet gehen.

Mit den entsprechenden Programmen ist es möglich, auf einem einzelnen ESP32 nicht nur das ESP-NOW-Protokoll laufen zu lassen, sondern auch einen kompletten Netzwerk-Server zu installieren. Damit ist man in der Lage, die von einzelnen Boards eingesammelten Daten über ein zentrales ESP32-Board ins Internet zu übertragen.

So können auch umfangreichere Datensätze anschaulich dargestellt werden. Die Web-Applikation „ThingSpeak“ erlaubt es, sowohl Einzelwerte als auch grafische Darstellungen in verschiedenen Varianten anzuzeigen. Ein besonderer Vorteil dieses Vorgehens ist, dass die Daten nun auch außerhalb des eigenen WLANs zur Verfügung gestellt werden können. ELV

Material	Artikel-Nr.
3x ESP32-Board	145164
LEDs mit Vorwiderständen, Taster und Potenziometer sind z. B. im Prototypenadapterset PAD-PRO-EXSB enthalten	158980

Weitere Infos

Download-Paket: Artikel-Nr. 253714