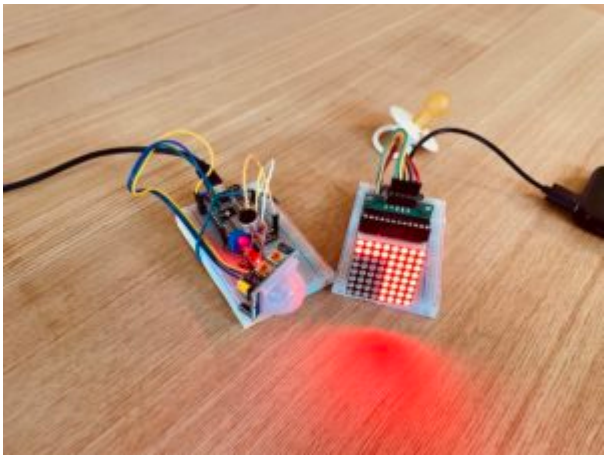
Babyphone mit Geräusch- und Bewegungsmelder



Dieses Projekt dreht sich rund um den Schlaf von Babys und kleinen Kindern: Du baust ein Babyphone, das aus zwei ESP8266 besteht, die mit **ESP-NOW** miteinander verbunden sind. Am ersten Microcontroller schließt du einen PIR-Sensor, um Bewegungen zu erkennen sowie einen Geräuschsensor an, um die Lautstärke zu ermitteln.

Gesendet werden die Daten an einen zweiten ESP8266, an dem wiederum ein LED-Matrix-Display angeschlossen ist. Hierauf kannst du die aktuelle Lautstärke im Schlafzimmer ermitteln. Außerdem leuchtet das Display auf, wenn eine Bewegung erkannt wurde.

Den ESP8266 mit den Sensoren stellst du neben das Bett der oder des Kleinen und jenen mit dem Display dorthin, wo du dich selbst aufhältst. So hast du das Geschehen im Schlafzimmer immer im Blick.

Du lernst mit diesem Projekt, wie du

- einen PIR- und einen Geräuschsensor am ESP8266 anschließt,
- ein LED-Matrix-Display verwendest und
- zwei ESP8266 per ESP-NOW miteinander verbindest.

Diese Bauteile benötigst du

- 2x NodeMCU ESP8266
- 1x PIR-Bewegungssensor
- 1x KY-037 oder KY-038 Geräuschsensor
- 1x 8×8 LED-Matrix mit MAX7219-Treiber
- Kabel und 2 Breadboards
- Optional: 1 oder 2 Powerbanks für den Betrieb

Der Empfänger

Los geht es mit dem Teil des Babyphones, auf dessen Display du ablesen kannst, was sich im Schlaf- oder Kinderzimmer so tut. Hierfür verwendest du ein 8×8 LED-Matrix-Display. Auf der einen Seite des Displays erstellst du eine Art Equalizer, der dir die gemessene Lautstärke anzeigt. Die andere Seite leuchtet rot auf, sobald eine Bewegung erkannt und vom Sender weitergegeben wurde.

Verbinde das Matrix-Display wie folgt mit dem ESP8266:

Display	ESP8266
GND	GND
VCC	3,3V
DIN	D5
CLD	D6
CS	D7

Als nächstes musst du die MAC-Adresse des ESP8266 herausfinden, um ESP-NOW für die Kommunikation zwischen Sender und Empfänger nutzen zu können. MAC steht für Media Access

Control und ist eine eindeutige Adresse zur Identifikation eines Geräts in einem Netzwerk.

Um diese Adresse herauszufinden, kannst du folgenden kleinen Sketch auf den Empfänger hochladen und die MAC-Adresse im Seriellen Monitor ablesen:

```
#include <ESP8266WiFi.h>

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  Serial.println(WiFi.macAddress());
}

void loop() {}
```

Deine Adresse könnte zum Beispiel wie folgt aussehen: **EC:FA:BC:6E:EE:F4**

Diese Adresse wirst du gleich im Sketch des Senders benötigen, um die Messdaten auf dem Matrix-Display anzuzeigen. Zunächst jedoch der eigentliche Sketch, der die Lautstärke und Bewegungen entgegennimmt und visualisiert:

```
#include <ESP8266WiFi.h>
#include <espnow.h>
#include <LedControl.h>

// Pins des Displays
const int DIN_PIN = D5;
const int CLK_PIN = D6;
const int CS_PIN = D7;

// LED Matrix Setup
LedControl lc = LedControl(DIN_PIN, CLK_PIN, CS_PIN, 1);

// Struktur der zu empfangenden Daten
struct SensorData {
```

```
  bool motionDetected;
  int noiseLevel;
};

void setup() {
  Serial.begin(115200);

  // LED matrix initialisieren
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);  // Helligkeit (0-15)
  lc.clearDisplay(0);

  // ESP-NOW initialisieren
  WiFi.mode(WIFI_STA);
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Funktion für die empfangenen Daten
  esp_now_register_recv_cb(onDataReceived);
}

void loop() {
  // Leer, da du die Funktion onDataReceived verwendest
}

void onDataReceived(uint8_t *mac, uint8_t *incomingData,
uint8_t len) {
  SensorData data;
  memcpy(&data, incomingData, sizeof(data));

  // Update des Displays
  updateLEDMatrix(data.motionDetected, data.noiseLevel);

  // Empfangene Daten im Seriellen Monitor ausgeben
  Serial.print("Motion: ");
  Serial.print(data.motionDetected);
  Serial.print(" | Noise: ");
  Serial.println(data.noiseLevel);
}
```

```
void updateLEDMatrix(bool motionDetected, int noiseLevel) {
  lc.clearDisplay(0);

  // Bewegung anzeigen
  if (motionDetected) {
    for (int row = 0; row < 4; row++) {
      for (int col = 0; col < 8; col++) {
        lc.setLed(0, row, col, true);
      }
    }
  }

  // Lautstärke anzeigen
  int noiseLEDs = map(noiseLevel, 150, 300, 0, 8);
  for (int row = 4; row < 8; row++) {
    for (int col = 0; col < noiseLEDs; col++) {
      lc.setLed(0, row, col, true);
    }
  }
}
```

Erklärung des Sketchs

Lass uns einen genaueren Blick auf den Code in diesem Sketch werfen.

Bibliotheken und Definitionen

```
#include <ESP8266WiFi.h>
#include <espnow.h>
#include <LedControl.h>

const int DIN_PIN = D5;
const int CLK_PIN = D6;
const int CS_PIN = D7;

LedControl lc = LedControl(DIN_PIN, CLK_PIN, CS_PIN, 1);

struct SensorData {
```

```
  bool motionDetected;
  int noiseLevel;
};
```

Der Code verwendet die ESP8266WiFi- und espnow-Bibliotheken für die drahtlose Kommunikation und die LedControl-Bibliothek für die Steuerung der LED-Matrix. **Letztere musst du wahrscheinlich noch über den Bibliotheksmanager installieren.**

Die Pins für die LED-Matrix werden definiert (DIN, CLK, CS) und ein LedControl-Objekt erstellt, um die LED-Matrix zu steuern. Zuletzt erzeugst du eine Struktur, um die empfangenen Daten in Variablen zu speichern.

Setup-Funktion

```
void setup() {
  Serial.begin(115200);

  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);

  WiFi.mode(WIFI_STA);
  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  esp_now_register_recv_cb(onDataReceived);
}
```

- Die serielle Kommunikation wird initialisiert.
- Die LED-Matrix wird eingeschaltet, die Helligkeit auf 8 (von 15) gesetzt und das Display gelöscht.
- Der ESP8266 wird in den Station-Modus versetzt.

- ESP-NOW wird initialisiert und ein Fehler wird gemeldet, falls die Initialisierung fehlschlägt.
- Die Funktion onDataReceived wird als Callback für empfangene Daten registriert.

Hauptschleife

```
void loop() {
}
```

Der Loop des Sketchs belibt leer, da die Datenverarbeitung in der folgenden Funktion erfolgt.

Datenempfangs-Callback

```
void onDataReceived(uint8_t *mac, uint8_t *incomingData, uint8_t len) {
  SensorData data;
  memcpy(&data, incomingData, sizeof(data));

  updateLEDMatrix(data.motionDetected, data.noiseLevel);

  Serial.print("Motion: ");
  Serial.print(data.motionDetected);
  Serial.print(" | Noise: ");
  Serial.println(data.noiseLevel);
}
```

- Diese Funktion wird aufgerufen, wenn Daten empfangen werden.
- Die empfangenen Daten werden in die SensorData-Struktur kopiert.
- Die LED-Matrix wird mit den empfangenen Daten

aktualisiert.
- Die empfangenen Daten werden im Seriellen Monitor ausgegeben.

LED-Matrix-Aktualisierung

```
void updateLEDMatrix(bool motionDetected, int noiseLevel) {
  lc.clearDisplay(0);

  if (motionDetected) {
    for (int row = 0; row < 4; row++) {
      for (int col = 0; col < 8; col++) {
        lc.setLed(0, row, col, true);
      }
    }
  }

  int noiseLEDs = map(noiseLevel, 150, 300, 0, 8);
  for (int row = 4; row < 8; row++) {
    for (int col = 0; col < noiseLEDs; col++) {
      lc.setLed(0, row, col, true);
    }
  }
}
```

- Diese Funktion aktualisiert die LED-Matrix basierend auf den empfangenen Sensordaten.
- Zuerst wird das Display gelöscht.
- Wenn Bewegung erkannt wurde, werden die oberen 4 Reihen der Matrix vollständig beleuchtet.
- Der Geräuschpegel wird auf die unteren 4 Reihen abgebildet. Die Anzahl der beleuchteten LEDs hängt vom Geräuschpegel ab, wobei Werte zwischen 150 und 300 auf 0 bis 8 LEDs abgebildet werden.

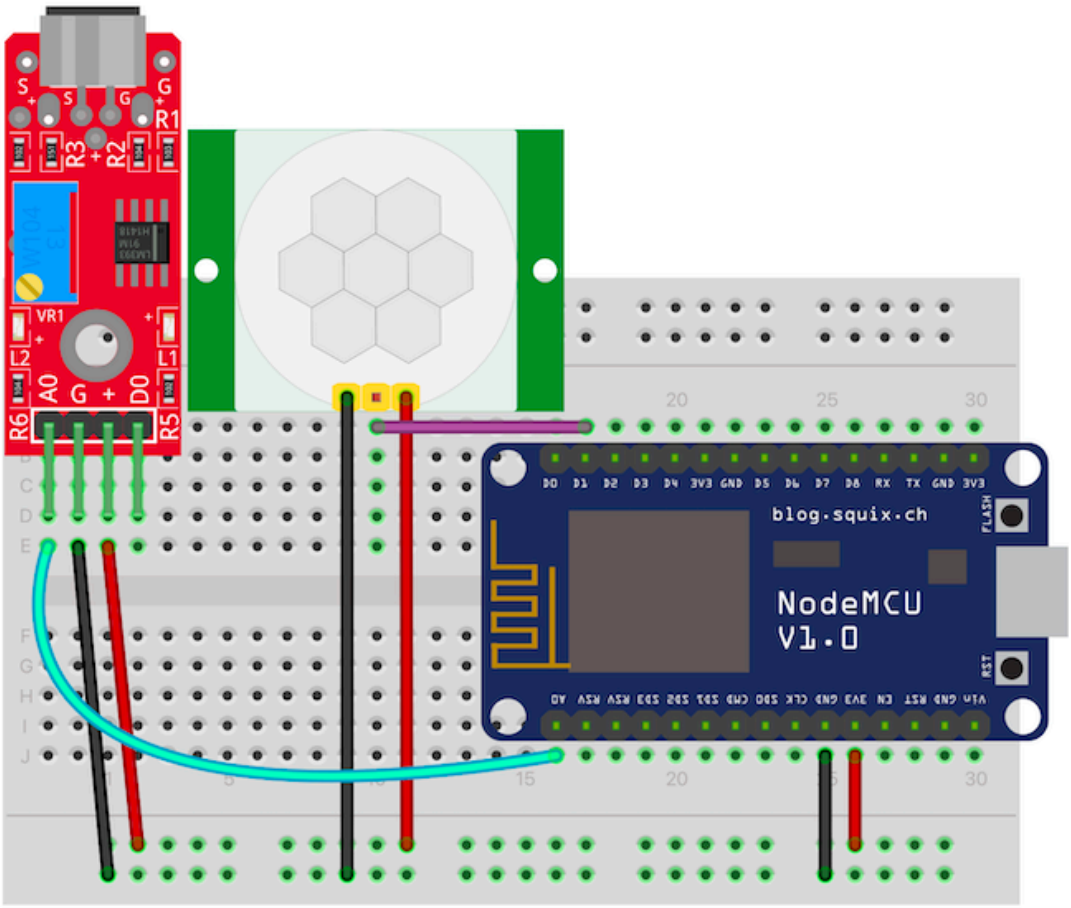
Dieser Code empfängt also kontinuierlich Daten von einem Sender, verarbeitet diese und stellt sie visuell auf einer LED-Matrix dar.

Wichtig ist hier noch die Map-Funktion, die im Code oben Werte zwischen 150 und 300 auf die acht Zeilen des Displays „mappt“. Das sind Werte, die du einerseits direkt am Geräuschsensor einstellst und andererseite experimentell ermitteln musst. Also wie weit nach oben geht der Wert des Sensors bei einer bestimmten Geräuschkulisse? Idealerweise werden alle 8 Zeilen erleuchtet, wenn eine maximal „übliche“ Lautstärke zum Beispiel durch Schreien erreicht wird.

Der Sender des Babyphones

Kommen wir zum zweiten Teil des Babyphones – dem Sender samt des Geräusch- und Bewegungsmelders. **Dieser ESP8266 ermittelt, ob sich das Kind bewegt oder gar weint oder schreit.** Hierfür eignet sich ein PIR-Sensor, der Bewegungen im Dunkeln erkennt und dessen Empfindlichkeit du so einstellen kannst, dass nicht jedes Umdrehen im Bett gleich eine Meldung auslöst. Dazu kommt noch ein einfacher Geräuschsensor zum Einsatz, der die Lautstärke im Raum in eine Zahl zwischen 0 und 1023 darstellt.

Schließe zunächst beide Sensoren wie folgt am ESP8266 an:



Vom Geräuschsensor liest du den analogen Ausgang aus, was bedeutet, dass du den analogen Eingang A0 des ESP8266 verwenden musst. Bei der Wahl des Digitalpins hast du freie Wahl – im Folgenden ist das der Pin D1.

Kommen wir gleich zum Sketch:

```
#include <ESP8266WiFi.h>
#include <espnow.h>

const int PIR_PIN = D1;
const int NOISE_PIN = A0;

//Die MAC-Adresse des Empfängers
uint8_t receiverAddress[] = {0xEC, 0xFA, 0xBC, 0x6E, 0xEE, 0xF4};

struct SensorData {
  bool motionDetected;
```

```
  int noiseLevel;
};

void OnDataSent(uint8_t *mac_addr, uint8_t sendStatus) {
  Serial.print("Last Packet Send Status: ");
  if (sendStatus == 0){
    Serial.println("Delivery success");
  }
  else{
    Serial.println("Delivery fail");
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(PIR_PIN, INPUT);
  pinMode(NOISE_PIN, INPUT);

  WiFi.mode(WIFI_STA);
  WiFi.disconnect();

  if (esp_now_init() != 0) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  esp_now_set_self_role(ESP_NOW_ROLE_CONTROLLER);
  esp_now_register_send_cb(OnDataSent);

   esp_now_add_peer(receiverAddress, ESP_NOW_ROLE_SLAVE, 1,
NULL, 0);

  Serial.println("Sender initialized");
}

void loop() {
  SensorData data;
  data.motionDetected = digitalRead(PIR_PIN);
  data.noiseLevel = analogRead(NOISE_PIN);

    esp_now_send(receiverAddress, (uint8_t *) &data,
```

```
sizeof(data));

  Serial.print("Sending - Motion: ");
  Serial.print(data.motionDetected);
  Serial.print(" | Noise: ");
  Serial.println(data.noiseLevel);

  delay(200);
}
```

Hast du noch die MAC-Adresse des Empfängers zu Hand? Diese trägst du im oberen Bereich des Sketchs ein – ersetze hierfür die obige Adresse durch deine eigene.

Vieles dürfte dir in diesem Sketch schon aus jenem des Empfängers bekannt sein. Im Loop erfasst du die Daten des Sensors aus und sendest sie per ESP-NOW an den ESP8266 des Empfängers. Ganz zum Schluss stellst du mit einem Delay ein, wie häufig die Daten gemessen werden – hier ist ein Wert von 200 Millisekunden praktikabel. Du kannst aber natürlich damit experimentieren.

Den Geräuschsensor einstellen

Eine Sache musst du jedoch an der Hardware des Babyphone noch erledigen und den Sketch des Empfängers entsprechend aktualisieren. Am Geräuschsensor findest du eine kleine Schraube, mit der einstellen kannst, welcher Wert Stille bedeuten soll. Lade den obigen Sketch auf deinen ESP8266 und beobachte im Seriellen Monitor die gemessenen Werte. Drehe nun an der Schraub, bis die Wert sich etwas unter 200 einpendeln.

Mach nun in etwas Entfernung ein lauteres Geräusch (zum Beispiel in der Lautstärke eines weinenden Kindes) und beobachte, wie sich die Messwerte verhalten. Dieser obige Wert wird der Wert, bei dem alle 8 Zeilen des Matrix-Displays leuchten sollen. **Passe nun die Zahl 300 im Empfänger-Sketch**

entsprechend an:

```
int noiseLEDs = map(noiseLevel, 150, 300, 0, 8);
```

Und das war es schon. Versorge beide ESP8266 mit Strom, zum Beispiel mit Hilfe von Powerbanks. Anschließend sollten sie eine Verbindung miteinander aufbauen und das Display aufleuchten.

Falls nicht, überprüfe noch einmal die MAC-Adresse, die du eingetragen hast. Stimmt sie mit jener des Empfängers überein?

Weitere Ideen für dein Babyphone

Neben dem Display könntest du natürlich auch noch andere Methoden nutzen, um dich informieren zu lassen. Das könnte ein einfacher Piezo-Summer sein, aber auch eine WhatsApp-Nachricht auf dein Smartphone.