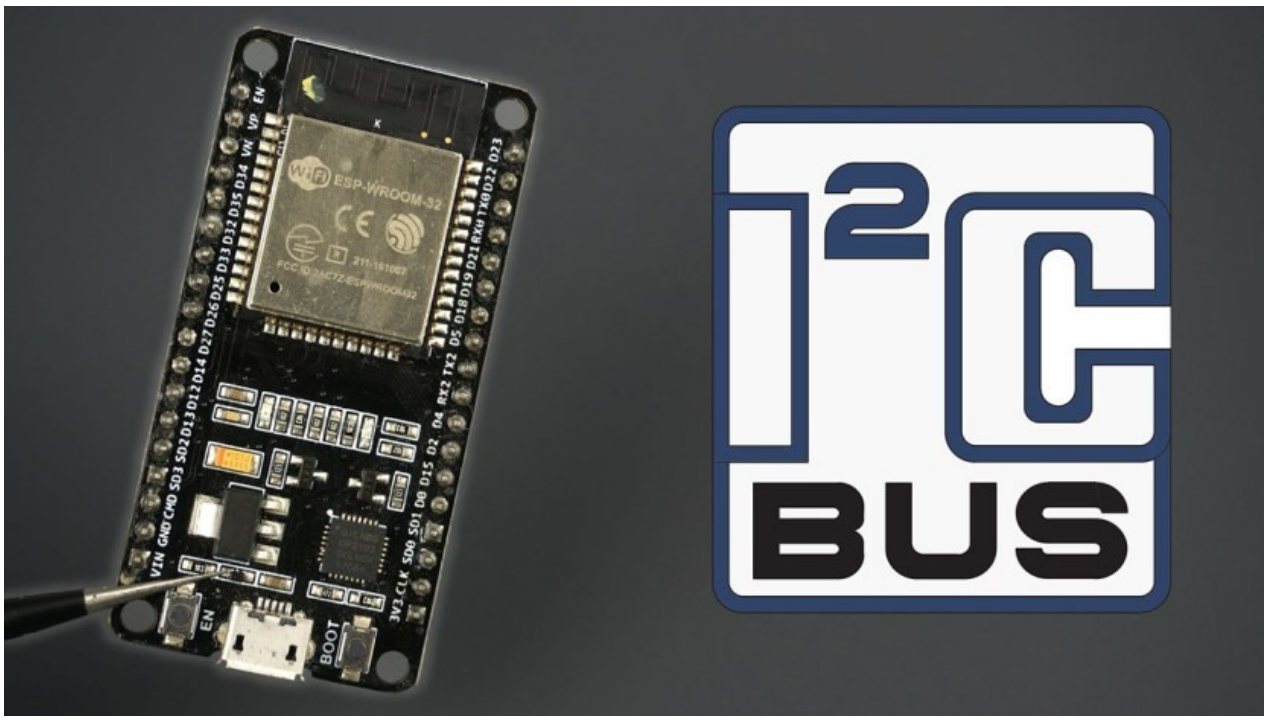
ESP32 I2C Kommunikation- Set Pins, mehrere Busschnittstellen und Peripheriegeräte (Arduino IDE)

Das ESP32 verfügt über zwei I2C-Busschnittstellen, die als I2C-Master oder Slave dienen können. In diesem Tutorial werfen wir einen Blick auf das I2C-Kommunikationsprotokoll mit dem ESP32 mit Arduino IDE: Wie wählen Sie I2C-Pins, verbinden mehrere I2C-Geräte mit demselben Bus und wie Sie die beiden I2C-Busschnittstellen verwenden.



In diesem Tutorial behandeln wir folgende Konzepte:

- I2C Geräte mit ESP32 verbinden
- Scan I2C Adresse mit ESP32
- Verwenden Sie verschiedene I2C-Pins mit ESP32 (Standard-I2C-Pins ändern)
- ESP32 mit mehreren I2C Geräten
    - gleicher Bus, verschiedene Adressen
    - dieselbe Adresse
- ESP32 mit zwei I2C Busschnittstellen
- ESP32 I2C Master und Slave (I2C Kommunikation zwischen zwei ESP32)

Wir programmieren das ESP32 mit Arduino IDE, also sollten Sie vor diesem Tutorial das ESP32 Add-on in Ihrer Arduino IDE installieren lassen. Folgen Sie dem nächsten Tutorial, um das ESP32 auf der Arduino IDE zu installieren, wenn Sie es noch nicht getan haben.

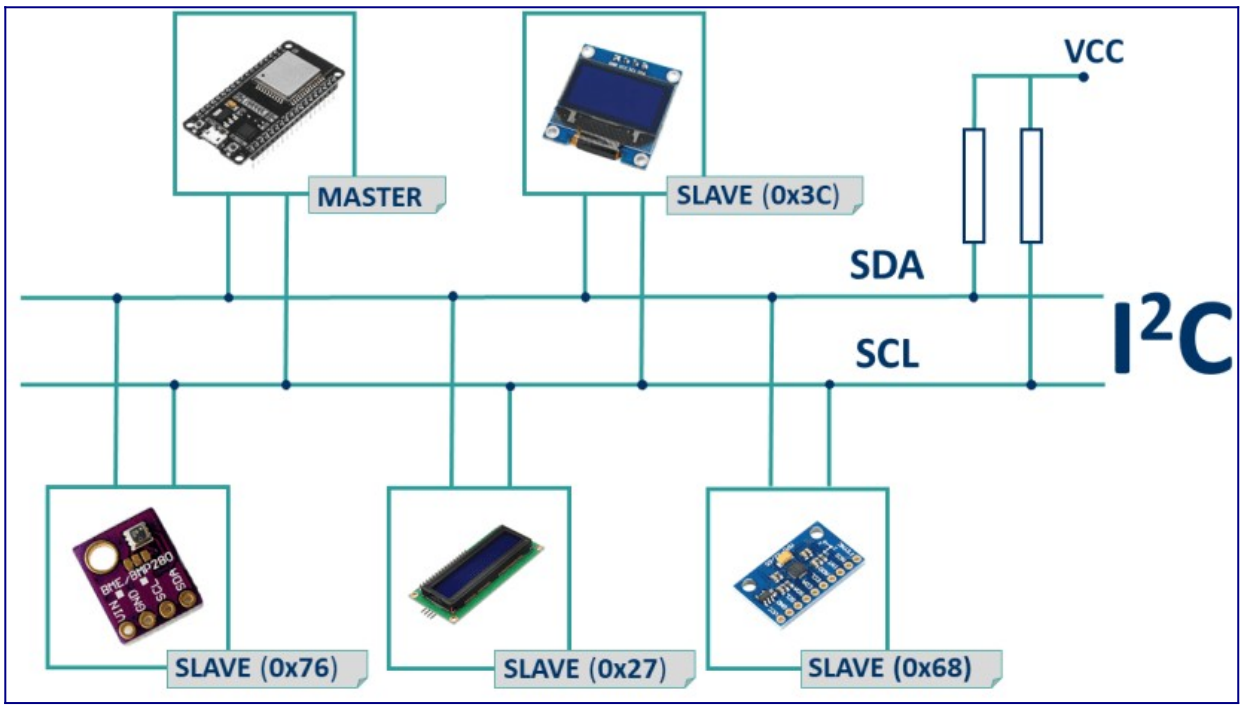
- Installieren des ESP32 Boards in Arduino IDE (Windows, Mac OS X und Linux)

ESP32 I2C Kommunikationsprotokoll

I2C bedeutet, **dass ich Iden** C-Irst (es ist ausgeprägter I-Quadrat-C) ntegred, und es ist ein synchrones, multimasterfarbenes, multisklaviges Kommunikationsprotokoll. Sie können verbinden :

- **Mehrere Slaves zu einem Master:** So liest Ihr ESP32 aus einem BME280-Sensor mit I2C und schreibt die Sensorwerte in ein I2C OLED-Display.
- **Mehrere Meister steuern denselben Slave:** zum Beispiel zwei ESP32-Boards, die Daten auf dasselbe I2C OLED-Display schreiben.

Wir verwenden dieses Protokoll oft mit dem ESP32, um mit externen Geräten wie Sensoren und Displays zu kommunizieren. In diesen Fällen ist das ESP32 der Master-Chip und die externen Geräte sind die Slaves.



Wir haben mehrere Tutorials mit dem ESP32-Zug zu I2C-Geräten:

- 0,96 Zoll I2C OLED Display mit ESP32
- ESP32 Eingebautes OLED-Board
- I2C LCD Display mit ESP32
- BMP180 mit ESP32
- BME280 mit ESP32

ESP32 I2C Busschnittstellen

Das ESP32 unterstützt die I2C-Kommunikation über seine zwei I2C-Busschnittstellen, die je nach Benutzerkonfiguration als I2C-Master oder Slave dienen können. Entsprechend dem ESP32-Datenblatt unterstützt die I2C-Schnittstellen des ESP32:

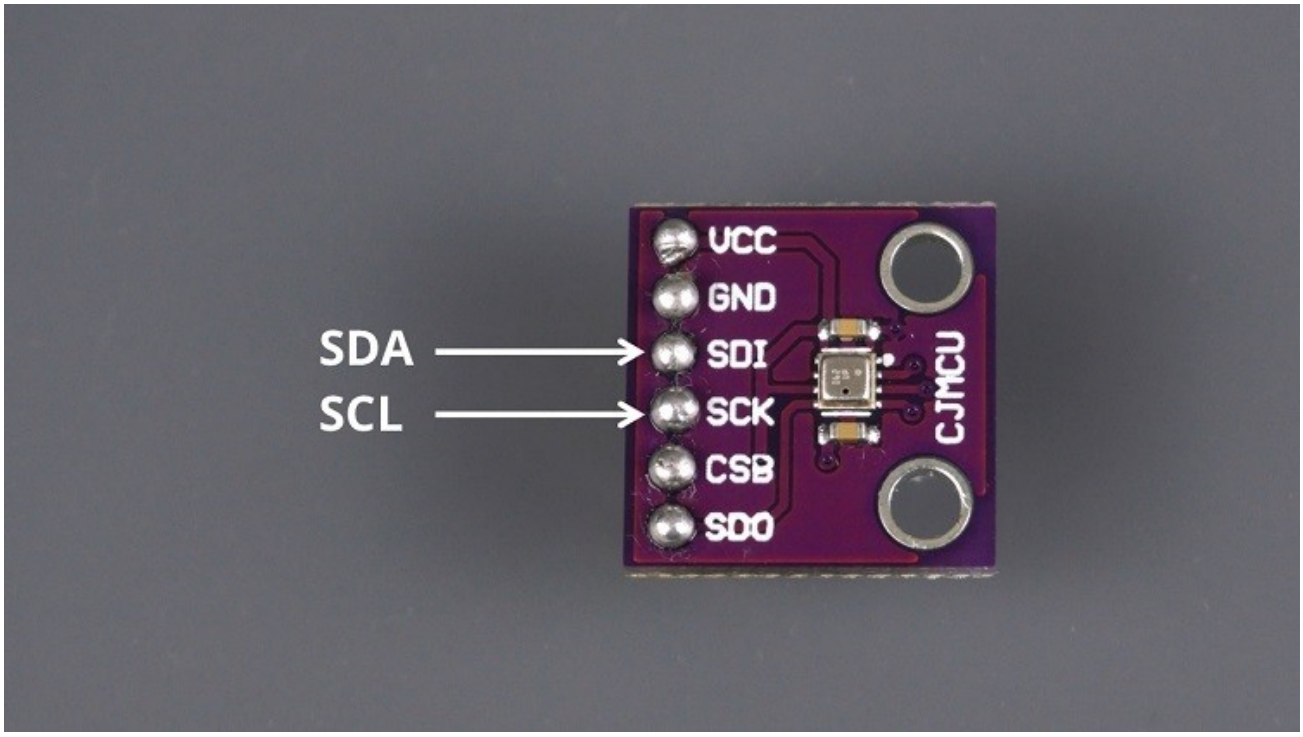
- Standardmodus (100 Kbit/s)
- Schneller Modus (400 Kbit/s)
- Bis zu 5 MHz, aber durch SDA-Pull-up-Stärke eingeschränkt
- 7-Bit/10-Betonungsmodus

- Dual-Adressing-Modus. Benutzer können Befehlsregister programmieren, um I2C-Schnittstellen zu steuern, damit sie mehr Flexibilität haben

I2C Geräte mit ESP32 verbinden

Das Kommunikationsprotokoll I2C verwendet zwei Leitungen, um Informationen zu teilen. Eines wird für das Taktsignal (**SCL**) verwendet und das andere zum Senden und Empfangen von Daten (**SDA**).

Hinweis: In vielen Breakout-Boards kann die SDA-Linie auch als SDI und die SCL-Linie als SCK bezeichnet werden.



Die Linien SDA und SCL sind schwach, so dass sie mit Widerständen hochgezogen werden sollten. Typische Werte sind 4,7k Ohm für 5V-Geräte und 2,4k Ohm für 3.3V Geräte.

Die meisten Sensoren, die wir in unseren Projekten verwenden, sind Breakout-Boards, die bereits die Widerstände eingebaut haben. Also, normalerweise, wenn Sie es mit dieser Art von Elektronikkomponenten zu tun haben, müssen Sie sich darüber keine Sorgen machen.

Die Verbindung eines I2C-Geräts an ein ESP32 ist normalerweise so einfach wie die Verbindung von GND mit GND, SDA mit SDA, SCL und einer positiven Stromversorgung zu einem peripheren, normalerweise 3.3V (aber es hängt von dem Modul ab, das Sie verwenden).

I2C Gerät	**ESP32**
SDA	SDA (Standard ist GPIO 21)
SCL	SCL (Standard ist GPIO 22)
GND	GND
VCC	in der Regel 3.3V oder 5V

Bei Verwendung des ESP32 mit Arduino IDE sind die Standard-I2C-Pins GPIO 22 (SCL) und GPIO 21 (SDA), aber Sie können Ihren Code für alle anderen Pins konfigurieren.

Leseempfehlung: ESP32 GPIO Referenzhandbuch

Scan I2C Adresse mit ESP32

Mit der I2C-Kommunikation hat jeder Sklave im Bus seine eigene Adresse, eine Hexadezimalzahl, die es dem ESP32 ermöglicht, mit jedem Gerät zu kommunizieren.

Die I2C-Adresse ist in der Regel auf dem Datenblatt des Bauteils zu finden. Wenn es jedoch schwierig ist, es herauszufinden, müssen Sie möglicherweise eine I2C-Scannerskizzen erstellen, um die I2C-Adresse herauszufinden.

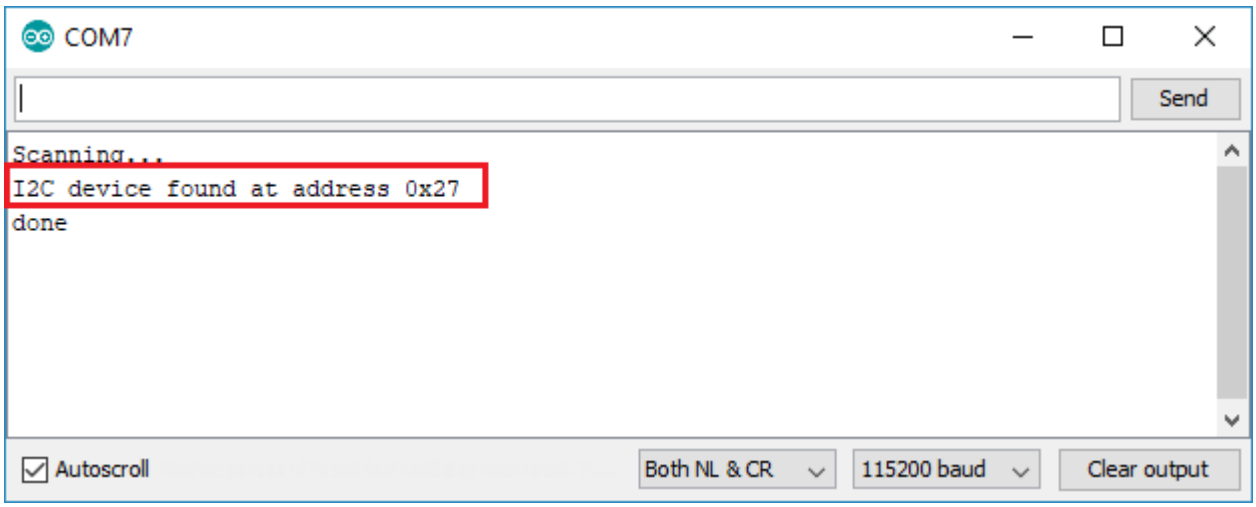
Mit der folgenden Skizze finden Sie die I2C-Adresse Ihrer Geräte.

```
/*********

  Rui Santos

  Complete project details at https://randomnerdtutorials.com

*********/


#include <Wire.h>


void setup() {

  Wire.begin();

  Serial.begin(115200);

  Serial.println("\nI2C Scanner");

}


void loop() {

  byte error, address;

  int nDevices;

  Serial.println("Scanning...");

  nDevices = 0;

  for(address = 1; address < 127; address++ ) {

    Wire.beginTransmission(address);

    error = Wire.endTransmission();
```

```
    if (error == 0) {
      Serial.print("I2C device found at address 0x");
      if (address<16) {
        Serial.print("0");
      }
      Serial.println(address,HEX);
      nDevices++;
    }
    else if (error==4) {
      Serial.print("Unknow error at address 0x");
      if (address<16) {
        Serial.print("0");
      }
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0) {
    Serial.println("No I2C devices found\n");
  }
  else {
    Serial.println("done\n");
  }
  delay(5000);
}
```

View raw code

Sie erhalten etwas Ähnliches in Ihrem Serial Monitor. Dieses konkrete Beispiel ist für ein I2C LCD Display.



Verwenden Sie verschiedene I2C-Pins mit ESP32 (Standard-I2C-Pins ändern)

Mit dem ESP32 können Sie fast jeden Pin auf I2C-Funktionen setzen, Sie müssen das nur in Ihrem Code festlegen.

Verwenden Sie das ESP32 mit der Arduino IDE Wire.h Bibliothek zur Kommunikation mit Geräten mit I2C. Mit dieser Bibliothek initialisiert man das I2C wie folgt:

```
Wire.begin(I2C_SDA, I2C_SCL);
```

Also müssen Sie nur Ihre gewünschten SDA- und SCL-GPIOs auf die I2C-SDA und I2C-SCL Variablen.

Wenn Sie jedoch Bibliotheken verwenden, um mit diesen Sensoren zu kommunizieren, funktioniert dies möglicherweise nicht und es könnte ein bisschen schwierig sein, andere Pins auszuwählen. Das passiert, weil diese Bibliotheken Ihre Pins überschreiben könnten, wenn Sie nicht Ihre eigenen weitergeben Draht Beispiel bei der Initialisierung der Bibliothek.

In diesen Fällen müssen Sie die *.cpp* Bibliotheksdateien und sehen, wie Sie Ihre eigenen passieren Zwei Wire Parameter.

Zum Beispiel, wenn Sie einen genaueren Blick auf die Adafruit BME280 Bibliothek, du wirst herausfinden, dass du deinen eigenen passieren kannst Zwei Wire an die start() Methode.

```
/*!
 *   @brief  Initialise sensor with given parameters / settings
 *   @param addr the I2C address the device can be found on
 *   @param theWire the I2C object to use
 *   @returns true on success, false otherwise
 */
bool Adafruit_BME280::begin(uint8_t addr, TwoWire *theWire) {
  _i2caddr = addr;
  _wire = theWire;
  return init();
}
```

Also die Beispielskizze aus dem BME280 mit anderen Pins zu lesen, zum Beispiel GPIO 33 als SDA und GPIO 32 wie SCL wie folgt.

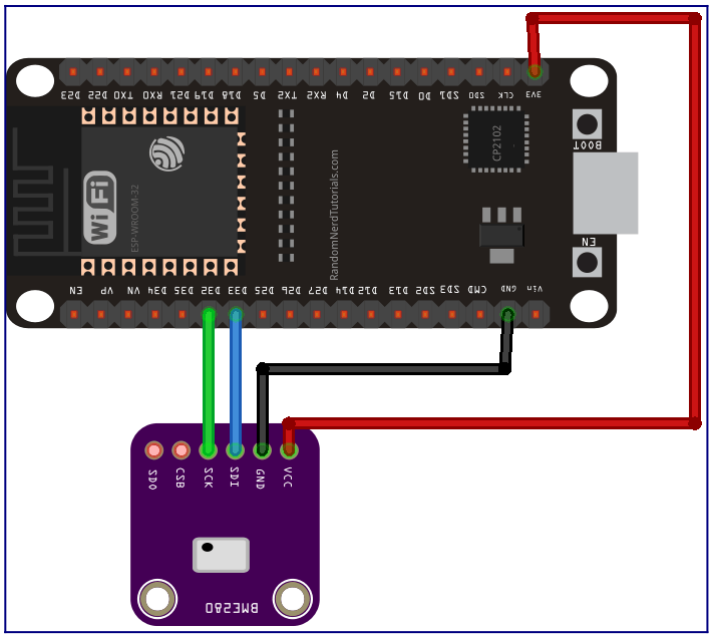
```
/*

  Rui Santos

  Complete project details at https://RandomNerdTutorials.com/esp32-i2c-
communication-arduino-ide/



  Permission is hereby granted, free of charge, to any person obtaining a copy

  of this software and associated documentation files.



  The above copyright notice and this permission notice shall be included in all

  copies or substantial portions of the Software.

*/



#include <Wire.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_BME280.h>



#define I2C_SDA 33

#define I2C_SCL 32
```

```
#define SEALEVELPRESSURE_HPA (1013.25)

TwoWire I2CBME = TwoWire(0);
Adafruit_BME280 bme;

unsigned long delayTime;

void setup() {
  Serial.begin(115200);
  Serial.println(F("BME280 test"));
  I2CBME.begin(I2C_SDA, I2C_SCL, 100000);

  bool status;

  // default settings
  // (you can also pass in a Wire library object like &Wire2)
  status = bme.begin(0x76, &I2CBME);
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }

  Serial.println("-- Default Test --");
  delayTime = 1000;

  Serial.println();
}
```

```
void loop() {
  printValues();
  delay(delayTime);
}

void printValues() {
  Serial.print("Temperature = ");
  Serial.print(bme.readTemperature());
  Serial.println(" *C");

  // Convert temperature to Fahrenheit
  /*Serial.print("Temperature = ");
  Serial.print(1.8 * bme.readTemperature() + 32);
  Serial.println(" *F");*/

  Serial.print("Pressure = ");
  Serial.print(bme.readPressure() / 100.0F);
  Serial.println(" hPa");

  Serial.print("Approx. Altitude = ");
  Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  Serial.println(" m");

  Serial.print("Humidity = ");
  Serial.print(bme.readHumidity());
  Serial.println(" %");
```

```
  Serial.println();

}
```

View raw code



Werfen wir einen Blick auf die relevanten Teile, um andere I2C-Pins zu verwenden.

Definieren Sie zuerst Ihre neuen I2C-Pins auf der I2C-SDA und I2C-SCL Variablen. In diesem Fall verwenden wir GPIO 33 und GPIO 32.

```
#define I2C_SDA 33
#define I2C_SCL 32
```

Erstellen Sie ein neues Zwei Wire z.B. In diesem Fall heißt es I2CBME. Das schafft einfach einen I2C-Bus.

```
TwoWire I2CBME = TwoWire(0);
```

Im setup(), initialisieren Sie die I2C-Kommunikation mit den Pins, die Sie zuvor definiert haben. Der dritte Parameter ist die Taktfrequenz.

```
I2CBME.begin(I2C_SDA, I2C_SCL, 400000);
```

Schließlich initialisieren Sie ein BME280-Objekt mit Ihrer Sensoradresse und Zwei Wire Objekt.

```
status = bme.begin(0x76, &I2CBME);
```

Danach können Sie die üblichen Methoden auf Ihrem bme objekt, um Temperatur, Feuchtigkeit und Druck zu verlangen.

Hinweis: wenn die Bibliothek, die Du verwendest, eine Aussage wie wire.begin() In der Datei müssen Sie diese Zeile möglicherweise kommentieren, damit Sie Ihre eigenen Pins verwenden können.

ESP32 mit mehreren I2C Geräten

Wie wir bereits erwähnt haben, hat jedes I2C-Gerät seine eigene Adresse, so dass es mehrere I2C-Geräte im selben Bus haben kann.

Mehrere I2C Geräte (gleicher Bus, verschiedene Adressen)

Wenn wir mehrere Geräte mit verschiedenen Adressen haben, ist es trivial, wie man sie einrichtet:

- beide Peripheriegeräte an die ESP32 SCL und SDA anschließen;
- im Code, beziehen Sie sich auf jedes Randgebiet durch seine Adresse;

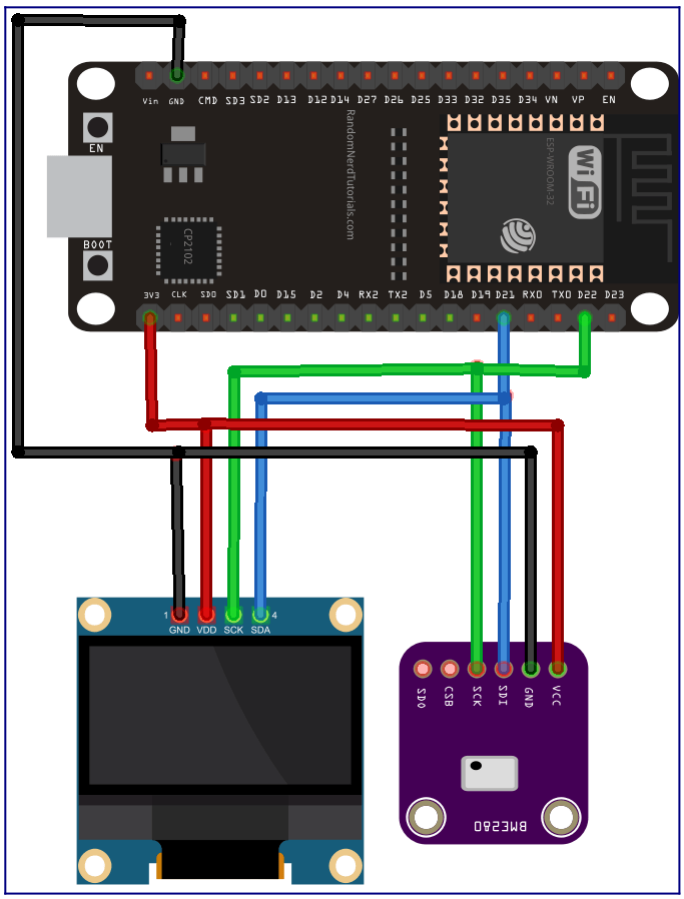
Werfen Sie einen Blick auf das folgende Beispiel, das Sensorwerte von einem BME280-Sensor (via I2C) erhält und die Ergebnisse auf einem I2C OLED-Display anzeigt.

```
/*

  Rui Santos

  Complete project details at https://RandomNerdTutorials.com/esp32-i2c-
communication-arduino-ide/



  Permission is hereby granted, free of charge, to any person obtaining a copy

  of this software and associated documentation files.



  The above copyright notice and this permission notice shall be included in all

  copies or substantial portions of the Software.

*/



#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_BME280.h>



#define SCREEN_WIDTH 128 // OLED display width, in pixels

#define SCREEN_HEIGHT 64 // OLED display height, in pixels
```

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

Adafruit_BME280 bme;

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

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }

  bool status = bme.begin(0x76);
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }

  delay(2000);
  display.clearDisplay();
  display.setTextColor(WHITE);
}

void loop() {
  display.clearDisplay();
  // display temperature
  display.setTextSize(1);
  display.setCursor(0,0);
```

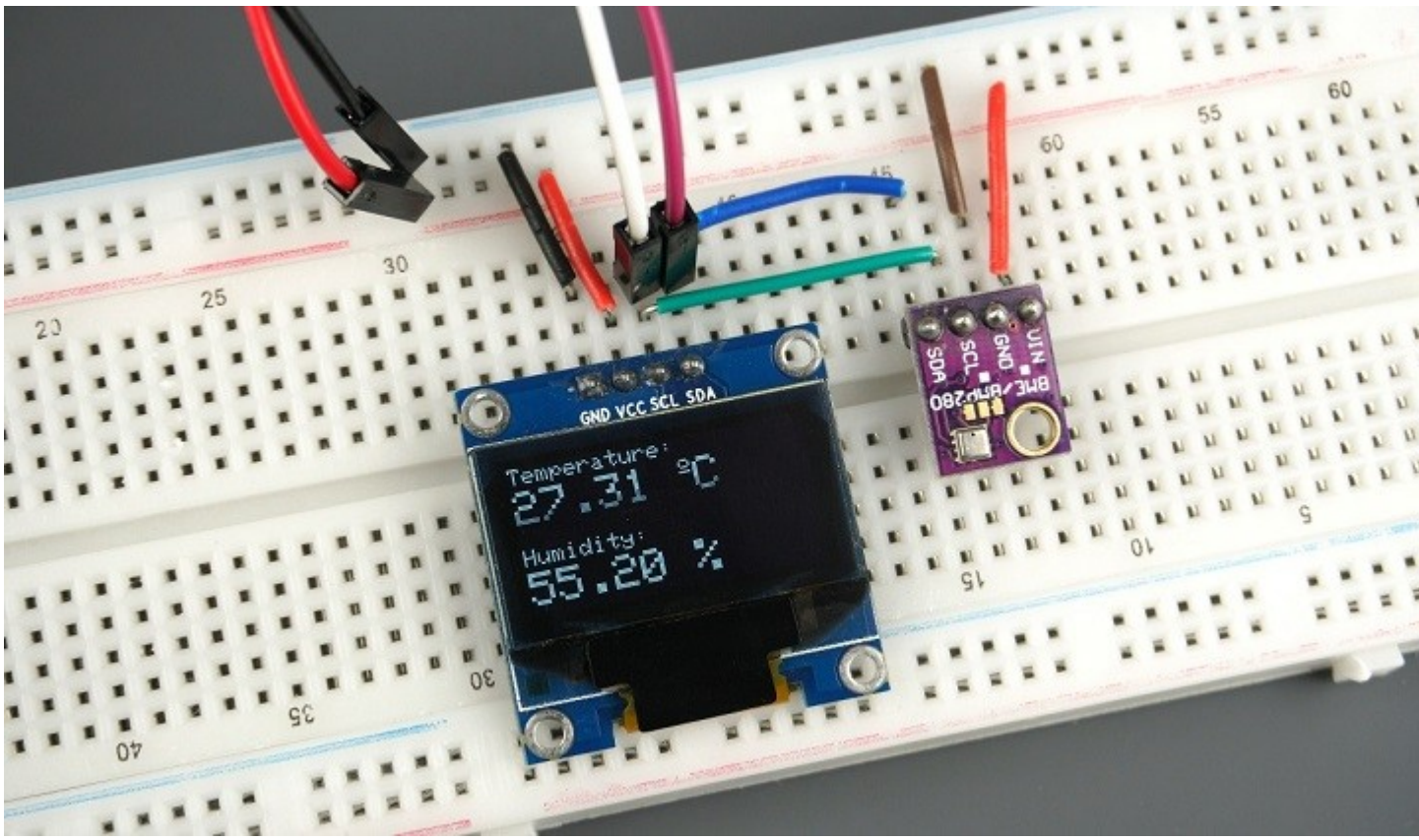
```
  display.print("Temperature: ");
  display.setTextSize(2);
  display.setCursor(0,10);
  display.print(String(bme.readTemperature()));
  display.print(" ");
  display.setTextSize(1);
  display.cp437(true);
  display.write(167);
  display.setTextSize(2);
  display.print("C");

  // display humidity
  display.setTextSize(1);
  display.setCursor(0, 35);
  display.print("Humidity: ");
  display.setTextSize(2);
  display.setCursor(0, 45);
  display.print(String(bme.readHumidity()));
  display.print(" %");

  display.display();

  delay(1000);
}
```

View raw code



Da die OLED und der BME280 unterschiedliche Adressen haben, können wir problemlos die gleichen SDA- und SCL-Leitungen verwenden. Die OLED-Anzeigeadresse ist 0x3C und die BME280 Adresse 0x76.

```
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
  Serial.println(F("SSD1306 allocation failed"));
  for(;;);
}

bool status = bme.begin(0x76);
if (!status) {
  Serial.println("Could not find a valid BME280 sensor, check wiring!");
  while (1);
}
```



Empfohlene Lese: [ESP32 OLED Display mit Arduino IDE]

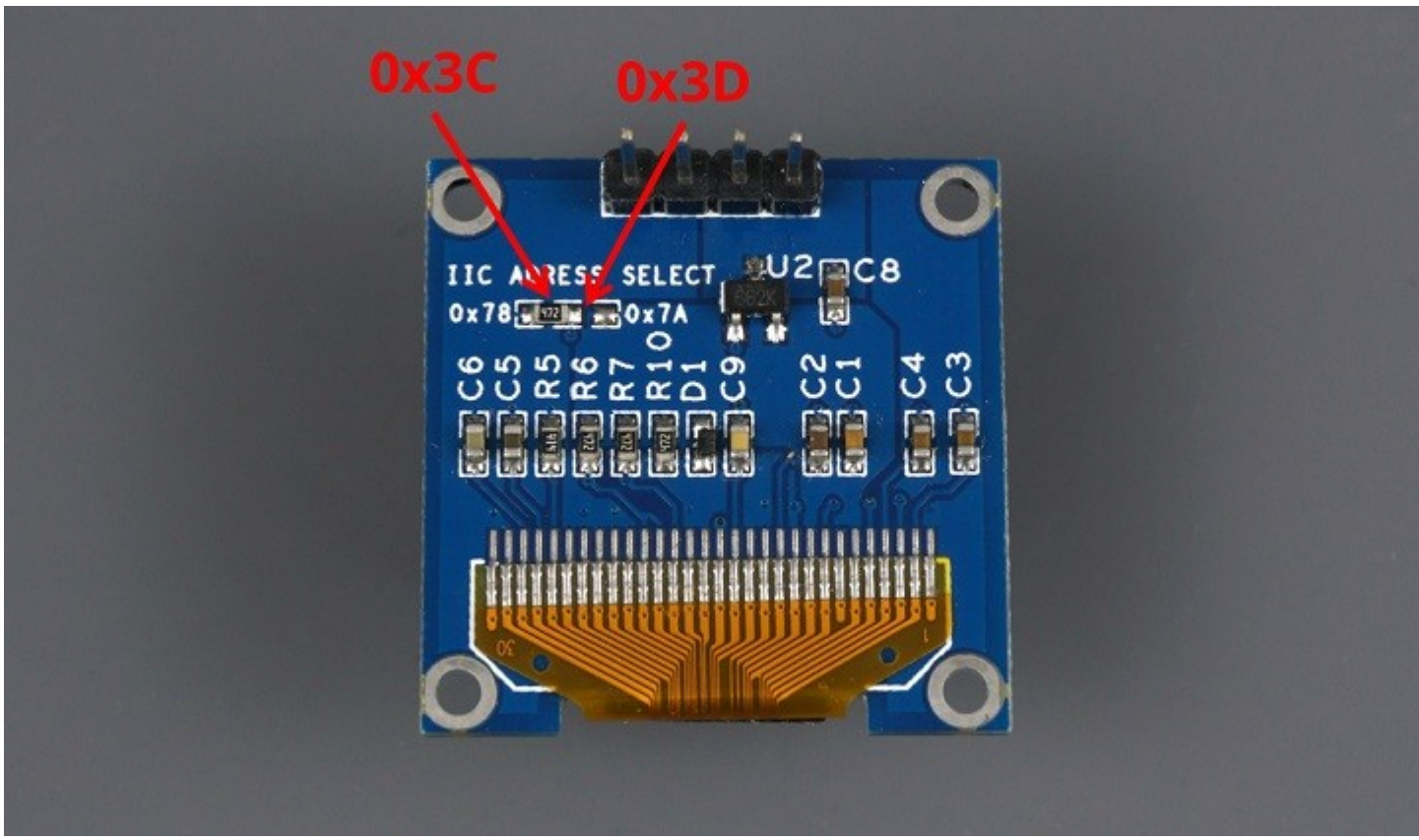
Mehrere I2C Geräte (gleiche Adresse)

Aber was, wenn Sie mehrere Peripheriegeräte mit der gleichen Adresse haben? Zum Beispiel mehrere OLED-Displays oder mehrere BME280-Sensoren? Es gibt mehrere Lösungen.

- Ändern der I2C-Adresse des Geräts;
- Verwenden Sie einen I2C Multiplexer.

Ändern der I2C-Adresse

Viele Breakout-Boards haben die Möglichkeit, die I2C-Adresse je nach Schaltung zu ändern. Zum Beispiel, dass ein Blick auf das folgende OLED-Display.

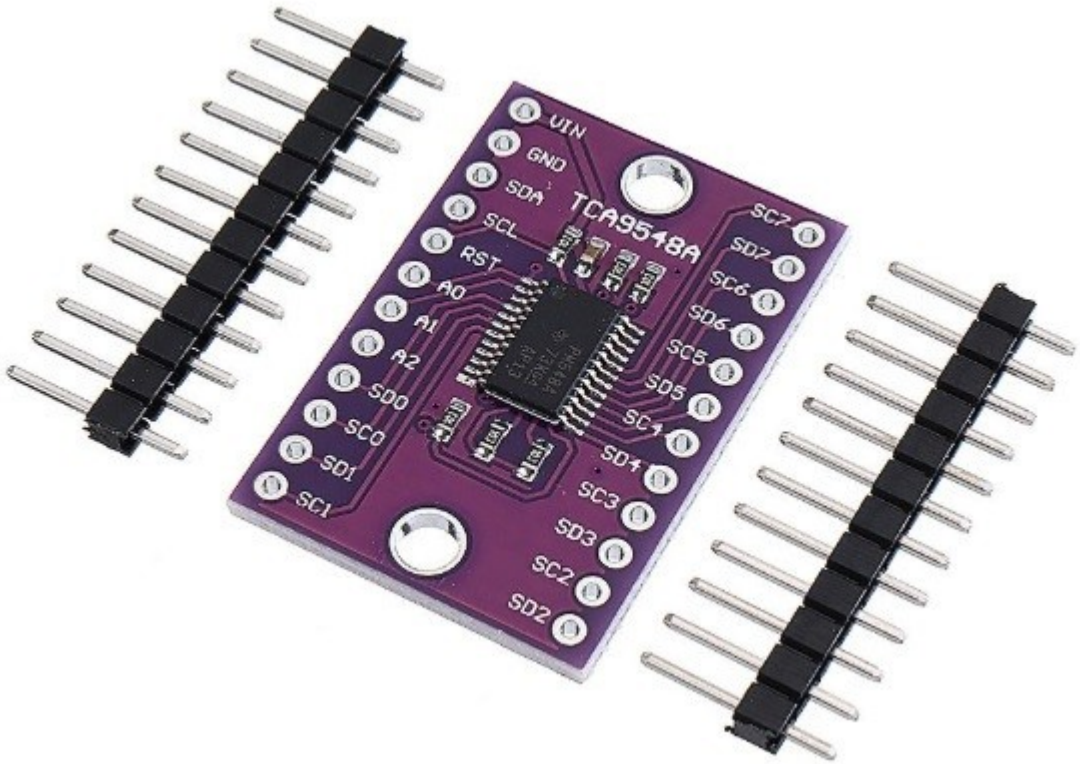


Durch die Platzierung des Widerstands auf der einen oder anderen Seite können Sie verschiedene I2C-Adressen auswählen. Das gilt auch für andere Komponenten.

Mit einem I2C Multiplexer

In diesem vorherigen Beispiel können Sie jedoch nur zwei I2C-Displays im selben Bus haben: eines mit 0x3C-Adresse und ein anderes mit 0x3D-Adresse.

Darüber hinaus ist es manchmal nicht trivial, die I2C-Adresse zu ändern. Um mehrere Geräte mit derselben Adresse im selben I2C-Bus zu haben, können Sie einen I2C-Multiplexer wie den TCA9548A verwenden, mit dem Sie mit bis zu 8 Geräten mit derselben Adresse kommunizieren können.



Wir haben ein detailliertes Tutorial, das erklärt, wie man einen I2C Multiplexer verwendet, um mehrere Geräte mit der gleichen Adresse mit dem ESP32 zu verbinden: Anleitung für TCA9548A I2C Multiplexer: ESP32, ESP8266, Arduino.

ESP32 mit zwei I2C Busschnittstellen

Um die beiden I2C-Busschnittstellen des ESP32 zu nutzen, müssen Sie zwei erstellen Zwei Wire Instanzen.

```
TwoWire I2Cone = TwoWire(0);
TwoWire I2Ctwo = TwoWire(1)
```

Dann initialisieren Sie die I2C-Kommunikation auf Ihren gewünschten Pins mit definierter Frequenz.

```
void setup() {
  I2Cone.begin(SDA_1, SCL_1, freq1);
  I2Ctwo.begin(SDA_2, SCL_2, freq2);
}
```

Dann können Sie die Methoden aus der Wire.h Bibliothek, um mit den I2C-Busschnittstellen zu interagieren.

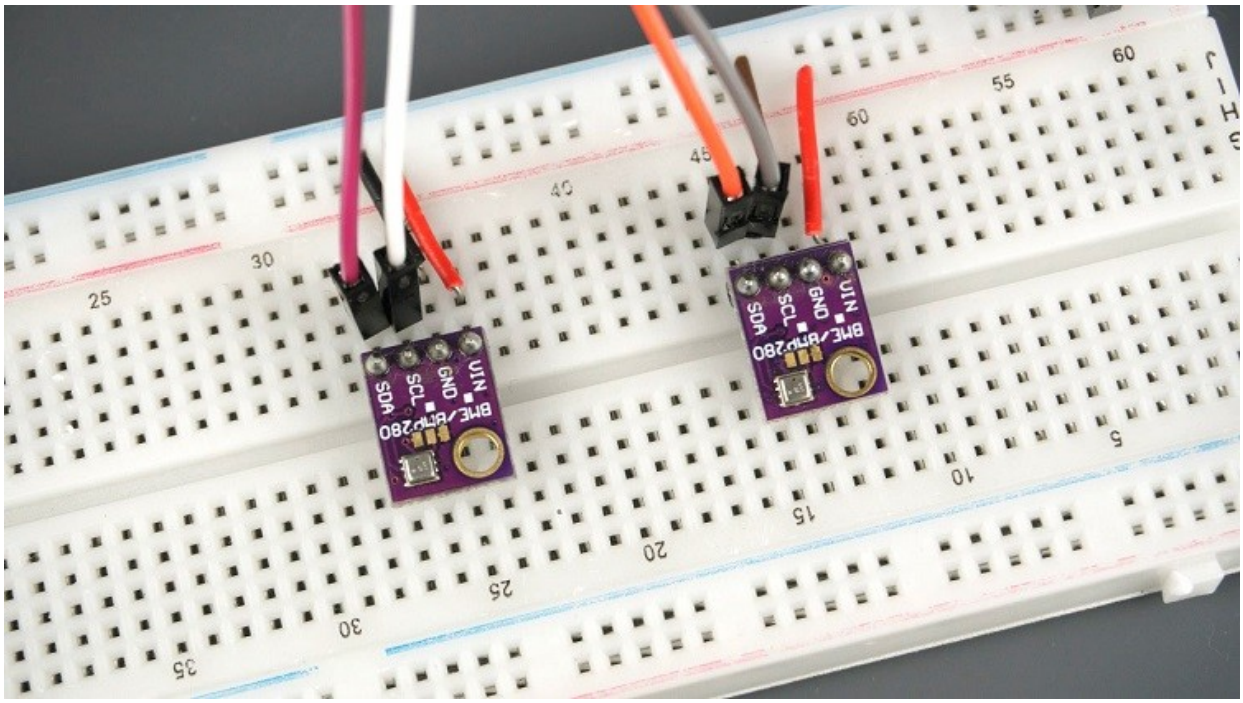
Eine einfachere Alternative ist die vordefinierte Wire() und Wire1() Objekte. Wire().begin() erzeugt eine I2C-Kommunikation im ersten I2C-Bus mit den Standardstiften und der Standardfrequenz. Für Wire1.begin() Sie sollten Ihre gewünschten SDA- und SCL-Pins sowie die Frequenz weitergeben.

```
setup(){
  Wire.begin(); //uses default SDA and SCL and 100000HZ freq
  Wire1.begin(SDA_2, SCL_2, freq);
```

```
}
```

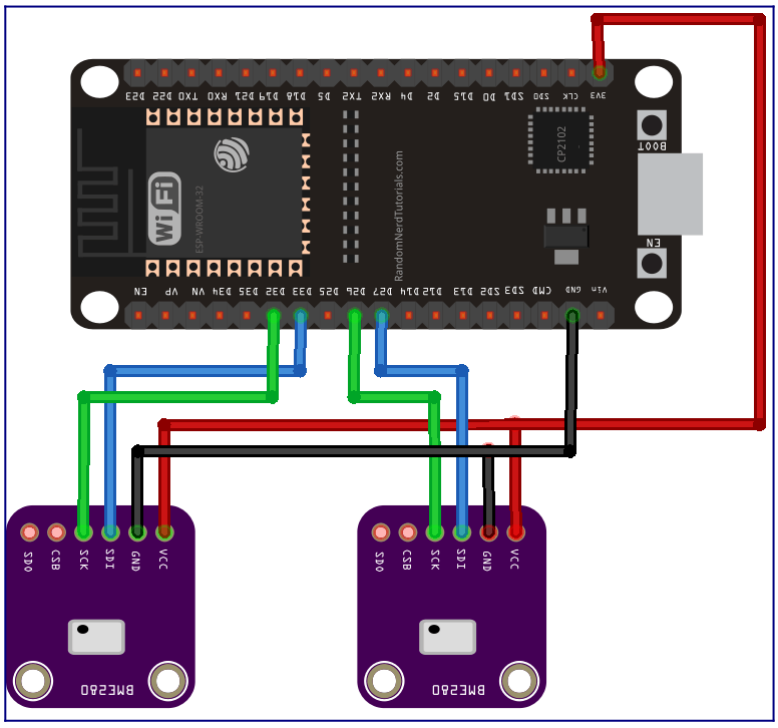
Diese Methode ermöglicht es Ihnen, zwei I2C-Busse zu verwenden, einer davon verwendet die Standardparameter.

Um besser zu verstehen, wie das funktioniert, werfen wir einen Blick auf ein einfaches Beispiel, das Temperatur, Feuchtigkeit und Druck von zwei BME280-Sensoren liest.



Jeder Sensor ist an einen anderen I2C-Bus angeschlossen.

- I2C Bus 1: Nutzung GPIO 27 (SDA) und GPIO 26 (SCL);
- I2C Bus 2: Nutzung GPIO 33 (SDA) und GPIO 32 (SCL);




```
/*
```

```
  Rui Santos

  Complete project details at https://RandomNerdTutorials.com/esp32-i2c-
communication-arduino-ide/



  Permission is hereby granted, free of charge, to any person obtaining a copy

  of this software and associated documentation files.



  The above copyright notice and this permission notice shall be included in all

  copies or substantial portions of the Software.

*/



#include <Wire.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_BME280.h>



#define SDA_1 27

#define SCL_1 26



#define SDA_2 33

#define SCL_2 32



TwoWire I2Cone = TwoWire(0);

TwoWire I2Ctwo = TwoWire(1);



Adafruit_BME280 bme1;

Adafruit_BME280 bme2;



void setup() {
```

```
  Serial.begin(115200);
  Serial.println(F("BME280 test"));

  I2Cone.begin(SDA_1, SCL_1, 100000);
  I2Ctwo.begin(SDA_2, SCL_2, 100000);

  bool status1 = bme1.begin(0x76, &I2Cone);
  if (!status1) {
    Serial.println("Could not find a valid BME280_1 sensor, check wiring!");
    while (1);
  }

  bool status2 = bme2.begin(0x76, &I2Ctwo);
  if (!status2) {
    Serial.println("Could not find a valid BME280_2 sensor, check wiring!");
    while (1);
  }

  Serial.println();
}

void loop() {
  // Read from bme1
  Serial.print("Temperature from BME1= ");
  Serial.print(bme1.readTemperature());
  Serial.println(" *C");

  Serial.print("Humidity from BME1 = ");
```

```
  Serial.print(bme1.readHumidity());
  Serial.println(" %");

  Serial.print("Pressure from BME1 = ");
  Serial.print(bme1.readPressure() / 100.0F);
  Serial.println(" hPa");

  Serial.println("--------------------");

  // Read from bme2
  Serial.print("Temperature from BME2 = ");
  Serial.print(bme2.readTemperature());
  Serial.println(" *C");

  Serial.print("Humidity from BME2 = ");
  Serial.print(bme2.readHumidity());
  Serial.println(" %");

  Serial.print("Pressure from BME2 = ");
  Serial.print(bme2.readPressure() / 100.0F);
  Serial.println(" hPa");

  Serial.println("--------------------");

  delay(5000);
}
```

View raw code

Werfen wir einen Blick auf die relevanten Teile, um die beiden I2C-Busschnittstellen zu nutzen.

Definieren Sie die SDA- und SCL-Pins, die Sie verwenden möchten:

```
#define SDA_1 27
#define SCL_1 26

#define SDA_2 33
#define SCL_2 32
```

Zwei schaffen Zwei Wire Objekte (zwei I2C Busschnittstellen):

```
TwoWire I2Cone = TwoWire(0);
TwoWire I2Ctwo = TwoWire(1);
```

Erstellen Sie zwei Instanzen Adafruit-BME280 Bibliothek, um mit Ihren Sensoren zu interagieren: bme1 und bme2.

```
Adafruit_BME280 bme1;
Adafruit_BME280 bme2;
```

Initialisieren Sie eine I2C-Kommunikation über die definierten Pins und Frequenz:

```
I2Cone.begin(SDA_1, SCL_1, 100000);
I2Ctwo.begin(SDA_2, SCL_2, 100000);
```

Dann sollten Sie bme1 und bme2 Objekte mit der richtigen Adresse und I2C Bus. bme1 Verwendung I2Cone:

```
bool status = bme1.begin(0x76, &I2Cone);
```

Und bme2 Verwendung I2Ctwo:

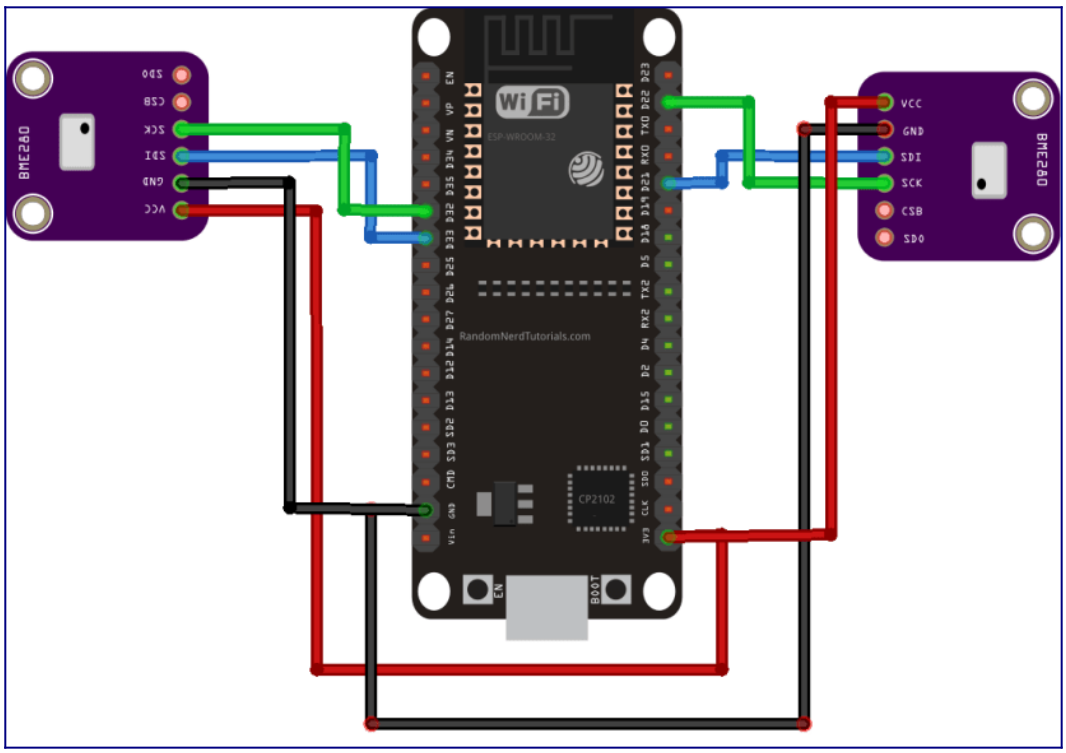
```
bool status1 = bme2.begin(0x76, &I2Ctwo);
```

Jetzt können Sie die Methoden aus dem Adafruit-BME280 Bibliothek auf Ihrer bme1 und bme2 Objekte zum Ablesen von Temperatur, Feuchtigkeit und Druck.

Eine weitere Alternative

Einfachheitlich können Sie die vordefinierten Wire() und Wire1() Objekte:

- Wire(): erstellt einen I2C-Bus auf den Standardstiften GPIO 21 (SDA) und GPIO 22 (SCL)
- Wire1(SDA 2, SCL 2, freq): erstellt einen I2C-Bus auf dem definierten SDA 2 und SCL 2 Pins mit der gewünschten Frequenz.



Hier ist das gleiche Beispiel, aber mit dieser Methode. Jetzt verwendet einer Ihrer Sensoren die Standardstifte, und der andere verwendet GPIO 32 und GPIO 33.

```
/*

  Rui Santos

  Complete project details at https://RandomNerdTutorials.com/esp32-i2c-communication-arduino-ide/



  Permission is hereby granted, free of charge, to any person obtaining a copy

  of this software and associated documentation files.



  The above copyright notice and this permission notice shall be included in all

  copies or substantial portions of the Software.

*/



#include <Wire.h>

#include <Adafruit_Sensor.h>

#include <Adafruit_BME280.h>
```

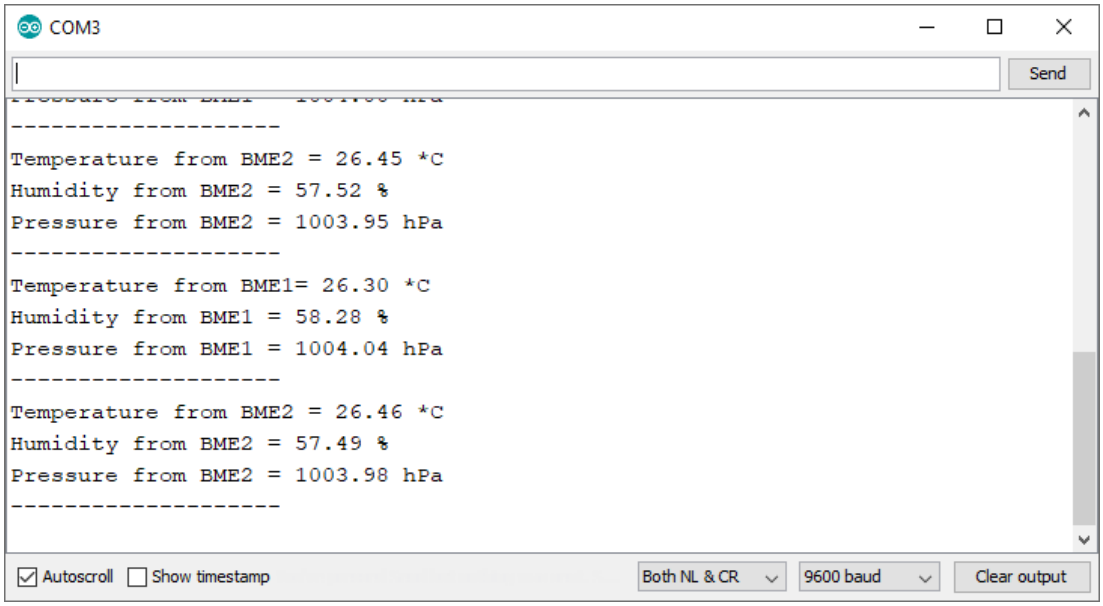
```
#define SDA_2 33
#define SCL_2 32

Adafruit_BME280 bme1;
Adafruit_BME280 bme2;

void setup() {
  Serial.begin(115200);
  Serial.println(F("BME280 test"));

  Wire.begin();
  Wire1.begin(SDA_2, SCL_2);

  bool status1 = bme1.begin(0x76);
  if (!status1) {
    Serial.println("Could not find a valid BME280_1 sensor, check wiring!");
    while (1);
  }

  bool status2 = bme2.begin(0x76, &Wire1);
  if (!status2) {
    Serial.println("Could not find a valid BME280_2 sensor, check wiring!");
    while (1);
  }

  Serial.println();
}
```

```
void loop() {
  // Read from bme1
  Serial.print("Temperature from BME1= ");
  Serial.print(bme1.readTemperature());
  Serial.println(" *C");

  Serial.print("Humidity from BME1 = ");
  Serial.print(bme1.readHumidity());
  Serial.println(" %");

  Serial.print("Pressure from BME1 = ");
  Serial.print(bme1.readPressure() / 100.0F);
  Serial.println(" hPa");

  Serial.println("--------------------");

  // Read from bme2
  Serial.print("Temperature from BME2 = ");
  Serial.print(bme2.readTemperature());
  Serial.println(" *C");

  Serial.print("Humidity from BME2 = ");
  Serial.print(bme2.readHumidity());
  Serial.println(" %");

  Serial.print("Pressure from BME2 = ");
  Serial.print(bme2.readPressure() / 100.0F);
  Serial.println(" hPa");
```

```
  Serial.println("--------------------");

  delay(5000);

}
```

View raw code

Sie sollten beide Sensorwerte auf Ihrem Serial Monitor erhalten.

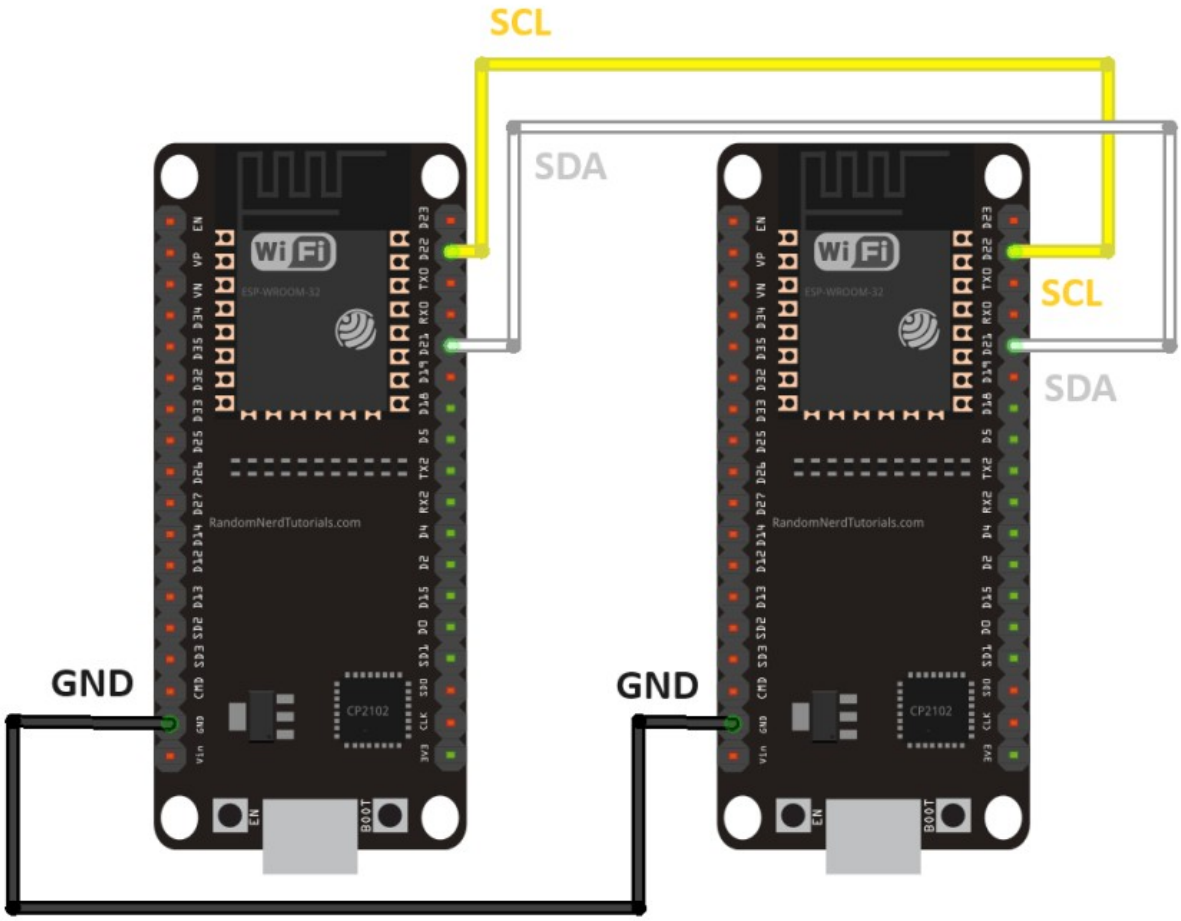


ESP32 I2C Master und Slave (I2C Kommunikation zwischen zwei ESP32)

Sie können Daten zwischen zwei ESP32-Boards über das I2C-Kommunikationsprotokoll austauschen. Ein Board muss als Sklave und ein anderes als Hauptgerät eingestellt werden.

Die Slave-Platte enthält die Daten, die das Hauptgerät anfordern kann.

Sie sollten beide ESP32-Platten mit ihren jeweiligen I2C-Pins überspannen. Vergessen Sie nicht, die GND-Pins miteinander zu verbinden.



Das vorherige Bild zeigt die Verkabelung für zwei ESP32 DOIT V1 Boards. Die Standard-I2C-Pins sind GPIO 21 (SDA) und GPIO 22 (SCL). Wenn Sie ein ESP32-C3, ESP32-S3 oder ein anderes Modell verwenden, könnten die Standard-I2C-Pins anders sein. Bitte überprüfen Sie die Pinout für das Board, das Sie verwenden. Sie können auch benutzerdefinierte I2C-Pins verwenden.

Für einen vollständigen Leitfaden zum Datenaustausch zwischen zwei ESP32-Boards mit dem I2C-Kommunikationsprotokoll überprüfen Sie den folgenden Link:

- ESP32 I2C Master und Slave (I2C Kommunikation zwischen zwei ESP32) – Arduino IDE

Einpacken

In diesem Tutorial erfahren Sie mehr über das I2C-Kommunikationsprotokoll mit dem ESP32. Wir hoffen, dass Sie die Informationen in diesem Artikel nützlich gefunden haben.

Um mehr über die ESP32 GPIOs zu erfahren, lesen Sie unseren Referenzführer: ESP32 Pinout Reference: Welche GPIO-Pins sollten Sie verwenden?

Erfahren Sie mehr über das ESP32 mit unseren Ressourcen:

- **ESP32 mit Arduino IDE (Videokurs + eBook)**
- MicroPython Programmierung mit ESP32 und ESP8266 (eBook)
- Mehr ESP32 Tutorials...