Workshop
Arduino-Programmierung #7

I^2C, BME280, request/receive, byte arrays

Joachim Baur

E-Mail: post@joachimbaur.de

ZTL-Alias: @joachimbaur

Download für diesen Workshop: www.joachimbaur.de/WS7.zip

I^2C Datenbus

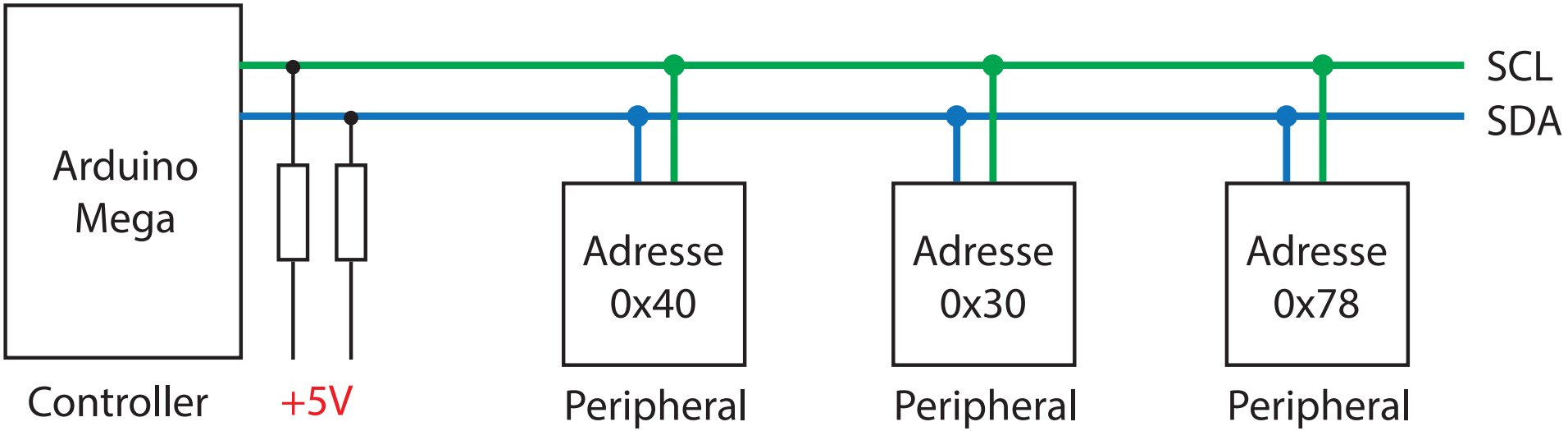
- „Inter-Integrated-Circuit“-Datenbus von Philips, in den 80ern entwickelt
- Verwendet 2 Leitungen („SCL“ für „Serial CLock“ = Takt, und „SDA“ für „Serial DAta“ = serielle Datenbits)
- Für kürzere Leitungslängen ausgelegt (< 1m)
- Benutzt Pull-Up-Widerstände an der SCL- und SDA-Leitung (sind beim Arduino eingebaut in Hardware I2C)
- Wird im Code mittels der „Wire“-Bibliothek umgesetzt

https://www.arduino.cc/reference/en/language/functions/communication/wire/

https://docs.arduino.cc/learn/communication/wire

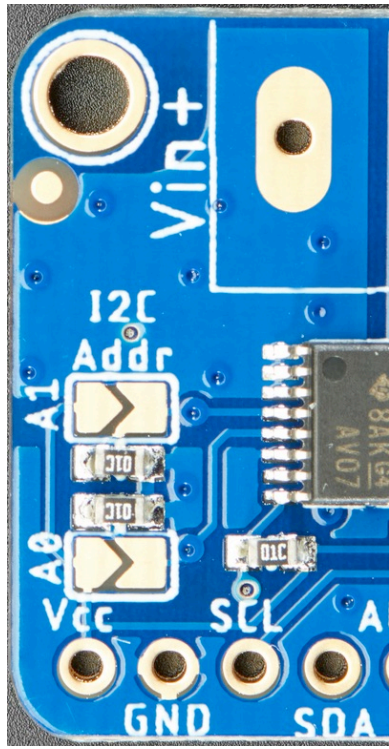
I2C Bus

- Der Bus läuft mit 100 kHz bzw. 400 kHz („Fast Mode“)
- In jedem I2C-Verbund gibt es nur 1 Controller
- Der Controller kann bis zu 112 Peripherals ansprechen
- Jedes Peri hat seine eigene Adresse und reagiert nur auf Befehle, die an diese Adresse gesendet werden

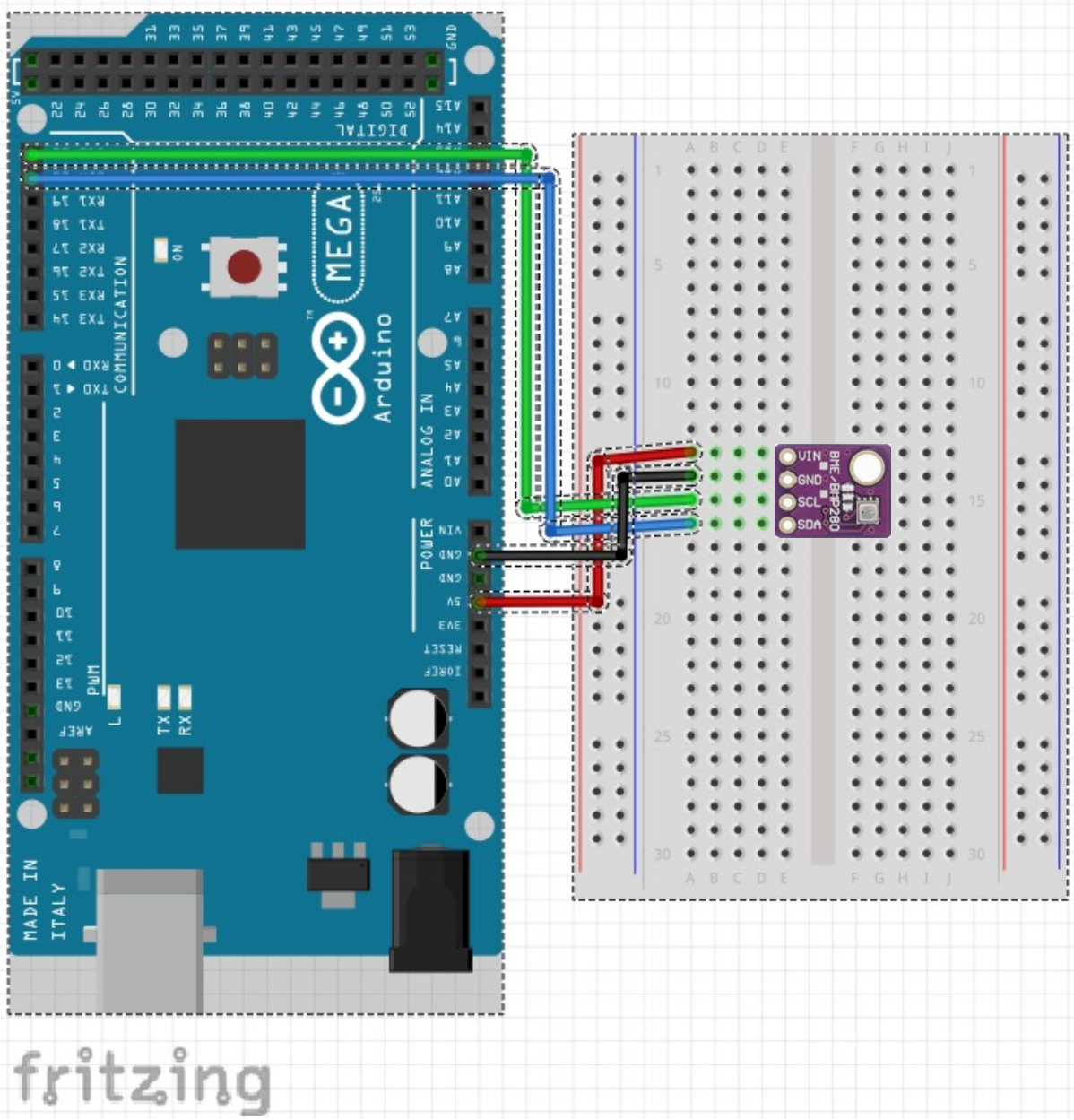


I2C Kommunikation

- Der **Controller** kann Daten an die Peris **senden** („send“, z.B. RGB-Werte für LEDs) oder Daten von diesen **anfordern** („request“, z.B. Sensormesswerte)
- Bei vielen Boards sind die **I2C-Adressen** mit Lötpads/ Jumpern änderbar, damit mehrere Sensoren des gleichen Typs am selben Bus verwendet werden können
- Da alle Peripherals am selben Bus hängen, kann immer nur 1 Gerät (Controller bzw. Peripheral) Daten senden (= relativ langsam).
- Die Anzahl der Daten für 1 send/request-Vorgang ist in der Arduino Wire-Library auf 32 Bytes begrenzt.
- ACHTUNG: Viele I2C-Sensoren verwenden eine Logik-Spannung von nur 3,3 Volt.



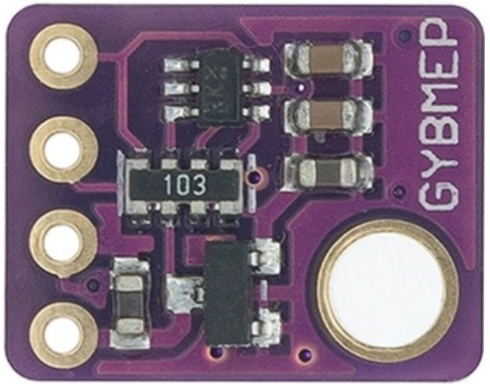
I2C Sensor anschließen



Sensor BME280
für Temperatur, Luftfeuchtigkeit und Luftdruck

VIN = 5V
GND
SCL
SDA

SDA
SCL
GND
VIN = 5V



I2C Scanner

- Sketch, der die Adressen (I2C IDs) aller am Mega angeschlossenen Peripherals in der seriellen Konsole ausgibt
- https://playground.arduino.cc/Main/I2cScanner/

Sketch: 32_I2CScanner.ino

```
#include <Wire.h>

void setup() {
    Wire.begin(); // ohne Adresse in Klammern = Controller
}

void loop() {
    for(address = 1; address < 127; address++ ) {
        Wire.beginTransmission(address);

        // The i2c_scanner uses the return value of the Write.endTransmission
        // to see if a device did respond to the address
        byte error = Wire.endTransmission();

        if (error == 0) Serial.println("I2C device found: 0x" + String(address,HEX));
    }
}
```

Adafruit BME280 Library

- Library installieren
- Datei -> Beispiele -> Adafruit BME280 -> bme280test

Sketch: 33_BME280_Test.ino

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

#define I2C_ADRESSE 0x76 // von I2C Scanner ermittelt
#define SEALEVELPRESSURE_HPA (1013.25)

Adafruit_BME280 bme; // C++ Klasse, kein struct wie CRGB

void setup() {
    Serial.begin(115200);
    bme.begin( I2C_ADRESSE );
}

void loop() {
    printValues();
    delay(1000);
}

void printValues() {
    Serial.print("Temperature = "); // etc
}
```

I2C senden von Texten

Sketch: 34_I2C_SendeText.ino

```
#include <Wire.h>

const int I2C_EMPFAENGER = 0x40; // I2C-Adresse des Empfängers
String eingabe = "";

void setup() {
    Serial.begin( 115200 ); // für Eingabe per serieller Konsole
    Wire.begin(); // I2C als Controller starten
}

void loop() {
    eingabe = "";
    while (Serial.available() > 0) {
        eingabe += char(Serial.read());
        delay(1);
    }
    if (eingabe != "") sendeI2C( eingabe );
}

void sendeI2C( String text ) {
    Wire.beginTransmission( I2C_EMPFAENGER );
    Wire.write(text.c_str()); // kein Arduino String Objekt!
    Wire.endTransmission();
    Serial.println("Gesendet I2C: " + text);
}
```

I2C Empfangen von Texten

- Die von der Hardware I2C-Schnittstelle empfangenen Daten werden wie bei Serial mit `Wire.available()` und `Wire.read()` aus dem Buffer ausgelesen.
- Sobald Daten per I2C gelesen wurden, wird ein sogenannter Interrupt ausgelöst, der den laufenden regulären Programmfluss in `loop()` unterbricht
- ACHTUNG: In der Interrupt-Funktion so wenig Code wie möglich und auf KEINEN FALL Serial-Funktionen (diese benutzen wiederum selbst Interrupts -> Riesenchaos und Fehler…)

Sketch: 35_I2C_EmpfangeText.ino

```
#include <Wire.h>

const int I2C_ADRESSE = 0x40; // unsere I2C-Adresse als Empfänger-Peripheral

bool I2C_empfangen = false;
String I2C_message = ""; // Text der empfangenen Nachricht

void setup() {
    Serial.begin( 115200 );
    Wire.begin( I2C_ADRESSE ); // I2C als Peripeheral starten
    Wire.onReceive(receiveEvent); // einen "interrupt handler" einrichten
}

void loop() {
    if (I2C_empfangen) {
        Serial.println("Empfangen I2C: " + I2C_message);
        I2C_empfangen = false; // Flag wieder zurücksetzen
    }
}

void receiveEvent(int numBytes) {
    I2C_message = "";
    while (Wire.available()) {
        I2C_message += char(Wire.read());
    }
    if (I2C_message.length() > 0) I2C_empfangen = true;
}
```

I2C-Komm zwischen 2 Megas

- Aufgabe: 2 Megas per I2C verbinden (Leitungen SCL, SDA, GND)
- Ein Mega als Controller, den anderen als Peripheral flashen
- Eingabe der seriellen Konsole des Controllers per I2C zum Peripheral senden
- Das Peripheral gibt den per I2C empfangenen Text wieder in seiner seriellen Konsole aus

Controller: Sketch: 34_I2C_SendeText.ino

Peripheral: Sketch: 35_I2C_EmpfangeText.ino

I2C Daten anfordern (Controller)

- Der I2C-**Controller** kann nicht nur Daten senden, sondern muss auch Daten von den Peripherals lesen können (Sensor-Messwerte etc)
- Dazu wird ein "**request**" (Anfrage)-Befehl vom Controller an ein Peripheral geschickt (im Prinzip ein I2C-Header ohne Daten)
- Das Peripheral sendet daraufhin seine Daten über I2C zurück an den Controller
- Peripherals können **ohne** einen request des Controllers **nicht** selbstständig Daten senden (müssen also zB periodisch abgefragt werden)!

Sketch: 36_I2C_RequestData.ino

```
#include <Wire.h>

const int I2C_EMPFAENGER = 0x40;

String I2C_message = ""; // Text der empfangenen Nachricht

void setup() {
  Serial.begin(115200);
  Wire.begin(); // I2C als Controller starten
}

void loop() {
  I2C_message = "";

  // 5 bytes (bzw. Zeichen) vom Empfänger anfordern
  Wire.requestFrom(I2C_EMPFAENGER, 5);

  while (Wire.available()) {
    I2C_message += char(Wire.read());
  }

  if (I2C_message.length() > 0) Serial.println( I2C_message );

  delay(1000); // 1 Sekunde warten
}
```

I2C Daten "onRequest" senden

- Auf dem Peripheral richtet man einen Interrupt-Handler für "**onRequest**" ein, ähnlich wie "onReceive"

Sketch: 37_I2C_OnRequestAntwort.ino

```
#include <Wire.h>

const int I2C_ADRESSE = 0x40;

void setup() {};
  Wire.begin( I2C_ADRESSE ); // I2C als Peripeheral starten
  //Wire.onReceive(receiveEvent); // einen "interrupt handler" einrichten
  Wire.onRequest(requestEvent);
}

void loop() {
  delay(1); // hier passiert irgendwas oder gar nichts
}

void requestEvent() {
  Wire.print("Daten"); // 5 Bytes (Zeichen) senden
}
```

Daten als byte arrays übertragen

- Oft macht das Übertragen von Daten in Textform wenig Sinn, wenn es z.B. um Messwerte oder mehrere und größere Zahlen geht
- Zum Beispiel RGB-Werte (3 x 8 Byte, Werte jeweils von 0….255) oder große Zahlen wie 1000000
- Solche Daten werden in byte arrays übertragen und müssen dann auch wieder in byte arrays eingelesen werden
- RGB-Werte als String: "255,255,255" (11 Byte)
- RGB-Werte als Byte Array: [255, 255, 255] (3 Byte)

Sketch: 38_I2C_SendeByteArray.ino

```
#include <Wire.h>

const int I2C_EMPFAENGER = 0x40;

byte dataArray[3] = { 0, 0, 0 };
int zaehler = 0;

void setup() {
  Wire.begin(); // controller
}

void loop() {
  // dataArray mit 3x zaehler befüllen
  dataArray[0] = zaehler;
  dataArray[1] = zaehler;
  dataArray[2] = zaehler;
  // per I2C an peripheral senden
  sendeI2C();
  zaehler++;
  delay(1000); // 1 Sekunde warten
}

void sendeI2C() {
  Wire.beginTransmission( I2C_EMPFAENGER );
  Wire.write( dataArray, 3 );
  Wire.endTransmission();
}
```

I2C ByteArray empfangen

Sketch: 39_I2C_EmpfangeByteArray.ino

```
#include <Wire.h>

const int I2C_ADRESSE = 0x40
bool I2C_empfangen = false;
byte dataArray[3] = { 0, 0, 0 }; // empfangene Bytes

void setup() { /* wie bei Sketch 29_I2C_EmpfangeText */ }

void loop() { /* wie bei Sketch 29_I2C_EmpfangeText, nur println angepasst */ }

void receiveEvent(int numBytes) {
  int index = 0;

  while (Wire.available()) {
    dataArray[index] = Wire.read();
    index++;
    if (index == 3) break; // while-Schleife auf jeden Fall nach 3 Bytes beenden
  }

  if (index > 0) I2C_empfangen = true;
}
```

Byte array per Serial senden

- Auch mit der **seriellen** Schnittstelle können byte arrays statt reinen Texten übertragen werden:

I2C

```
Wire.write( dataArray, 3 );

dataArray[index] = Wire.read();
```

Serial

```
Serial.write( dataArray, 3 );

dataArray[index] = Serial.read();
```

https://www.arduino.cc/reference/en/language/functions/communication/serial/write/