Arduino verstehen und anwenden

Teil 22: I²C – der Inter-IC-Bus – Grundlagen und Anwendungen

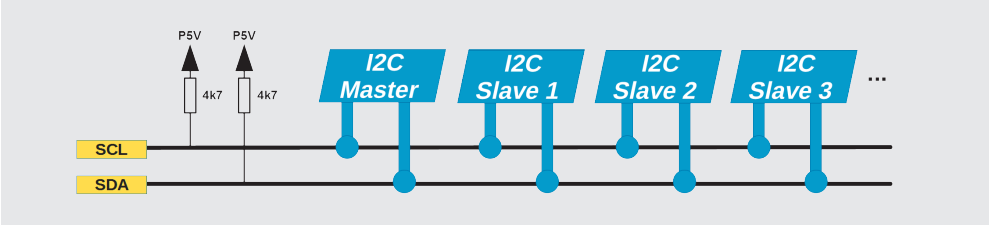

Bild 1: Prinzipieller Aufbau des I²C-Busses

Der Arduino verfügt zwar über 14 digitale I/O-Pins und diese sind für viele kleinere Projekte durchaus ausreichend. Bei größeren Praxisanwendungen stößt man jedoch schnell an Grenzen. Insbesondere wenn Shields, wie etwa das im letzten Beitrag vorgestellte MP3-Shield, benutzt werden, sind oft nur noch wenige Pins frei verfügbar, da die Shields selbst bereits viele Pins belegen.

Eine Lösung wäre, eine größere Arduino-Version einzusetzen, etwa den Arduino Mega oder den Due. Dies ist allerdings mit erheblichen Kosten verbunden und in vielen Fällen auch gar nicht notwendig.

Sogenannte Bus-Systeme bieten die Möglichkeit, umfangreiche Funktionen mit nur wenigen Pins zu steuern. Ein klassisches Beispiel ist die Verwendung eines I²C-Displays. Dieses benötigt nur zwei I/O-Pins im Gegensatz zu den sechs Pins, die ein klassisches HD44780-LCD belegt.

In diesem Beitrag soll daher der I²C-Bus im Vordergrund stehen. Weitere häufig verwendete Bus-Systeme wie etwa SPI oder One-Wire werden in späteren Beiträgen behandelt.

Der I²C-Bus

Der I²C-Bus (Inter Integrated Circuit, meist als „I-squared-C-Bus" oder deutsch „I-Quadrat-C-Bus" ausgesprochen) ist ein serieller Zweidraht-Bus, der vor über 30 Jahren in der Entwicklungsabteilung der Firma Philips, heute NXP, entstand. Rasch entwickelte sich das System zum Industriestandard für Steuerungs-, Diagnose- und Überwachungslösungen in unzähligen Embedded-Applikationen.

Durch die einfache Implementierung, niedrige Kosten und eine Übertragungsrate von bis zu 3,4 MBit/s hat der Bus bis heute nichts von seiner Aktualität verloren.

Ursprünglich sollte ein Bus-System entstehen, über das mehrere ICs auf einer Leiterplatte mit geringem Aufwand miteinander kommunizieren sollten. Daher wurde eine serielle Struktur bevorzugt, um mit wenigen Leiterbahnen auszukommen.

So entstand ein bidirektionaler Zweidraht-Bus in Master/Slave-Architektur mit integriertem Übertragungsprotokoll und Software-Adressierung, der nur zwei Verbindungsleitungen erfordert:

- die Taktleitung SCL (Serial Clock),
- die Datenleitung SDA (Serial Data).

Ein Mikrocontroller kann so ein ganzes Netzwerk von Komponenten mit nur zwei I/O-Pins ansteuern. Bevorzugte Anwendungen finden sich insbesondere im Bereich der Unterhaltungselektronik, z. B. in der Abstimmung von Autoradios oder TV-Geräten oder bei vollelektronischen Lautstärkereglern.

Zunächst genügte hierfür eine Übertragungsrate von lediglich 100 kBit/s. Aufgrund zunehmender Leistungsanforderungen wurde die Übertragungsrate dann aber immer weiter angehoben. Ein großer Vorteil des I²C-Busses besteht darin, dass auch langsamere Komponenten am Bus betrieben werden können. Hierfür kann das sogenannte Clock-Stretching eingesetzt werden (s. u.).

Inzwischen ist der I²C-Bus nicht mehr auf einzelne Platinen beschränkt. Vielmehr kommt er auch in größeren Systemen mit mehreren Boards zum Einsatz. Die einfache Steuersoftware macht den Bus sehr flexibel. Da keine festen Taktzeiten eingehalten werden müssen, können sowohl langsame als auch sehr schnelle Busteilnehmer simultan betrieben werden. Auf der Softwareseite ist so auch der Einsatz langsamerer Programmiersprachen wie etwa Python möglich.

Elektrischer Anschluss und Takt

I²C ist als Master-Slave-Bus konzipiert. Ein Datentransfer wird immer durch einen Master gestartet. Anschließend reagiert der über seine Adresse angesprochene Slave auf die Anfrage. Im sogenannten Multimaster-Mode können aber auch mehrere Master an einem Bus betrieben werden. Hier können dann auch zwei Master direkt miteinander kommunizieren, indem ein Gerät kurzzeitig als Slave arbeitet. Die Zugriffsregelung auf den Bus ist über eine entsprechende Spezifikation detailliert geregelt.

Bild 1 zeigt, wie die einzelnen Buskomponenten hardwaremäßig miteinander verbunden werden.
Beide Busleitungen (Takt und Daten) liegen mit den Pull-up-Widerständen (typischerweise 4,7 bis 10 kΩ) an der Versorgungsspannung Vdd, die üblicherweise 5 V beträgt. Sämtliche daran angeschlossene Geräte haben Open-Collector-Ausgänge. Zusammen mit den Pull-up-Widerständen ergibt sich so eine Wired-AND-Schaltung. Der High-Pegel der Bussignale sollte mindestens 0,7×Vdd betragen, der Low-Pegel soll höchstens bei 0,3×Vdd liegen. Falls es zu Übertragungsfehlern kommt, ist es immer eine gute Idee, wenn man diese Spannungen mit einem Oszilloskop nachmisst und eventuell die Pull-up-Widerstände anpasst. Einige Controller haben auch bereits interne Pull-ups eingebaut, sodass die externen Widerstände entfallen können. Im Zweifelsfall hilft hier ein Blick ins Datenblatt weiter.

Der I²C-Bus arbeitet mit positiver Logik. Das bedeutet, dass ein High-Pegel auf der Datenleitung einer logischen „Eins", der Low-Pegel einer „Null" entspricht.

Der Bustakt wird immer vom Master ausgegeben. Für die verschiedenen Modi ist jeweils ein maximal erlaubter Bustakt vorgegeben. In der Regel können

	Adresse				Geräte Subadresse			R/W		Daten								
Start	0	1	0	0	x	x	x	1	ACK	x	x	x	x	x	x	x	x	Stopp

Bild 2: Aufbau eines I²C-Telegramms

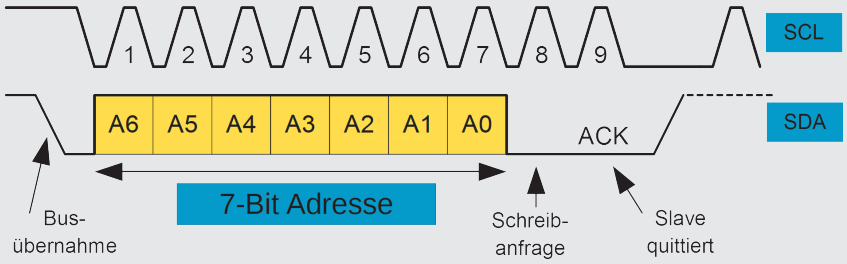

Bild 3: I²C-Timing

aber auch beliebig langsamere Taktraten verwendet werden, falls diese vom Master-Interface unterstützt werden. Einige ICs (z. B. Analog-digital-Umsetzer) benötigen jedoch eine bestimmte minimale Taktfrequenz, um ordnungsgemäß zu funktionieren. Die folgende Tabelle listet die gängigsten Taktraten auf:

100-kHz-Takt	Standard-Mode
400-kHz-Takt	Fast-Mode
1,0-MHz-Takt	Fast-Mode-Plus
3,4-MHz-Takt	High-Speed-Mode

Wenn der Slave mehr Zeit benötigt, als durch den Takt des Masters vorgegeben ist, kann er zwischen der Übertragung einzelner Bytes die Taktleitung auf „low" halten. Über dieses im letzten Abschnitt bereits erwähnte Clock-Stretching kann der Mastertakt reduziert werden, falls dies erforderlich sein sollte.

In einem I²C-System sind Daten nur gültig, wenn sich ihr logischer Pegel während einer Clock-High-Phase nicht ändert. Ausnahmen von dieser Regel sind das Start- und Stoppsignal. Das Startsignal wird durch eine fallende Flanke auf SDA gekennzeichnet, während SCL „high" ist. Das Stoppsignal dagegen wird durch eine steigende Flanke auf SDA signalisiert, die SCL-Leitung bleibt dabei auf High-Pegel.

Eine Dateneinheit besteht aus 8 Datenbits und wird häufig auch als Oktett bezeichnet. Ein Oktett kann entweder als Wert oder als Adresse interpretiert werden. Dazu kommt noch ein ACK-Bit als Bestätigung (von engl. Acknowledge). Dieses wird vom Slave durch einen Low-Pegel auf der Datenleitung erzeugt.

Protokoll und Adressierung

Eine Standard-I²C-Adresse ist das erste vom Master gesendete Byte, wobei die ersten sieben Bit die eigentliche Adresse darstellen und das achte Bit (R/W-Bit) dem Slave mitteilt, ob er Daten vom Master empfangen soll (LOW) oder Daten an den Master zu übertragen hat (HIGH). In I²C-Systemen ist daher ein Adressraum von 7 Bit verfügbar. Da 16 der 2^7 = 128 möglichen Adressen für Sonderzwecke reserviert sind, können so bis zu 112 Teilnehmer an einem Bus betrieben werden. Dazu hat jedes I²C-fähige IC eine vom Hersteller festgelegte Adresse, von der häufig drei Bits, die sogenannte Subadresse, über drei Steuerpins individuell festgelegt werden können (siehe Bild 2 und 3). In diesem Fall können bis zu acht ICs des gleichen Typs an einem I²C-Bus betrieben werden. Im Praxisteil wird dies anhand eines Temperatursensors vom Typ LM75 genauer erläutert.

Für Hobbyanwendungen sind 112 Teilnehmer sicherlich in den meisten Fällen ausreichend. Bei großen, professionellen Systemen dagegen kann es durchaus zu Adressknappheit kommen. Deshalb wurde eine 10-Bit-Adressierung eingeführt. Sie ist abwärtskompatibel zum 7-Bit-Standard durch Nutzung von 4 der 16 reservierten Adressen. Beide Adressierungsarten sind gleichzeitig verwendbar, sodass nun bis zu 1136 Komponenten auf einem Bus erlaubt sind.

Die Übertragung beginnt mit einem Startsignal des Masters. Dann folgt die Adresse. Diese wird durch das ACK-Bit des angesprochenen Slaves bestätigt. Abhängig vom R/W-Bit werden nun Daten byteweise geschrieben oder gelesen. Beim Schreiben wird das ACK vom Slave gesendet, beim Lesen vom Master. Eine Übertragung wird durch das Stoppsignal beendet.

Die Datenübertragung startet mit dem höchstwertigen Bit (MSB first). Für den High-Speed-Mode wird zuerst im Fast- oder Standard-Mode ein Master-Code geschickt, bevor auf die erhöhte Frequenz umgeschaltet wird.

Anwendungen

Einer der wichtigsten Vorteile von Bus-Systemen wie I²C ist, dass ein Mikrocontroller ein ganzes Netzwerk an integrierten Schaltungen mit nur zwei I/O-Pins und einfacher Software ansteuern kann.

Systeme mit einer möglichst geringen Anzahl von erforderlichen I/O-Pins sind deshalb so vorteilhaft, da ein erheblicher Teil der Kosten einer integrierten Schaltung und der verwendeten Leiterplatte von der Größe des IC-Gehäuses und der Anzahl der Pins abhängt. Ein großer Chip mit vielen Pins benötigt mehr Platz auf der Leiterplatte. Zudem stellen zusätzliche Verbindungen immer ein Ausfallrisiko dar und erfordern so einen erhöhten Prüfaufwand. All das steigert die Entwicklungs-, Produktions- und Testkosten.

Obwohl das System langsamer ist als neuere Busse, ist die I²C-Schnittstelle wegen des geringen Aufwands vorteilhaft für Peripheriegeräte, die keine hohen Datenraten erfordern. Häufig wird sie für die Übertragung von Steuer- und Konfigurationsdaten verwendet. Beispiele sind Lautstärkeregler, Analog-digital- oder Digital-analog-Wandler mit niedriger Abtastrate, Echtzeituhren, kleine, nichtflüchtige Speicher oder

bidirektionale Schalter und Multiplexer. Auch elektronische Sensoren haben oft einen Analog-digital-Wandler mit I²C-Schnittstelle integriert.

In der folgenden Tabelle sind einige der am meisten verwendeten I²C-fähigen Chips zusammengefasst.

IC / Komponente	Funktion
LM75	Temperatursensor
BMP180	Barometrischer Luftdrucksensor
DS1307	Echtzeituhr mit Batteriepufferung
MAX127	A/D-Wandler
PCF8574	Portexpander
PCF8583	Echtzeituhr
LC-I2C	4x14-Segment-LC-Display
AZDelivery-Display	OLED-Graphik-Display 0,96"
AT24C256	EEPROM

Natürlich ist diese Tabelle bei Weitem nicht vollständig, sie liefert jedoch einen guten Überblick über die wichtigsten und am häufigsten verwendeten I²C-Bausteine.

Ein weiterer Vorteil des Busses ist, dass Bauelemente während des Betriebs zum Bus hinzugefügt oder entfernt werden können (sog. Hot-Plug-Fähigkeit). Allerdings muss diese Eigenschaft durch besondere Vorkehrungen in der Programmierung berücksichtigt werden. Ansonsten kann es auch zu unerwünschten Programmabbrüchen kommen, wenn Elemente vom Bus getrennt werden.

Eine gewisse Bedeutung hat das I²C-Protokoll auch im Chipkartenbereich. So wurde das System für Krankenversichertenkarten eingesetzt. Unter den Kontaktflächen der Chipkarte befand sich ein einfaches I²C-EEPROM, das vom Kartenleser ausgelesen und beschrieben werden konnte. Allerdings werden hier zunehmend auch andere Technologien eingesetzt.

Sensoren am I²C-Bus

Zu den wichtigsten Anwendungen des I²C-Busses zählt die Erfassung von Sensorwerten. Hierfür steht eine fast unüberschaubare Vielfalt von Komponenten zur Verfügung. So lassen sich praktisch alle physikalischen Größen mit I²C-Sensoren erfassen. Im nichtprofessionellen Bereich haben sich allerdings einige Bauelemente besonders etabliert. Die wichtigsten davon sollen als Anwendungsbeispiele für das Bus-System im Folgenden vorgestellt werden.

Präzise Temperaturmessung

Eine der einfachsten Möglichkeiten, Temperaturen zu erfassen, ist die Verwendung von NTC-Sensoren (Negative Temperature Coefficient). Diese Methode ist allerdings mit mehreren Nachteilen verbunden.

Zunächst sind diese einfachen Bauelemente herstellungsbedingt mit großen Toleranzen behaftet. Deshalb muss man jeden Sensor individuell kalibrieren, wenn eine gewisse Genauigkeit erreicht werden soll. Was im Hobbybereich durchaus noch möglich ist, wäre bei professionellen Anwendungen viel zu aufwendig.

Ein weiterer Nachteil ist, dass die Messungen sehr ungenau werden können, sobald zwischen dem Sensor und dem Controller bzw. dem Arduino größere Distanzen liegen. Eine Ferntemperaturerfassung wird mit einfachen analogen Sensoren sehr unzuverlässig.

Thermospannungen, Leitungswiderstände oder elektromagnetische Einstreuungen führen zu erheblichen Messfehlern. Digitale Sensoren sind hier deutlich weniger störanfällig. Der Messwert wird damit direkt am Ort der Messung in ein digitales Signal umgewandelt. Dieses ist gegenüber den oben genannten Einflüssen wesentlich unempfindlicher. Auf diese Weise kann man auch größere Distanzen zwischen dem Sensor und einer zentralen Messstation überbrücken.

Ein weit verbreiteter Sensortyp mit integrierter A/D-Wandlung ist der LM75. Dieses Bauelement wandelt die aktuell gemessene Umgebungstemperatur direkt in ein I²C-Signal um. Damit wird es auch sehr einfach möglich, mehrere Sensoren mit einem einzigen Controller auszuwerten. Auf diese Weise kann man auch eine Vielzahl von Sensoren an einen Arduino anschließen, ohne dass die Analogeingänge belegt werden. Diese bleiben so für andere Anwendungen frei.

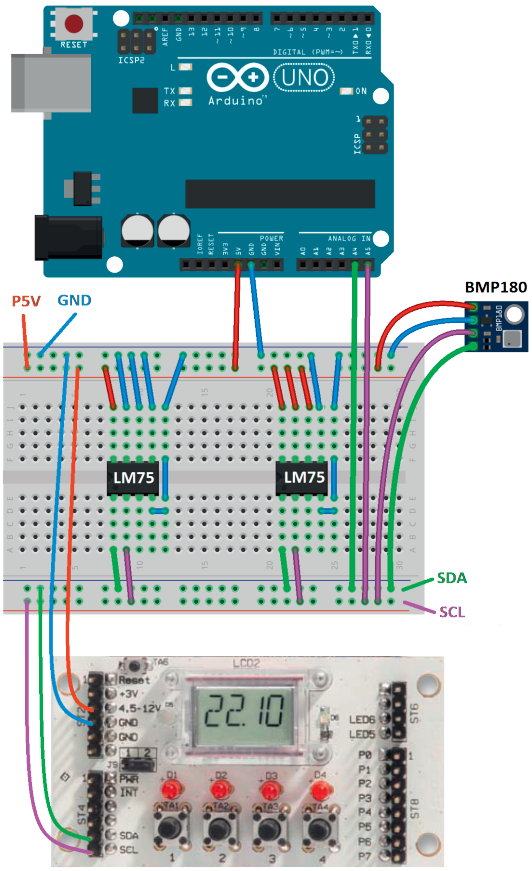

Bild 4: Sensoren und Display via I²C-Bus am Arduino

Bild 4 zeigt, wie zwei LM75-Sensoren parallel am I²C-Bus eines Arduinos betrieben werden können. Zusätzlich ist hier bereits auch noch ein I²C-fähiges LCD-Modul und ein Drucksensor eingezeichnet. Diese beiden Komponenten werden weiter unten genauer erläutert.

Die beiden Temperatursensoren müssen auf unterschiedliche Adressen konfiguriert werden. Dazu werden die Adresspins A0 bis A2 in geeigneter Weise mit Vcc oder GND verbunden. Bild 5 zeigt ein Beispiel.

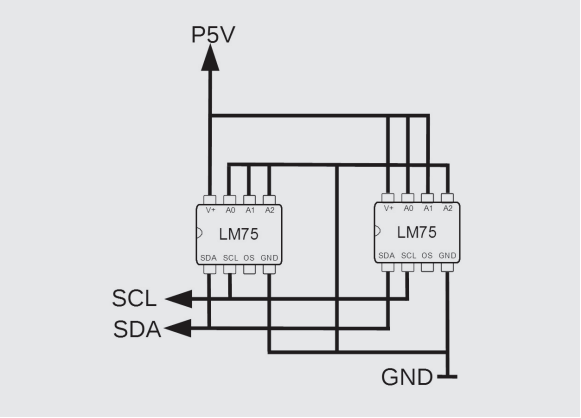

Bild 5: Einstellung von I²C-Adressen am LM75

Daraus ergeben sich die folgenden Adressen für die beiden Bausteine:

LM75 – 1:
A2 = 0, A1 = 0, A0 = 0
LM75 – 2:
A2 = 0, A1 = 1, A0 = 1

Da der fest eingestellte Teil der Adresse 1001 lautet, ergeben sich die vollständigen Adressen zu:

LM75 – 1:
1001000 = 48 hex
LM75 – 2:
1001011 = 4B hex

Sollen weitere Sensoren angeschlossen werden, müssen für diese natürlich andere Adressen gewählt werden.

I²C-fähiger Luftdrucksensor

Wie bereits aus der oben stehenden Tabelle ersichtlich ist, steht mit dem BMP180 auch für die Messung des Luftdrucks ein I²C-fähiger Baustein zur Verfügung. Dieser kann problemlos gemeinsam mit den beiden LM75-Sensoren am Arduino betrieben werden, da seine Adresse fest auf den Wert

1110111 = 77 hex

eingestellt ist.

Zusätzlich zum Luftdruck liefert dieser Sensor auch noch einen präzisen Temperaturmesswert. Zusammen mit den beiden LM75-ICs stünden damit dann bereits drei Temperatursensoren zur Verfügung, mit denen Messwerte an verschiedenen Orten erfasst werden könnten. Beispielsweise ist es so möglich, die Temperatur in einem Wohnraum, im Keller und im Außenbereich zu erfassen.

Einfacher Anschluss von Displaymodulen

Auch Displaymodule verfügen häufig über einen I²C-Anschluss. So kann beispielsweise ein alphanumerisches LC-Display über lediglich zwei aktive Pins gesteuert werden. Ein solches Display wurde bereits im letzten Beitrag zu dieser Artikelserie eingesetzt. Beim Audio-Player waren nur noch wenige freie Pins verfügbar, sodass der I²C-Bus dort seine Vorteile voll ausspielen konnte.

Natürlich könnte ein derartiges Display auch hier zum Einsatz kommen, allerdings soll an dieser Stelle eine andere Version vorgestellt werden. Das ELV I²C-Displaymodul bietet eine vierstellige 14-Segmentanzeige, die bestens zur Darstellung von Messwerten geeignet ist. Möchte man lediglich Zahlenwerte ausgeben, ist ein derartiges Display vollkommen ausreichend. Die Platine ist unter anderem auch direkt als Arduino-Shield einsetzbar.

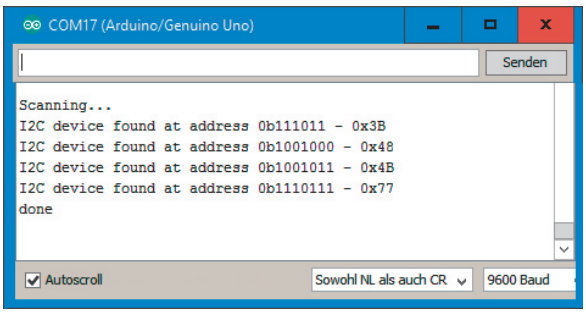

Bild 6: Der I²C-Scanner findet vier Komponenten am Bus

Der Anschluss des Displays ist ebenfalls aus Bild 4 ersichtlich.

Die I²C-Adresse des ELV Displays ist auf

0111011 = 3B hex

eingestellt und kann ebenfalls nicht verändert werden.

Test des Bus-Systems

Insbesondere wenn eine Vielzahl von Komponenten an einem Bus angeschlossen sind, ist es empfehlenswert, das Gesamtsystem vor der Inbetriebnahme zu testen.

Die folgende Tabelle fasst alle im System vorhandenen Adressen zusammen:

	binär	**hex**
LM75 – 1	0b1001000	0x48
LM75 – 2	0b1001011	0x4B
BMP180	0b1110111	0x77
I²C-Display	0b0111011	0x3B

Alle am Bus vorhandenen Sensoren können also aufgrund ihrer individuell verschiedenen Adressen problemlos angesprochen werden.

Um die Funktion des Busses zu testen, kann ein sogenannter I²C-Scanner verwendet werden.

Für den Arduino existiert dazu ein spezielles Programm, das unter [1] aus dem Internet geladen werden kann. Sind die Komponenten korrekt angeschlossen, zeigt der Scanner die jeweiligen Bus-Adressen im seriellen Monitor an (Bild 6).

Klimastation mit I²C-Komponenten

Die hier vorgestellten Komponenten ermöglichen den Aufbau einer präzisen und flexibel einsetzbaren Klimastation. Sind alle Komponenten erst einmal mit dem Arduino verbunden, fehlt nur noch ein passendes Auswertungsprogramm.

Das ELV I²C-Displaymodul verfügt neben der LCD-Anzeige auch noch über vier LEDs. Durch eine entsprechende Zuordnung und Beschriftung kann man damit jeden aktuell dargestellten Messwert kennzeichnen. Für die Anwendung als Klimastation wurde folgende Zuordnung gewählt:

LED 1: aktueller Luftdruck (P)
LED 2: Innentemperatur 1 (T1)
LED 3: Innentemperatur 2 (T2)
LED 4: Außentemperatur (T3)

Da die LEDs ebenfalls über den I²C-Bus angesteuert werden können, werden keine zusätzlichen Arduino-Pins belegt.

Um das Display anzusteuern, ist wieder eine passende Library erforderlich. Diese kann unter [2] von der ELV Produktseite des Moduls geladen werden. Das Programm dazu ist im folgenden Listing in verkürzter Form dargestellt. Der vollständige Code kann dem Download-Paket (siehe „Download-Paket zum Artikel" am Ende des Beitrags) entnommen werden.

Zunächst werden im Sketch alle erforderlichen Bibliotheken eingebunden. Dann erfolgt die Deklaration der Variablen:

P: Luftdruck in mbar
T1: z. B. Innentemperatur 1
T2: z. B. Innentemperatur 2
T3: z. B. Außentemperatur

Im Set-up werden die einzelnen Module (Wire, LCD und Serial) gestartet. In der Hauptschleife werden dann die Werte aus den jeweiligen Modulen der Reihe nach ausgelesen und sowohl an das Displaymodul als auch an die serielle Schnittstelle ausgegeben. Weitere Details dazu können bei Bedarf in [3] nachgelesen werden.

Wenn die Klimastation ohne Verbindung zu einem PC laufen soll, kann auf die Ausgabe zur seriellen Schnittstelle verzichtet werden und

die entsprechenden Programmzeilen können entfallen. Die Leuchtdioden des Displays werden dabei so angesteuert, dass für jeden Ausgabewert eine einzelne LED aufleuchtet. Zusammen mit einer entsprechenden Frontplattenbeschriftung kann man so die angezeigten Werte den jeweiligen Sensoren leicht zuordnen.

```
// I2C_climate_station.ino

#include "TwoWireLCD.h"
#include <Wire.h>
#include <Adafruit_BMP085.h>
#define LM75_1 0b1001000    // LM75 #I 7-bit adresse
#define LM75_2 0b1001011    // LM75 #II 7-bit adresse
Adafruit_BMP085 bmp;
float P, T1, T2, T3;

void setup()
{ // Initialize I2C and I2C-LCD-Moduls
  Wire.begin();  LCD.begin();  Serial.begin(115200);
  while (!Serial) {}
  Serial.println(F("Init I2C"));
  // Initialisierung des Sensors
  if (!bmp.begin())
  { Serial.println("No BMP085 sensor found!");
    while (1) {}
  }
}

void loop()
{ byte msb_1,lsb_1=0, msb_2,lsb_2=0;
  float Degrees_1=0, Degrees_2=0;
  // read LM75_1
    Wire.beginTransmission(LM75_1);
    Wire.write((byte)0x00);
    Wire.endTransmission();

    Wire.requestFrom(LM75_1, 2);
    while(Wire.available() < 2);
    msb_1 = Wire.read();
    lsb_1 = Wire.read();
…
 // read LM75_2
…

  P = bmp.readPressure();
  T1 = bmp.readTemperature();
  T2 = Degrees_1;
  T3 = Degrees_2;

  Serial.println(P);  Serial.println(T1);
  Serial.println(T2);  Serial.println(T3);

  // show pressure P in display
…
  // show temperature T1 in display
…
 // show temperature T2 in display;
…
  // show temperature T3 in display
}
```

Wenn man den Aufbau in ein geeignetes Gehäuse einbaut, hat man eine praxistaugliche Klimastation vor sich. Bild 7 zeigt einen Vorschlag dazu. Neben der Anzeige der Werte im Display kann man diese auch noch über die USB-Schnittstelle und den seriellen Monitor aufzeichnen. Man erhält so einen recht komfortablen Temperatur- und Luftdrucklogger für kontinuierliche Wetterbeobachtungen.

Bild 7: Eine Klima-Messstation mit I²C-Sensoren im Eigenbau

Ausblick

Nachdem in diesem Beitrag Grundlagen und erste Anwendungen des I²C-Busses vorgestellt wurden, folgen im nächsten Artikel weitere Praxisanwendungen. Dabei soll insbesondere der Einsatz von Portexpandern im Vordergrund stehen. Diese nützlichen Bauelemente gestatten es, die I/O-Ports eines Mikrocontrollersystems zu erweitern. So können alleine mit den beiden seriellen Signalen SCL und SDA nahezu beliebig viele weitere „Portpins" angesprochen werden. Damit ist es dann sogar möglich, ein alphanumerisches, HD44780-kompatibles Flüssigkristall-Display mit kostengünstigen Bausteinen an den I²C-Bus anzuschließen und so mit lediglich zwei Pins das gesamte Display zu steuern. ELV

Weitere Infos:

[1] http://playground.arduino.cc/Main/I2cScanner
[2] https://www.elv.de: Webcode #10094
[3] Grundlagen zur elektronischen Schaltungstechnik finden sich in der E-Book-Reihe „Elektronik!" (www.amazon.de/dp/B000XNCB02)

- FRANZIS AVR-Mikrocontroller in C Programmieren, Best.-Nr. CN-09 73 52
- Elektor-Praxiskurs AVR-XMEGA-Mikrocontroller, Best.-Nr. CN-12 07 62
- FRANZIS Arduino-Projects-Lernpaket, Best.-Nr. CN-11 51 22
- FRANZIS Physical Computing, Best.-Nr. CN-12 21 81
- FRANZIS Lernpaket Motoren & Sensoren mit Arduino, Best.-Nr. CN-12 74 74

Preisstellung April 2017 – aktuelle Preise im ELV Shop

Empfohlene Produkte	Best.-Nr.	Preis
Arduino UNO	CN-10 29 70	€ 27,95
Komplettbausatz ELV I²C-Bus-Displaymodul I²C-LCD	CN-09 92 53	€ 13,95

Alle Arduino-Produkte wie Mikrocontroller-Platinen, Shields, Fachbücher und Zubehör finden Sie unter: www.arduino.elv.de

Download-Paket zum Artikel:

Die Sketche und Beispieldateien zu diesem Artikel können unter www.elv.de: Webcode #10095 heruntergeladen werden.