Thomas
Brühlmann

mitp

Heimautomation

mit Arduino, ESP8266 und Raspberry Pi

Das eigene Heim als Smart Home für Heimwerker, Bastler und Maker

Thomas Brühlmann

Heimautomation mit Arduino, Raspberry Pi und ESP8266

Das eigene Heim als Smart Home für Heimwerker, Bastler und Maker

mitp

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

ISBN 978-3-95845-672-3
1. Auflage 2021

www.mitp.de
E-Mail: mitp-verlag@sigloch.de
Telefon: +49 7953 / 7189 - 079
Telefax: +49 7953 / 7189 - 082

Lektorat: Sabine Schulz
Sprachkorrektorat: Petra Heubach-Erdmann
Coverbild: @denisismagilov/stock.adobe.com
Satz: III-satz, Husby, www.drei-satz.de

Inhaltsverzeichnis

Einleitung

Die Automatisierung der eigenen Wohnung oder des eigenen Hauses ist ein spannendes Thema für jeden praktisch veranlagten Bastler. Dank der vielen Module und Lösungen kann jeder Anwender sein »Smart Home« individuell und nach eigenen Wünschen aufbauen.

Mit Arduino, ESP8266 und Raspberry Pi können Sie kostengünstig einzelne Lösungen realisieren: Sei es die Raumüberwachung mittels eines Sensor-Netzwerks oder die Lichtsteuerung und deren Visualisierung mit Raspberry Pi und Node-Red.

Eine Schaltzentrale mit Raspberry Pi und Standardschnittstellen wie MQTT erlauben die einfache und offene Integration von Selbstbau-Modulen wie auch fertigen, kommerziellen Modulen und Anwendungen.

Dieses Buch richtet sich an Bastler und Maker, die bereits etwas Erfahrung mit Arduino und Raspberry Pi gesammelt haben und nun praktische Anwendungen in ihrem Heim aufbauen möchten.

Aufbau des Buches

Dieses Buch ist so aufgebaut, dass Sie zuerst Grundlagen über das Arduino-Board, den ESP8266 und den Raspberry Pi lernen. Anschließend werden in verschiedenen Themenkapiteln praktische Projekte aufgebaut und in den nachfolgenden Anwendungskapiteln 8 bis 10 realisiert.

In **Kapitel 1** wird die im Buch verwendete Hardware mit Arduino und Raspberry Pi vorgestellt und in Betrieb genommen.

Die Verbindung dieser Microcontroller-Boards mit dem heimischen Netzwerk oder WLAN wird in **Kapitel 2** erklärt.

Die sehr verbreiteten Module der ESP8266-Reihe werden in **Kapitel 3** in Betrieb genommen. Mit der Installation der bekannten Firmware Tasmota haben Sie eine optimale Basis für die Projekte in den weiteren Kapiteln.

In **Kapitel 4** werden die verbreiteten Protokolle HTTP und MQTT vorgestellt.

Ein Arduino, der über MQTT ins Netzwerk integriert ist, bietet eine einfache Hardware als Sensor- und Aktormodul in der Heimautomation. In **Kapitel 5** nutzt das Arduino-Board dabei die weitverbreitete `PubSubClient`-Bibliothek.

In **Kapitel 6** wird der Minicomputer Raspberry Pi als Schaltzentrale aufgebaut. Die zentrale Anwendung dabei ist die webbasierte Entwicklungsumgebung Node-Red.

In jeder Heimautomation fühlen Sensoren die Umwelt. In **Kapitel 7** werden verschiedene Sensoren eingesetzt, um Zustände im Heim zu erfassen. Mittels verschiedener Gateway-Lösungen können Sensordaten empfangen und verarbeitet werden.

Das **Kapitel 8** beschreibt praktische MQTT-Anwendungen, die man über Node-Red schalten und verwalten kann. Eine drahtlose Fernbedienung erlaubt die Ansteuerung des Fernsehers über Tablet oder Smartphone. Weiter werden analoge Daten eingelesen, die Klingel an der Haustür vernetzt und die Überwachung des Briefkastens ins heimische Netz integriert.

In **Kapitel 9** werden Heimautomations-Lösungen vorgestellt. In einfachen Schritten kann die Anwendung Home Assistant als Basis für eine Heimautomation eingerichtet und konfiguriert werden.

Weitere praktische Projekte wie ein webbasierter Aquarium-Timer, ein Stromwächter zur Energie-Überwachung oder die Temperatur-Überwachung des Gefrierschranks werden in **Kapitel 10** beschrieben.

Mehr Informationen

Weitere Informationen zu den Heimautomations-Projekten im Buch sind auf meiner Website erhältlich:

`https://555circuitslab.com`

Im Downloadbereich finden Sie alle Beispielskripte, 3D-Vorlagen, Ergänzungen und Erweiterungen.

Bei Anmerkungen und Anregungen können Sie mich gerne per E-Mail oder über Twitter kontaktieren.

E-Mail: `maker@555circuitslab.com`

Twitter: `https://twitter.com/arduinopraxis`

Weiterführende Informationen zum Thema Arduino und Sensoren und laufend neue Projekte beschreibe ich in meinem Arduino-Blog.

`http://arduino-praxis.ch`

Auf der Verlags-Website finden Sie Details zu meinen bisherigen Buchprojekten:

Arduino Praxiseinstieg

`https://mitp.de/0054`

Sensoren im Einsatz mit Arduino

`https://mitp.de/150`

Danksagung

Ich möchte mich ganz herzlich bei meiner Familie, meiner Frau Aga und meinen Jungs Tim und Nik bedanken, dass sie mir die Zeit und den Freiraum für dieses Buch-Projekt gewährt haben.

Ein großer Dank geht auch wieder an meine Lektorin Sabine Schulz vom mitp-Verlag. Es war wieder eine sehr angenehme und erfolgreiche Zusammenarbeit.

Dieses Buch widme ich meinem Vater Bernhard.

Im Februar 2021

Thomas Brühlmann

Kapitel 1

Smarthome-Hardware

Die Hardware im IoT- und Smarthome-Umfeld ist mittlerweile sehr vielfältig und viele Hersteller, Anbieter und auch Online-Shops bieten unterschiedliche Komponenten an. Als Anwender verliert man schnell die Übersicht über die verschiedenen Systeme, Techniken und Technologien.

Grundsätzlich kann man bei einem Händler ein kommerzielles Produkt mit vielen verschiedenen Komponenten wie Sensoren, Türkontakten, Lichtschranken, Bewegungsmeldern, Alarmsirenen sowie einer zugehörigen Zentrale kaufen und installieren oder installieren lassen.

Als bastelfreudiger Anwender möchte man aber lieber eigene Module aufbauen und in sein lokales Smarthome integrieren. Meist fängt man dabei mit Sensoren zur Überwachung der Temperatur, Luftfeuchtigkeit oder des Lichts an und baut dann laufend sein System aus.

In diesem Kapitel werden die beiden Technologie-Familien Arduino und Raspberry Pi erklärt und für den Einsatz als IoT-Device und für das heimische Smarthome vorgestellt.

1.1 Arduino

Arduino-Boards sind kleine Microcontroller-Boards mit einer Anzahl von Ein- und Ausgängen. An den Eingängen können Sensoren, Schalter und Kontakte angeschlossen werden. Über die Ausgänge werden Relais, Motoren oder Schaltelemente angesteuert.

Die Erfolgsgeschichte der Arduino-Boards begann im Jahre 2005 an einer italienischen Universität. Die Ausbilder haben ein Microcontroller-Board, basierend auf einem Atmel-Microcontroller, entwickelt, damit die Studenten auf einfache Art und Weise interaktive Anwendungen realisieren konnten. Das Arduino-Board diente dabei als Zentraleinheit und wurde über eine einfache Entwicklungsumgebung in C/C++ programmiert.

Nachdem das Institut der Universität geschlossen wurde, hatten die Entwickler entschieden, dass das Arduino-Projekt unter einer Open-Source-Lizenz als Open-Source-Projekt weiterleben soll.

In der Maker- und Bastlerszene hat sich die Offenheit des Projekts schnell herumgesprochen und viele findige Entwickler haben neue Hardware-Erweiterungen (Shields) oder Bibliotheken realisiert.

Schnell gab es viele Beispiele und Anleitungen für den Einsatz dieser Arduino-Boards. Laufend werden neue Lösungen und Beispiele entwickelt.

Das Arduino-Projekt, ein Webshop, ein Forum und viele Anleitungen finden Sie unter: `https://www.arduino.cc/`.

1.1.1 Arduino als Sensor- und Aktormodul

Dank der Offenheit des Arduino-Projekts eignen sich die Arduino-Boards ideal für selbst gebaute Sensor- und Aktor-Anwendungen im IoT- und Smarthome-Bereich.

Über eine Drahtverbindung oder eine drahtlose Verbindung sind diese Sensor- und Aktormodule mit der Zentrale verbunden, senden Statussignale der Sensoren oder aktivieren einen angeschlossenen elektrischen Verbraucher (Motor, Lampe, Pumpe etc.).

Eine Drahtverbindung kann über folgende Technologien realisiert werden:

- USB-Kabelverbindung
- Bussystem über I2C-Bus
- Bussystem über RS485
- Netzwerkverbindung über Ethernet-Kabel

Eine drahtlose Verbindung kann mit einer der nachfolgenden Techniken realisiert werden:

- 433-MHz-Funktechnologie
- Wireless-Netzwerk (WLAN)
- Infrarot-Signal

Je nachdem, welche der oben genannten Technologien für einen Anwendungsfall eingesetzt werden, stehen dem Anwender entsprechende Erweiterungsplatinen (Shields) oder Zusatzmodule zur Verfügung.

Erfahrene Bastler und Anwender können eigene Arduino-Boards mit den entsprechenden Schnittstellen für eigene, spezifische Anwendungsfälle rund um ihr Smarthome realisieren.

Im nachfolgenden Abschnitt werden einige Arduino-Boards vorgestellt, die für IoT- und Smarthome-Einsätze geeignet sind.

1.1.2 Arduino-Boards

Das Arduino-Board ist eine blaue Leiterplatte mit aufgelöteten, elektronischen Bauelementen. Der Microcontroller ist die Zentrale oder das Gehirn des Boards

und ist als großer schwarzer Baustein, in der Umgangssprache als »Chip« bezeichnet, auf dem Board platziert. Der Microcontroller ist quasi der gesamte Computer und beinhaltet neben der Zentraleinheit auch den Speicher. Im Flash-Speicher werden die Programme, Sketche genannt, gespeichert. Die Zentraleinheit führt das Programm aus und verarbeitet die Ein- und Ausgangssignale.

Das Arduino-Board wird über den USB-Anschluss oder über ein externes Netzteil mit Spannung versorgt.

Arduino Uno

`https://store.arduino.cc/arduino-uno-rev3`

Der Arduino Uno ist das Standardboard der Arduino-Baureihe. In Abbildung 1.1 ist das Board abgebildet.

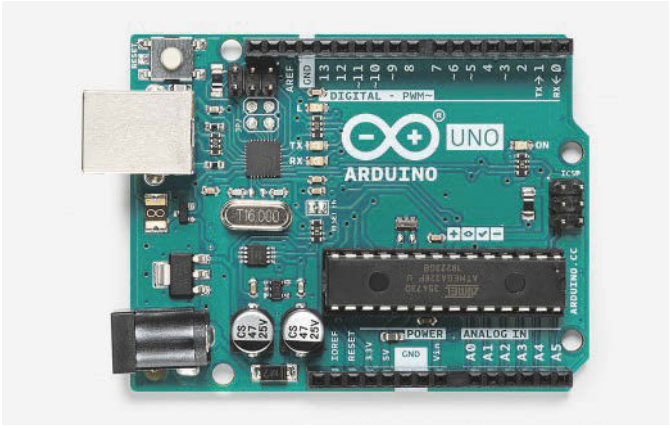

Abb. 1.1: Arduino Uno (Bild: arduino.cc)

Das Board Arduino Uno, Rev.3 dient als Basis für alle Projekte und Beispiele in diesem Buch. Der Arduino Uno eignet sich auch sonst als ideales Entwicklungsboard für den Einstieg in die Elektronik- und Microcontroller-Programmierung.

In Tabelle 1.1 sind die technischen Daten des Arduino Uno aufgelistet:

Bezeichnung	Details
Microcontroller	Atmega328
Spannungsversorgung	6–20 VDC (empfohlen 7–12 VDC)
Betriebsspannung	5 VDC
Digitale Ein/Ausgänge	14 (D0–D13, davon 6 als PWM-Ausgänge)
Analoge Eingänge	6 (A0–A5), Auflösung 10 Bit

Tabelle 1.1: Arduino Uno – technische Daten

Bezeichnung	Details
Strom pro digitalem Pin	20 mA DC
Flash Memory	32 kB (Atmega328P), wobei 0,5 kB vom Bootloader belegt werden
SRAM	2 kB (ATmega328P)
EEPROM	1 kB (Atmega328P)
Serielle Schnittstellen (UART)	1
Taktfrequenz	16 MHz
USB-Schnittstelle	ja
Resetschalter	ja
Onboard ICSP-Stecker	ja
Abmessungen Board (L x B)	70 x 53 mm

Tabelle 1.1: Arduino Uno – technische Daten (Forts.)

Auf dem Arduino Uno sind verschiedene Stecker und Anschlussmöglichkeiten platziert, die für verschiedene Funktionen ausgelegt sind. In Abbildung 1.2 sind die verschiedenen Anschlussmöglichkeiten rot dargestellt.

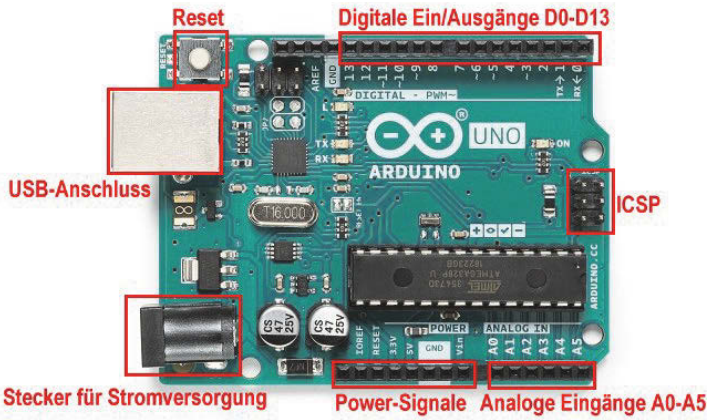

Abb. 1.2: Arduino Uno – Anschlussmöglichkeiten

Die verschiedenen Anschlussmöglichkeiten haben folgende Funktionen:

USB-Anschluss

USB-Anschluss vom Typ B für die Kommunikation des Arduino Uno mit dem angeschlossenen Rechner. Über diesen Anschluss kann ein Programm (Sketch) auf das Arduino-Board geladen werden. Gleichzeitig kann das Arduino-Board über den USB-Anschluss mit Spannung versorgt werden.

Bei der Programmierung via USB-Anschluss wird der auf dem Arduino-Board integrierte USB/Serial-Wandler verwendet.

Stecker für Stromversorgung

Dieser Anschluss für einen 5,5-mm/2,1-mm-Hohlstecker (Außen-/Innendurchmesser), in der Praxis auch Jack-Adapter genannt, dient zum Anschluss eines externen Netzteils oder einer Batterie zur Stromversorgung. Beim Anschluss einer Spannung über diesen Stecker wird die Stromversorgung aus dem USB-Anschluss deaktiviert.

Dieser Anschluss eignet sich auch, wenn Sie zusätzliche Energie für die Versorgung von Sensoren, Relais oder Motoren benötigen.

Reset-Taster

Der Reset-Taster ermöglicht das Zurücksetzen des Microcontrollers. Mit dem Betätigen des Tasters wird das Arduino-Board zurückgesetzt.

Digitale Ein/Ausgänge D0–D13

Über die obere einreihige Buchsenleiste können die digitalen Ein- und Ausgänge D0 bis D13 angesteuert werden.

ICSP

Die 2x3-polige Stiftleiste mit der Bezeichnung ICSP (In-Circuit Serial Programming) wird für die Programmierung mit einem externen Programmiergerät verwendet.

Analoge Eingänge A0–A5

Buchsenleiste für den Anschluss von 6 analogen Eingangssignalen. Die Eingangssignale müssen im Bereich von 0 bis 5 Volt liegen.

Power-Signale

Buchsenleiste mit den Spannungsversorgungen von 3,3 V und 5 V. Über Spannungsversorgungen, die auf dem Board geregelt werden, können externe Sensoren und Schaltungen auf einem Shield oder dem Steckbrett versorgt werden.

Arduino Mega 2560

`https://store.arduino.cc/arduino-mega-2560-rev3`

Der Arduino Mega ist quasi ein großer Arduino Uno mit einer größeren Anzahl von digitalen Ein- und Ausgängen, 4 seriellen Schnittstellen und einem bedeutend größeren Flash Memory. Der Arduino Mega wird mit einem Atmega2560 betrieben und eignet sich für Anwendungen, wo viele I/O-Pins erforderlich sind oder ein größerer Programmspeicher benötigt wird.

Wie Sie aus Abbildung 1.3 erkennen können, hat der Arduino Mega den gleichen Aufbau wie der Arduino Uno. Die zusätzlichen Pins sind auf erweiterten Pin-Reihen im rechten Bereich der Platine angeordnet.

Durch den gleichartigen Aufbau können viele Erweiterungsplatinen (Shields) auch auf dem Arduino Mega verwendet werden.

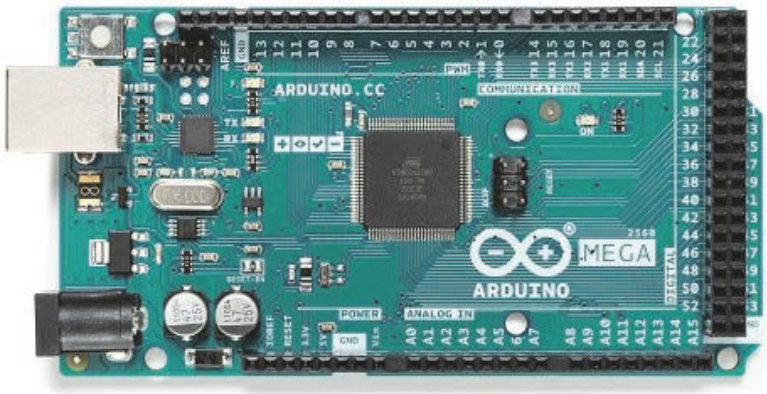

Abb. 1.3: Arduino Mega 2560 (Bild: arduino.cc)

In Tabelle 1.2 sind die technischen Daten des Arduino Uno aufgelistet.

Bezeichnung	Details
Microcontroller	Atmega2560
Spannungsversorgung	6–20 VDC (empfohlen 7–12 VDC)
Betriebsspannung	5 VDC
Digitale Ein/Ausgänge	54 (davon 15 als PWM-Ausgänge)
Analoge Eingänge	16, Auflösung 10 Bit
Strom pro digitalem Pin	20 mA DC
Flash-Memory	256 kB, wobei 8 kB vom Bootloader belegt werden
SRAM	8 KB
EEPROM	4 kB
Serielle Schnittstellen (UART)	4
Taktfrequenz	16 MHz
USB-Schnittstelle	ja
Resetschalter	ja
Onboard-ICSP-Stecker	ja
Abmessungen Board (L x B)	101 x 53 mm

Tabelle 1.2: Arduino Mega 2560 – technische Daten

Arduino Pro Mini

`https://store.arduino.cc/arduino-pro-mini`

Der Arduino Pro Mini ist ein abgespecktes Arduino-Board mit kleinen Abmessungen. Auf dem Board läuft der gleiche Microcontroller wie auf dem Arduino Uno.

Der Arduino Pro Mini ist in einer 5-V-Version mit 16 MHz und in einer 3,3-V-Version mit 8 MHz verfügbar.

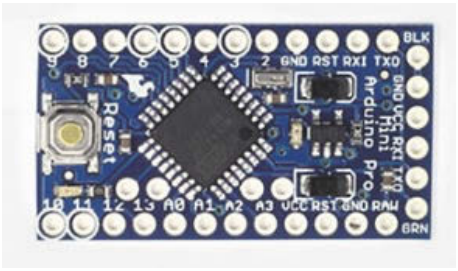

Abb. 1.4: Arduino Pro Mini (Bild: arduino.cc)

Beim Schaltungsaufbau wurden, im Vergleich zum Arduino Uno, etliche Funktionen weggelassen. Zu erwähnen sind dabei der fehlende USB-Stecker und die USB-Seriell-Schaltung.

Für den Programmupload muss ein externer USB-Seriell-Adapter an den Anschlusspins auf der schmalen Boardseite angeschlossen werden (Abbildung 1.5)

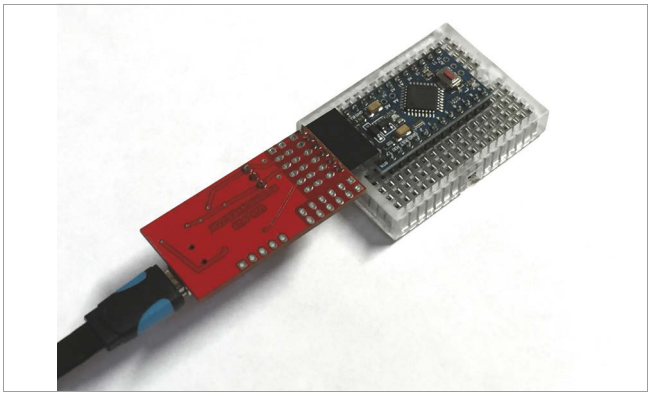

Abb. 1.5: Arduino Pro Mini mit angeschlossenem USB-Seriell-Adapter

Das Board ist im Arduino-Shop als »retired« gekennzeichnet, aber wird von vielen Händlern als Clone weitervertrieben.

Dank der schmalen Abmessungen und der kompakten Schaltung eignet sich dieses Board weiterhin ideal für drahtlose, batteriebetriebene Anwendungen.

Die Leiterplatte des Arduino Pro Mini passt in ein kleines Gehäuse und über die Lötpads an den Rändern des Boards können die externen Sensoren oder Schaltungselemente angeschlossen werden.

Das Thema Batteriebetrieb wird in einem späteren Kapitel noch im Detail beschrieben.

1.1.3 Entwicklungsumgebung IDE

Neben der Hardware des Arduino gehört zum Projekt auch eine kostenlose Entwicklungsumgebung.

Diese Entwicklungsumgebung, auch IDE (Integrated Development Environment) genannt, ist die Programmieroberfläche und ermöglicht dem Anwender das Erstellen, Testen und Hochladen von Arduino-Programmen. Die Arduino-Programme werden auch als Sketche bezeichnet.

Die Arduino-Entwicklungsumgebung ist ein Java-Programm. Die Software ist für die Betriebssysteme Windows, Mac OS X und Linux verfügbar.

Die Software wird laufend weiterentwickelt und ist aktuell in der Version 1.8.13 (Stand Herbst 2020) verfügbar und auf der Arduino-Website verfügbar.

`https://www.arduino.cc/en/Main/Software`

Installation

Im Downloadbereich der obigen Internetadresse steht die Software für das jeweilige Betriebssystem bereit.

Windows

Windows-Benutzer nutzen den praktischen Installer, der neben der Entwicklungsumgebung gleichzeitig den notwendigen Treiber installiert.

Mac OS X

Für Mac-Anwender wird die Entwicklungsumgebung als ZIP-Datei bereitgestellt. Nach dem Download und Entpacken kann die Anwendung in einen beliebigen Ordner kopiert und dann ausgeführt werden.

Linux

Linux-Anwender laden sich das passende Paket auf den Rechner und folgen den Schritten der Anleitung.

`https://www.arduino.cc/en/Guide/Linux`

Inbetriebnahme

Nach der erfolgreichen Installation der Entwicklungsumgebung kann das Programm gestartet werden. Die Entwicklungsumgebung startet mit einem leeren Codefenster (Abbildung 1.6).

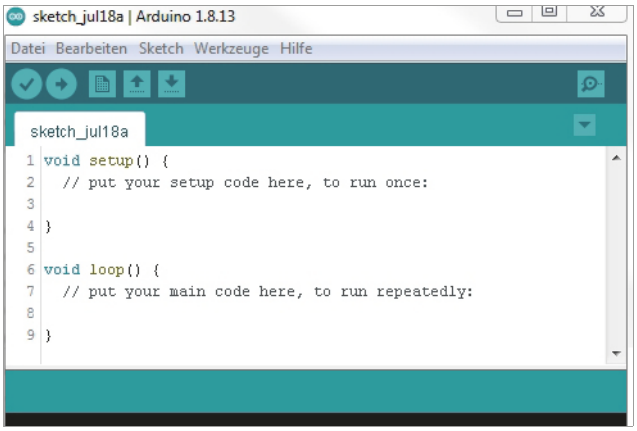

Abb. 1.6: Arduino-Entwicklungsumgebung

Nun kann ein Arduino-Board über ein USB-Kabel mit dem Rechner verbunden werden.

Für die korrekte Kommunikation zwischen dem Rechner und dem Arduino müssen in der Entwicklungsumgebung das verwendete Arduino-Board und der COM-Port ausgewählt werden.

Die Einstellungen dazu finden Sie unter WERKZEUGE|BOARD beziehungsweise WERKZEUGE|PORT (Abbildung 1.7).

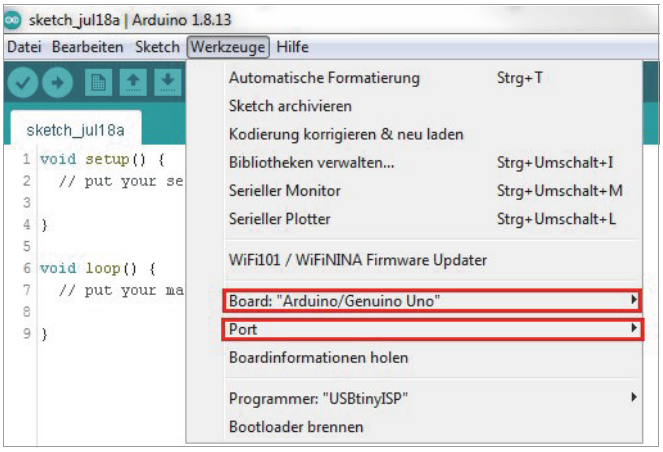

Abb. 1.7: Arduino-Entwicklungsumgebung – Auswahl Board und Port

Mit der richtigen Auswahl der beiden Optionen steht das Arduino-Board für einen ersten Test zur Verfügung.

Verbindungsaufnahme und Sketch »Blink«

Bei der Software-Entwicklung führt man meist als erstes Programm ein »Hello World« aus. Im Arduino-Umfeld nennt sich diese Programm *Blink* und wird mit der Entwicklungsumgebung als Beispiel mitgeliefert.

Das Beispielprogramm *Blink* ist unter DATEI|BEISPIELE|01.BASICS aufrufbar.

Der Blink-Sketch ist, wie der Name aussagt, ein Blink-Programm, das den Ausgang D13 des Arduino-Boards im Sekundentakt ein- und ausschaltet.

Für den ersten Test wird Blink ausgewählt und auf das Arduino-Board geladen. Dazu wird das Icon mit dem Pfeil (Hochladen) angeklickt (Abbildung 1.8).

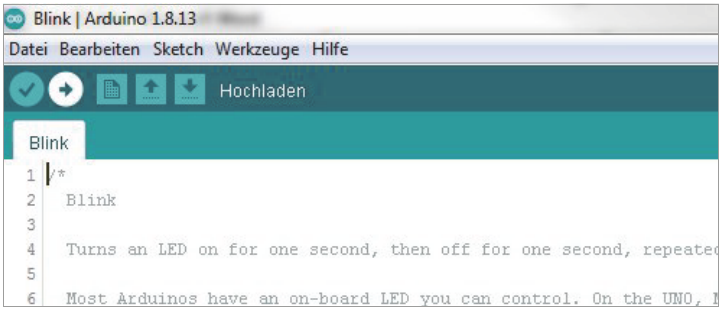

Abb. 1.8: Arduino-Entwicklungsumgebung – Blink hochladen

Nach dem Anklicken der Hochladen-Funktion wird der Blink-Sketch kompiliert und auf das Arduino-Board hochgeladen. Der gesamte Vorgang kann ein paar Sekunden dauern.

Nach dem erfolgreichen Hochladen des Sketches meldet die Entwicklungsumgebung dies mittels Erfolgsmeldung in der Fußzeile (Abbildung 1.9).

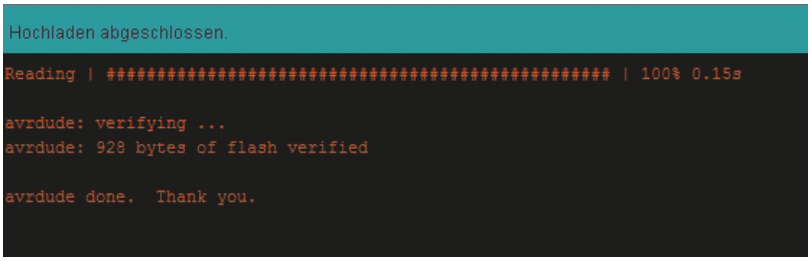

Abb. 1.9: Arduino-Entwicklungsumgebung – Blink erfolgreich hochgeladen.

Gleichzeitig wird der Arduino Uno neu gestartet und auf dem Board beginnt die Leuchtdiode mit der Bezeichnung L zu blinken. Gratulation!!

Falls beim Kompilieren oder Hochladen ein Fehler auftaucht, stoppt die Kompilierung und meldet einen Fehler. Wurde beispielsweise ein falsches Board ausgewählt oder die Kommunikation über den COM-Port bricht ab, so wird eine entsprechende Fehlermeldung ausgegeben.

1.1.4 Programmierung, Programmstruktur

Das einfache Blink-Programm zeigt den grundsätzlichen Aufbau eines Arduino-Sketches. Die minimalste Struktur besitzt eine Setup-Funktion `setup()` und ein Hauptprogramm `loop()` (smarthome_kap1_struktur.ino).

```
// Arduino-Sketch – Struktur

void setup()  // Programmstart
{
  // Anweisungen
}

void loop()   // Hauptprogramm
{
  // Anweisungen
}
```

Die Setup-Funktion `setup()` ist zwingend notwendig und wird bei jedem Programmstart einmalig aufgerufen. In dieser Funktion werden Grundeinstellungen und die Deklaration von Variablen und Einstellungen vorgenommen.

Das Hauptprogramm `loop()` wird nach dem Ausführen der Setup-Funktion nun endlos durchlaufen. Das Hauptprogramm wird ausgeführt, bis eine Spannungsunterbrechung oder ein Reset die Ausführung stoppt.

Die Programmstruktur aus diesem Beispiel finden Sie in jedem Arduino-Sketch.

1.1.5 Praxisbeispiel: Temperaturmesser mit NTC und LED

Die Blink-Anwendung aus dem vorherigen Abschnitt dient für den Einstieg in die Arduino-Programmierung und zum Test, ob ein Arduino-Board noch funktioniert.

Der LED-Blinker selbst ist nicht wirklich eine Praxisanwendung.

In diesem ersten Praxisbeispiel soll ein einfacher Temperatur-Sensor die Umgebungstemperatur messen und über eine Anzeigeeinheit mit LEDs ausgeben.

Der Messbereich dieser Anwendung von 0 bis 50 Grad Celsius wird optisch mit 5 Leuchtdioden dargestellt, jeweils in 10-Grad-Einheiten.

Die Aufteilung sieht gemäß Tabelle 1.3 aus.

LED1	LED2	LED3	LED4	LED5
0–10	10–20	20–30	30–40	40–50
Blau	Grün	Gelb	Orange	Rot

Tabelle 1.3: Temperatur-Anzeige mit LED

Idealerweise nimmt man, je nach Temperaturbereich, unterschiedliche Leuchtdioden, um eine optimale Darstellung zu erreichen. Bei den tieferen Temperaturen nimmt man die kalten Farben (Blau, Grün) und bei den höheren Temperaturen entsprechend die warmen Farben (Gelb, Orange, Rot).

Stückliste (Temperaturmesser mit NTC und LED)

- 1 Arduino Uno
- 1 NTC 10 kOhm
- 5 Widerstände 1 kOhm
- 1 Widerstand 10 kOhm
- 5 LED (verschiedenfarbig)
- 1 Steckbrett
- Jumper-Wires

Ein NTC ist ein einfacher Temperatursensor mit negativem Temperatur-Koeffizienten. Bei 25 Grad besitzt der eingesetzte NTC einen Widerstandswert von 10 kOhm. Der Widerstandswert vermindert sich bei höherer Temperatur. Die genauen Temperatur-Werte können aus dem Datenblatt des Lieferanten entnommen werden.

In der Praxis heißt das, dass der NTC bei Kälte einen hohen Widerstandswert hat und bei Hitze ist der Wert niedrig.

NTC-Sensoren gibt es in vielen Größen und Bauformen. In Abbildung 1.10 ist ein einfacher NTC abgebildet, der bei vielen Elektronik-Händlern verfügbar ist. Dieser Typ hat ein dichtes Gehäuse und eignet sich auch für Außenmessungen und Anwendungen im feuchten Umfeld. Der Messbereich dieses Sensor-Typs liegt bei –25 bis +125 Grad Celsius.

Abb. 1.10: NTC-Temperatursensor (Bild: Aliexpress)

Abbildung 1.11 zeigt den Steckbrett-Aufbau der Schaltung mit den Leuchtdioden als Temperaturanzeige. Für die Messung der aktuellen Temperatur wird der Sensor mit einem Widerstand in einer sogenannten Spannungsteiler-Schaltung betrieben. Das Arduino-Board kann nämlich nicht direkt einen Widerstandswert in Ohm messen. Beim verwendeten Messprinzip gibt der Spannungsteiler aus NTC und Widerstand eine proportionale Spannung ab. Diese Spannung, die dem gemessenen Wert entspricht, kann vom Arduino-Board über einen analogen Eingang eingelesen werden.

Der Temperaturbereich von 0 bis 50 Grad und die Anzeige mit Leuchtdioden können bei Bedarf erweitert werden.

Die Ansteuerung der Leuchtdioden erfolgt über die digitalen Ausgänge D2 bis D6.

Das analoge Messsignal wird am analogen Eingang A0 eingelesen.

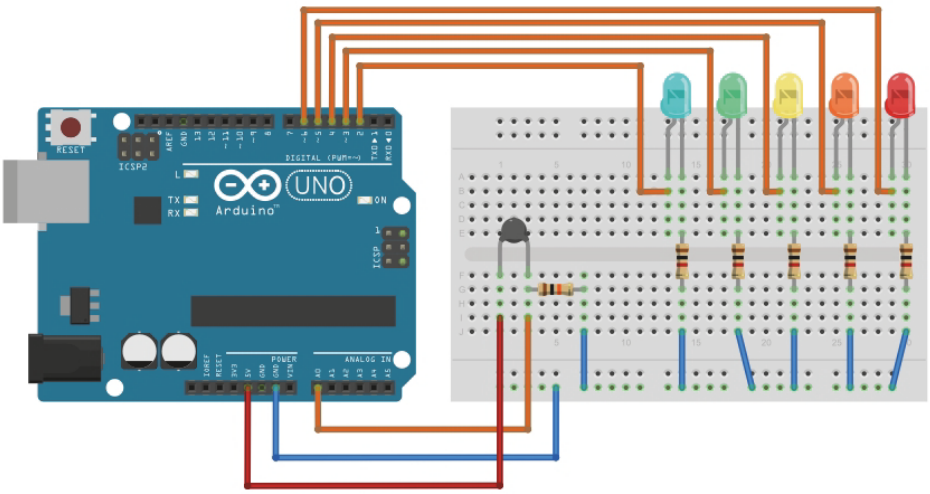

Abb. 1.11: Steckbrett-Aufbau: Temperaturmesser mit LED

Die Berechnung des aktuellen Temperaturwerts erfolgt im Arduino-Code mittels der »Steinhart-Hart-Gleichung«. Für die Berechnung ist die Mathematik-Bibliothek `math.h` erforderlich.

Diese Bibliothek wird zuerst im Programmcode eingebunden (`smarthome_kap1_ntc.ino`).

```
#include <math.h>
```

Die Berechnung der Temperatur benötigt den Widerstandswert des NTC bei 25 Grad Celsius in Ohm.

```
// Widerstandswert bei 25 Grad C
int Wid25Grad= 10000;
```

Nun werden die Variablen für den Messwert, den analogen Eingang und den Temperaturwert deklariert.

```
// analoger Messwert
int valTemp;
// analoger Eingang
int TempPin = 0;
// Wert Temp in Celsius
float tempC;
```

Weiter werden die Pins für die digitalen Ausgänge für die 5 Leuchtdioden definiert.

```
// digitale LED-Ausgänge
int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
int LED5 = 6;
```

Im Setup wird die serielle Schnittstelle vorbereitet und die digitalen Pins für die Leuchtdioden als Ausgänge gesetzt:

```
void setup()
{
  // Start serielle Ausgabe
```

```
  Serial.begin(9600);
  // Ausgänge setzen
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  pinMode(LED5, OUTPUT);
}
```

Im Hauptprogramm `loop()` wird nun bei jedem Programmdurchlauf der analoge Eingang A0 eingelesen und in der Variablen `valTemp` gespeichert. Der Wert wird anschließend an die Funktion `ThermistorC()` übergeben. Die Umrechnungsfunktion rechnet den gemessenen Wert in eine absolute Temperatur um und gibt den Wert zurück. Der Temperaturwert wird in der Variablen `tempC` gespeichert und über die serielle Schnittstelle ausgegeben:

```
void loop()
{
  valTemp = analogRead(TempPin);
  tempC = ThermistorC(valTemp);
  Serial.print("Temperatur: ");
  Serial.print(tempC);
  Serial.println(" C");
```

Nach der Temperaturmessung werden die einzelnen Leuchtdioden der Temperaturanzeige angesteuert. Sobald der Temperaturwert den Endwert des Anzeigebereichs überschreitet, wird die jeweilige Leuchtdiode eingeschaltet.

```
  // Anzeige Temperaturwert mit LEDs
  if (tempC > 10)
  {
    digitalWrite(LED1, HIGH);
  }
  if (tempC > 20)
  {
    digitalWrite(LED2, HIGH);
  }
  if (tempC > 30)
  {
    digitalWrite(LED3, HIGH);
  }
```

```
  if (tempC > 40)
  {
    digitalWrite(LED4, HIGH);
  }
  if (tempC > 50)
  {
    digitalWrite(LED5, HIGH);
  }
```

Nach einer Verzögerung von einer Sekunde beginnt der nächste Messvorgang mit dem nächsten Durchlauf.

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

Die oben erwähnte Umrechnungsfunktion `ThermistorC()` rechnet den Messwert des analogen Einganges A0 in den absoluten Temperaturwert in Grad Celsius um. Der Übergabeparameter `RawADC` ist der eingelesene Messwert.

```
double ThermistorC(int RawADC)
{
 double Temp;
 Temp = log(((10240000/RawADC) - Wid25Grad));
 Temp = 1 / (0.001129148 + (0.000234125 * Temp) + (0.0000000876741 *
Temp * Temp * Temp));
 // Umrechnung Kelvin / Grad Celsius
 Temp = Temp - 273.15;
 return Temp;
}
```

1.1.6 Bibliotheken

Arduino-Bibliotheken sind Software-Pakete, die die Funktion eines Arduino-Boards erweitern. In der Arduino-Entwicklungsumgebung werden einige Standardbibliotheken mitgeliefert. Die Standard-Bibliotheken wie `Servo`, `Ethernet`, SD oder Wire sind unter DATEI|BEISPIELE aufrufbar. Für jede Bibliothek stehen Beispielsketche zur Verfügung (Abbildung 1.12).