Arduino – eine Open Source Plattform

Der Begriff „Arduino“ bezeichnet eine Open-Source-Plattform, die aus einer einfachen Hardware (dem Arduino-Board) und einer leicht zu bedienenden Software besteht. Im Grundpraktikum Physik der TU Ilmenau verwenden wir das Arduino-Board „Uno“. Der Inhalt dieser Anleitung ist grundlegender Natur und somit auch für die meisten anderen Arduino-Boards gültig. Genaueres über die verschiedenen Arduino-Boards erfährt man auf der offiziellen Arduino-Website (https://www.arduino.cc).

Ein Arduino-Board ist ein Mikrocontroller, also ein System, das einem Computer ähnelt: Es hat einen Speicher, einen (Mikro-)Prozessor und Schnittstellen mit der Außenwelt. Der Arduino Uno ist nicht der leistungsstärkste Mikrocontroller, aber sowohl sein Aufbau als auch das Konzept dahinter sind quelloffen (*open-source*). Sucht man im Internet nach dem Stichwort „Arduino-Projekte“ zeigt sich schnell die große Vielfalt an Dingen, die man mit einem Arduino machen kann: Roboter zum Leben erwecken, das eigene Zuhause automatisieren, u.v.m.

Der Arduino Uno

Externe Stromversorgung 7 – 12 V: Diese wird benötigt, wenn der Arduino Uno nicht per USB Kabel an einen PC angeschlossen ist.

USB-Anschluss: Mit Hilfe eines USB-Kabels kann der Arduino Uno mit einem PC verbunden werden, um Daten zu übertragen und auch um das Board mit Strom zu versorgen.

LED: Das Leuchten der LED zeigt an, dass der Arduino mit Strom versorgt ist.

Anzeige serielle Kommunikation: Diese LED leuchten, wenn Daten zwischen dem Arduino und dem PC ausgetauscht werden.

Digitale Anschlüsse 2 bis 13: Die beiden einzig möglichen Spannungswerte betragen 0V und 5V. Der Wert eines digitalen Anschlusses kann also in einem Bit kodiert werden: 0 oder 1 (wahr oder falsch). Für jeden einzelnen Anschluss kann im Programm festgelegt werden, ob er als Ein- oder Ausgang

dienen soll. Wird ein Anschluss als Eingang deklariert, so misst das Board die Spannung, die an dem Anschluss anliegt. Ist diese geringer als ca. 3V, so liest der Arduino den Zustand als „0“ (LOW). Höhere Spannungen werden als „1“ (HIGH) gelesen. Wird im Gegensatz dazu der Anschluss als Ausgang deklariert, so dient er als steuerbare Spannungsquelle, die nur zwei Werte liefert: LOW (~GND, 0V) und HIGH (~5V).

Digitale Anschlüsse 0 und 1: Können wie die anderen digitalen Anschlüsse genutzt werden, wenn keine Kommunikation zwischen dem Arduino und dem PC stattfindet. Ansonsten werden sie zur seriellen Kommunikation mit dem PC eingesetzt und stehen nicht dem Projekt zur Verfügung.

Pseudo-analoge Ausgänge: Diese sind mit dem Symbol ~ gekennzeichnet und können durch Pulsweitenmodulation (PWM, Näheres s. Anhang) auch Spannungen zwischen 0 und 5V ausgeben.

Analoge Eingänge: Die Anschlüsse A0 bis A5 können Spannungen zwischen 0 und 5V messen (aber nicht erzeugen). Dabei wird stets mit Bezug auf GND gemessen. Der Arduino wandelt dann die gemessene Spannung in eine Zahl um mit Hilfe eines Analog-Digital-Wandlers (analog/digital converter ADC). Der ADC des Arduino hat eine 10bit-Auflösung, d.h. er akzeptiert als Eingang eine Spannung zwischen 0V und einer Referenzspannung (standardmäßig 5V) und gibt an den Mikrocontroller eine Zahl zwischen 0 und 1023 (10bit=2^{10}=1024) aus:

- Für $U \leq 0V$ gibt der ADC den Wert 0 aus
- Für $U \geq U_{ref}$ gibt der ADC den Wert 1023 aus
- Für $0 < U < U_{ref}$ gibt der ADV eine ganze Zahl zwischen 0 und 1023 aus einer linearen Skala folgend

Konstantspannungs-Ausgänge : Diese Ausgänge geben jeweils eine konstante Spannung aus:

Port 5V erzeugt eine Spannung von 5V;
Port 3V3 erzeugt eine Spannung von 3,3V;
die GND Ports sind mit Masse verbunden, d.h. 0V.

Die zentralen Elemente des Arduino sind seine Ein- und Ausgänge, da er mit deren Hilfe mit der Außenwelt kommuniziert. Die Ausgänge dienen als Spannungsquellen, die vom Arduino kontrolliert werden. So ist es dem Arduino möglich mit Hilfe der Ausgänge Aktionen auszulösen (z.B. Ein- und Ausschalten einer LED, eines Motors,…). Die Eingänge hingegen dienen als Spannungsmesser, deren Messwert vom Arduino ausgelesen wird. Somit kann der Arduino über die Eingänge den Zustand eines Systems erfassen, mit dem er verbunden ist (z.B. Auslesen eines Sensors).

Die Arduino IDE Software

Der Arduino weiß durch ein Programm in seinem Speicher, was er konkret tun soll. Um das gewünschte Programm zu erzeugen, benötigt man eine geeignete Software: die integrierte Entwicklungsumgebung (**I**ntegrated **D**evelopment **E**nvironment). Im Grundpraktikum Physik der TU Ilmenau verwenden wir die Arduino IDE, welche von der offiziellen Arduino-Website (https://www.arduino.cc) kostenlos heruntergeladen werden kann. Nachdem das Programm in der Arduino-IDE geschrieben wurde, wird es kompiliert und über die USB-Verbindung auf den Arduino geladen. Eine detaillierte Beschreibung der Arduino IDE findet man auf der Arduino-Webseite. Im Folgenden sollen die wesentlichen Elemente vorgestellt und danach Beispielprogramme diskutiert werden.

Die am häufigsten genutzten Features findet man als Symbole in der Kopfzeile (1.-4.) und in der Seitenleiste (5.-9.):

1. **Überprüfen** – Kompilieren des Programms.
2. **Hochladen** – Hochladen des Programms auf den Arduino.
3. **Board wählen** – Erkannte Arduino Boards werden automatisch angezeigt, zusammen mit ihrer jeweiligen Portnummer.
4. **Serieller Monitor** – Öffnet als neuen Tab in der Konsole einen seriellen Monitor, in dem man Nachrichten empfängt, die der Arduino an die serielle Schnittstelle sendet. Sinnvoll bei der Fehlersuche im Programm.
5. **Sketchbook** – Hier sind alle Sketche aufgelistet, die lokal auf dem PC gespeichert sind. Als Sketch bezeichnet man die Programme für den Arduino. Ihre Dateinamen enden auf .ino.
6. **Board-Verwaltung** – Auswahl an Packages von Arduino und anderer Anbieter, die installiert werden können. Der Arduino Uno beispielsweise erfordert die Installation des “Arduino AVR Boards“ Pakets.
7. **Bibliotheks-Verwalter** – Auswahl an zahlreichen Arduino Bibliotheken .
8. **Debugger** – Fehlersuche und -beseitigung im Programm in Echtzeit.
9. **Suche** – Stichwortsuche im Quelltext.

Im Hauptfenster wird der Quelltext für das Programm geschrieben, das auf dem Arduino gespeichert und von ihm ausgeführt werden soll. Direkt darunter befindet sich das Nachrichtenfenster des Compilers. Durch Klicken auf „Überprüfen“ wird das Programm kompiliert, d.h. in für den Mikrocontroller verständlichen Code umgewandelt. Treten Fehler auf, so werden diese als orangefarbene Nachrichten angezeigt. Nach Korrektur der Fehler und erfolgreicher Kompilierung wird das Programm durch Klicken auf „Hochladen“ mit Hilfe eines USB-Kabels auf den Arduino übertragen (die Anzeige-LED für serielle Kommunikation auf dem Board blinkt dabei). Ein eventuell bisher befindliches Programm auf dem Arduino wird überschrieben. Sobald das (neue) Programm vollständig auf den Arduino geladen ist, wird es ausgeführt. Ist eine externe Stromversorgung angeschlossen, so kann das USB-Kabel entfernt werden. Außerdem kann durch Drücken des Reset-Knopfes auf dem Board erzwungen werden, dass das Programm erneut von vorn startet.

Grundlegendes zur Programmierung

Beim Öffnen eines neuen Sketches enthält dieser bereits die beiden Kernelemente des Programms: die setup()-Prozedur und die loop()-Prozedur. Die setup() wird jedes Mal aufgerufen, wenn der Sketch startet. Sie soll benutzt werden, um Variablen, Pinmodi, Bibliotheken, usw. zu initialisieren. Die Funktion wird nur ein einziges Mal aufgerufen: Jedes Mal, wenn das Board gestartet oder resettet wird.

Die loop() hingegen ist eine Endlosschleife, die nach jedem Durchlauf erneut aufgerufen wird. Dadurch kann das Programm Variablen verändern, Daten lesen oder darauf reagieren und wird verwendet, um das Arduino-Board aktiv zu steuern.
In dem Menüpunkt “Datei – Beispiele“ findet man viele, gut kommentierte Beispiel-Sketche. Es ist empfehlenswert sich einige davon anzusehen, um eine Idee vom Programmaufbau zu bekommen oder sich einfach inspirieren zu lassen. Einige wichtige Elemente der Arduino-Programmiersprache (C++) werden im Folgenden vorgestellt. Eine ausführliche Übersicht mit detaillierten Erklärungen findet sich auf der Arduino-Webseite: https://www.arduino.cc/reference/de/ .

1. Variablen

Eine Variable ist ein abstrakter Behälter für einen Wert. Ihr wird im Quelltext ein Name zugewiesen und auch die Art der Daten, die in ihr abgelegt werden. Bei der Variablendeklaration kann bei Bedarf ein anfänglicher Wert zugewiesen werden.

```
int Variable1 ;           // deklariert die Variable „Variable1“ als Typ „int“,
                          // d.h. ganze Zahl zwischen -32768 und 32767
int Variable2 = 0 ;       // deklariert die Variable „Variable2“ als Typ „int“
                          // und setzt ihren Anfangswert auf 0
unsigned int Variable3 ;  // positive ganze Zahl zwischen 0 und 65535
float Variable4 = 0.0 ;   // reelle Zahl
boolean Var5 = true ;     // Boole’sche Variable, wahr/true oder falsch/false
```

Die Variablen können nur in der Prozedur verwendet werden, in der sie deklariert sind. Damit eine Variable global im gesamten Programm verwendet werden kann, muss sie am Anfang des Programms vor der setup()-Prozedur deklariert werden.

2. Eingänge und Ausgänge

In der setup()-Prozedur muss festgelegt werden, ob ein digitaler Port als Eingang oder Ausgang verwendet werden soll.

```
pinMode(3, OUTPUT);       // deklariert den digitalen Port 3 als Ausgang
pinMode(4, INPUT);        // deklariert den digitalen Port 4 als Eingang
```

Man kann auch direkt den Zustand des Ausgangs festlegen:

```
digitalWrite(3, HIGH);    // setzt Port 3 auf HIGH (d.h. 5V)
digitalWrite(3, LOW);     // setzt Port 3 auf LOW (d.h. 0V)
```

Für die Pseudo-analogen Ausgänge wird die auszugebende Spannung per Pulsweitenmodulation erzeugt (Illustration: s. Anhang) und wie folgt deklariert:

```
analogWrite(6,0) ;        // setzt den Wert von Port 6 auf 0 (0V)
analogWrite(6,255) ;      // setzt den Wert von Port 6 auf 255 (5V)
analogWrite(6,100) ;      // setzt den Wert von Port 6 auf 100 (Mittelwert von 2V)
```

Um einen digitalen Eingang einzulesen:

```
Var = digitalRead(4) ;    // liest den HIGH oder LOW Wert des digitalen Ports 4 und
                          // weist ihn der Variable „Var“ zu
```

Um einen analogen Eingang einzulesen:

```
Umess = analogRead(A5) ;  // misst die Spannung von Port A5, eine ganze Zahl
                          // zwischen 0 und 1023; weist sie der Variable „Umess“ zu
```

3. Datentransfer per USB

Die USB-Schnittstelle ist eine serielle Schnittstelle, die zur wechselseitigen Datenübertragung zwischen Arduino und PC dient. Zu Beginn des Programmes muss in der setup()-Prozedur die Datenübertragung aktiviert werden:

```
Serial.begin(9600) ;        // initialisiert die serielle Schnittstelle
                            // Übertragung mit einer Baudrate von 9600
```

Dann kann der Arduino Informationen an den PC senden:

```
Serial.print("Messwert:");// Arduino sendet den Text „Messwert:"
Serial.print(myVar) ;       // Arduino sendet den Wert der Variablen myVar
Serial.println("Hallo");   // Arduino sendet den Text "Hallo"
                            // und fügt Zeilenumbruch an
```

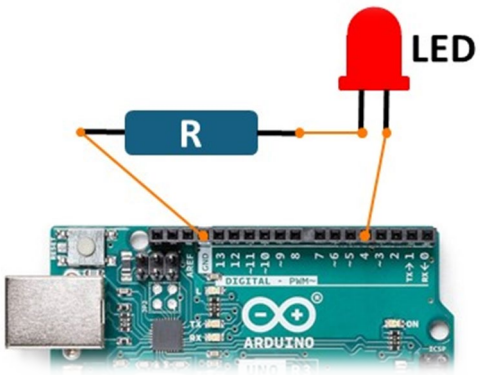
Verarbeitung der vom Arduino gesendeten Daten

Die vom Arduino Uno an den PC gesendeten Daten kann man sich in der Arduino-IDE mit Hilfe des seriellen Monitors anzeigen lassen. Es ist jedoch nicht möglich die Daten in einer separaten Datei abzulegen. Daher empfiehlt sich der zusätzliche Einsatz eines Terminal-Programmes. Im Grundpraktikum Physik der TU Ilmenau findet die Software HTerm Anwendung. Darin muss zunächst der Port ausgewählt werden mit dem der Arduino verbunden ist und die korrekte Baudrate eingestellt werden. Durch Klicken auf Connect wird dann die Verbindung hergestellt und die vom Arduino an den PC gesendeten Daten im Hauptfenster angezeigt. Diese können dann durch Auswahl von „Save output" in eine Textdatei geschrieben werden.
Insbesondere bei ausgelesenen Sensordaten möchte man die Werte oftmals in Tabellenformat haben. Dazu empfiehlt es sich die Textdatei in MS Excel zu importieren. Eine ausführliche Beschreibung, wie dies funktioniert, findet man unter https://support.microsoft.com/de-de/office/importieren-oder-exportieren-von-textdateien-txt-oder-csv-5250ac4c-663c-47ce-937b-339e391393ba .
Liegen die Daten dann in Tabellenform vor, kann man sie leicht in eine spezifische Auswerte-Software kopieren und dort analysieren.

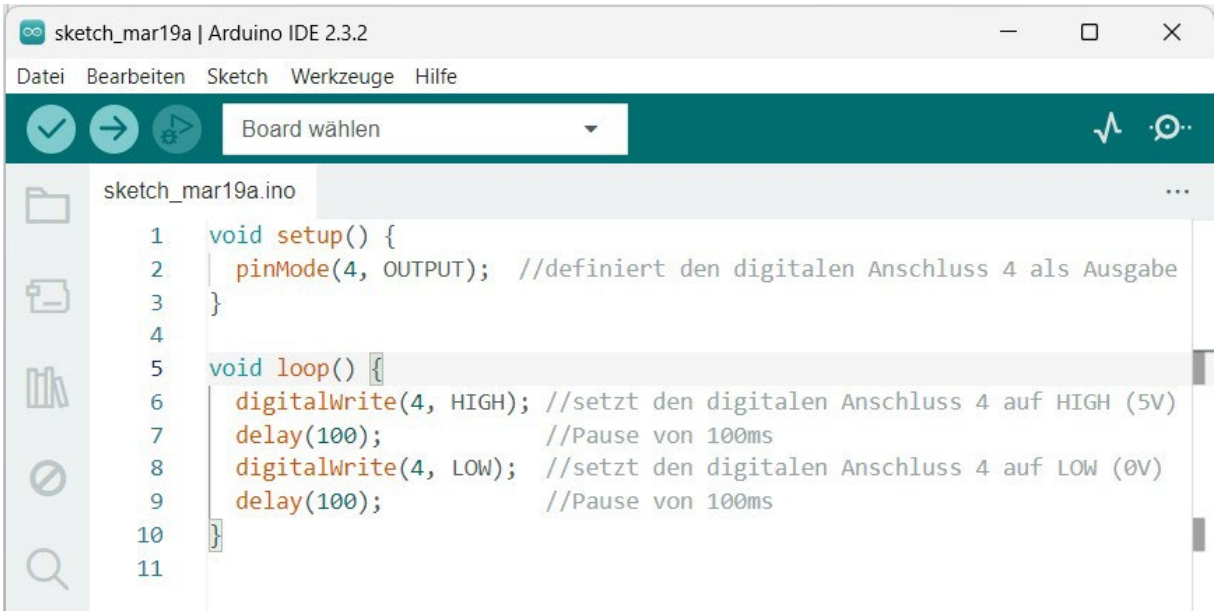
Anhang

1. Beispielprogramm: Ansteuerung einer LED

In dem Beispiel soll eine LED mit Hilfe eines Arduino an und aus geschaltet werden. Dazu wird ein digitaler Ausgang genutzt, um als Spannungsquelle entweder 0V oder 5V zur Verfügung zu stellen.
Nebenstehende Abbildung veranschaulicht eine mögliche Schaltung, wobei der digitale Anschluss 4 genutzt wird. Der Ohm`sche Widerstand wird benötigt, um den Strom zu begrenzen und somit sowohl die LED als auch den Arduino vor Überlastung zu schützen.
Das zugehörige Programm zur Steuerung des Arduino



kann wie folgt aussehen:



Da Befehle, die in der loop()-Prozedur stehen, kontinuierlich ausgeführt werden, wird die LED ständig an und aus geschaltet werden, so lange das Programm läuft. Dieses Umschalten erfolgt so schnell, dass man eventuell nur ein Flackern sehen kann. Daher wurde jeweils eine Pause von 100ms eingefügt. So lange soll das System in dem jeweiligen Zustand verharren.

2. Illustration Pulsweitenmodulation