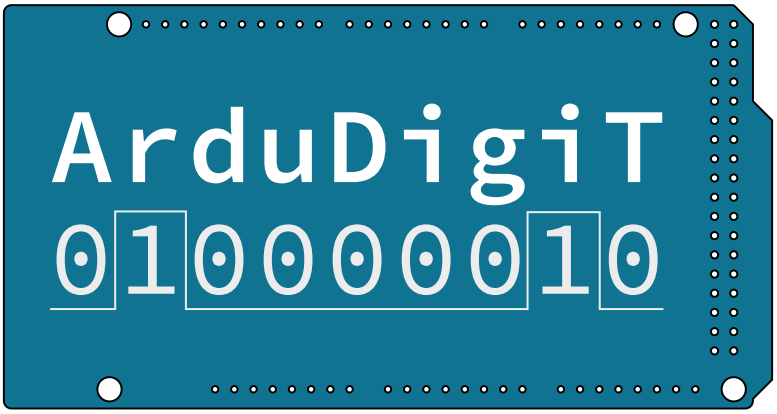
Handreichung für Lehrkräfte



Letzte Bearbeitung: 6. September 2023

Erstellt durch Max Kobrow (max.kobrow@gmail.com)
Materialien lizenziert unter CC BY-SA 4.0

Inhaltsverzeichnis

1 Allgemeines

1.1 Zielgruppe

Der Kurs richtet sich vornehmlich an Schüler*innen des Leistungskurses Informatik an Gymnasien beziehungsweise des Leistungskurses Informatiksysteme an beruflichen Gymnasien in Sachsen. Mögliche Lernbereiche am allgemeinbildenden Gymnasium sind die Lernbereiche 7 *Rechnernetze* und 1 *Technische Informatik*. Am beruflichen Gymnasien wäre ein Einsatz vor allem im Lernbereich 3 *Realisierung von IuK-Systemen in Netzwerken* denkbar.

Der Einsatz in anderen Bundesländern und in Grundkursen ist auch möglich, wird hier aber nicht weiter behandelt.

1.2 Voraussetzungen

1.2.1 Vorwissen

- Programmiererfahrungen, vorzugsweise in einer textuellen Sprache
- Wissen über logische Operatoren (fakultativ)
- Wissen über Binärzahlen (fakultativ)

Je nach Vorwissen lassen sich bestimmte Module des Kurses verkürzen. Geben Sie Ihren Schüler*innen also Hinweise, dass sie bestimmte Inhalte schon kennen und überspringen können.

1.2.2 Technik

Die hier beschriebenen Voraussetzungen sind nötig, damit der Kurs so, wie er ist eingesetzt werden kann. Mit selbstständigen Anpassungen kann aber auch andere Hardware eingesetzt werden.

- Nötige Hardware (pro Pärchen):
  - 2 Arduino Mega mit USB-Kabel
  - 8 LEDs
  - 8 Vorwiderstände
  - ungefähr 20 Jumperkabel M-M
  - 4 Jumperkabel F-M
  - 2 Breadboards

  - 1 I2C LCD mit 2 Zeilen à 16 Zeichen
  - 2 Knöpfe
- installierte Arduino IDE

1.2.3 Moodle

Geplant ist normalerweise die Verwendung der Moodle-Instanz des Sächsischen Bildungsservers, auf die über `schullogin.de` zugegriffen werden kann. Dafür wird dann mittels der Backup-Datei des Kurses dieser als neuer Kurs installiert.

Soll eine eigene Moodle-Instanz verwendet werden, ist das Coderunner-Plugin zwingend notwendig. Außerdem ist es sinnvoll die Filter für die Auto-Verlinkung der Aktivitätsnamen und der Glossareinträge zu aktivieren.

1.3 Hinweise

Die ersten drei Module besitzen denselben Aufbau. Sie beginnen mit einem oder mehreren theoretischen Inputs und enden mit einer praktischen Arbeit. Dabei müssen entweder Programmieraufgaben in Moodle erledigt werden (Modul 1 und 3) oder Arduino-Systeme gebaut und programmiert werden (Modul 2 und 4). Außerdem enthält jedes Modul ein Glossar, das wichtige Begriffe des Moduls zusammenfasst und zur Wiederholung verfügbar macht.

Um eine Überforderung der Schüler*innen zu verhindern, werden die Module und Aktivitäten nach und nach freigeschaltet. Das geschieht in den meisten Fällen automatisch, für den Abschluss der Module 2 und 4 mit den Arduino-Aufgaben müssen Sie allerdings tätig werden. Um den Abschluss für Ihre Schüler*innen zu markieren, gehen sie im Kopfbereich des Kurses auf „Bewertungen“ (siehe Abbildung 1). Ist der Bearbeitungsmodus (oben rechts) eingeschaltet, können Sie hier Bewertungen festlegen. Um den Abschluss von Modul 2 zu markieren, tragen Sie in der entsprechenden Zeile in der Spalte *Modul 2 Abschluss* eine 1 ein und drücken Sie auf „Änderungen speichern“ (1 steht im Kurs für erledigt/ bestanden, 0 für nicht erledigt/ bestanden). Für Modul 4 ist das Vorgehen analog.

Eine kleine frontale Einführung in den Kurs kann sinnvoll sein, um beispielsweise auf bestimmte Modalitäten bezüglich der Speicherung von Sketches zu geben oder auch um eine Hinführung auf das Thema zu geben.

Zeitlich sind für den Kurs, je nach Vorwissen 3 bis 6 Unterrichtsstunden einzuplanen. Das ist ein sehr großes zeitliches Spektrum, aber der Kurs soll auch ein großes Spektrum an Vorerfahrungen bezüglich Programmierung und Physical Computing abdecken. Die konkrete zeitliche Planung bleibt also Ihnen überlassen.

2 Modul 1: Wiederholung Programmierung

Abschluss automatisch

2.1 Lernziele

Die Schüler*innen...

- wiederholen die grundlegenden Programmierkonzepte Variablen, Verzweigungen, Schleifen und Funktionen, indem sie Erklärungen und Beispiele bearbeiten.
- kennen die grundlegende Syntax der Arduino-Programmierung (C/ C++).
- kennen mathematische und logische Operatoren, sowie ihre Syntax in der Arduino-Programmierung.
- werden sich der syntaktischen Nähe verschiedener Programmiersprachen bewusst.

Abbildung 1: Kopfbereich des Kurses

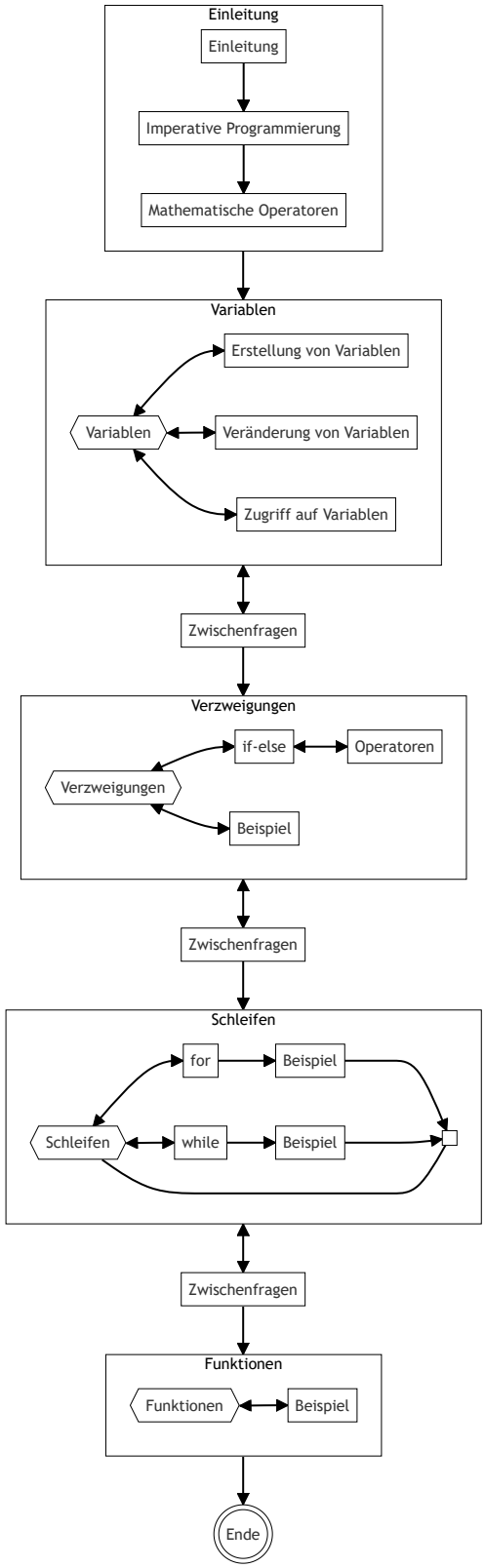
- wenden ihre Kenntnisse an, indem sie eine Lösung implementieren, um gerade Zahlen eines Arrays auf der Konsole auszugeben.

2.2 Elemente

2.2.1 Lektion „Crashkurs: imperative Programmierung in C/C++“

Die Lektion unterteilt sich in die Bereiche *Einleitung*, *Variablen*, *Verzweigungen*, *Schleifen* und *Funktionen* in denen die jeweiligen Konzepte anhand eines Beispiels erläutert werden. Die einzelnen Seiten und Übergänge sind in der Abbildung rechts detailliert aufgeschlüsselt.
Die einzelnen inhaltlichen Abschnitte werden jeweils durch Zwischenfragen verbunden, die sichern sollen, dass der vorige Input nachvollzogen wurde. Die Lösungen sind hier kurz aufgeschlüsselt:

- Falsche Zuweisungen
  - `char[] text = "Hallo";` - F
  - `int i = 2;` - R
  - `bool b = true;` - R
  - `int i < 5;` - F
- Zuordnung Datentypen
  - `4` - `int`
  - `"Imperative Programmierung"` - `char` Array
  - `'4'` - `char`
  - `true` - `bool`
- Verzweigung Frage 1 und 2
  - 10
- Schleifen Frage 1
  - 10
- Schleifen Frage 2
  - 5



2.2.2 Glossar - „Arduino (C/C++) Syntax - Nachschlagewerk“

Das Glossar enthält die Syntax der in der Lektion vorgestellten Inhalte in Kurzform zum Nachschlagen. Einige Befehle, wie `printf` werden hier auch tiefer behandelt, als in der Lektion.

2.2.3 Quiz - „Bist du bereit?“

Diese automatisiert ausgewertete Programmierfrage bildet den Abschluss des Moduls und überprüft, ob die grundlegenden Konzepte der Programmierung und die Arduino-Syntax verstanden wurden. Da es sich nur um ein Modul zur Wiederholung handelt, ist die Aufgabe vergleichsweise einfach gestaltet. Sie soll lediglich alle wichtigen Konstrukte abfragen, aber nicht viel Zeit in Anspruch nehmen.
Mögliche Lösung:

```
1  void printEven (int numbers[], int length) {
2      for (int i = 0; i < length; i++) {
3          if (numbers[i] % 2 == 0) {
4              printf("%d ", numbers[i]);
5          }
6      }
7  }
```

3 Modul 2: Informatisch Basteln mit Arduinos

Abschluss manuell

3.1 Lernziele

Die Schüler*innen...

- kennen spezifische elektrotechnische Bauteile, die für die Bearbeitung des Projekts benötigt werden, vollziehen die Kombination dieser in einem Schaltplan nach und verwenden sie um den Schaltplan nachzubauen.
- sind vertraut im Umgang mit der Arduino-IDE und verwenden diese um einen Arduino-Sketch zur Steuerung verschiedener Bauteile mit einem Arduino zu implementieren.
- schlagen die Syntax für die Programmierung eigenständig im Kursglossar bzw. in der Arduino-Referenz nach.

3.2 Elemente

3.2.1 Buch - „Hardware“

Diese Inputs behandeln, je nach Gruppe, die relevanten elektronischen Bauteile. Außerdem leiten sie dazu an einen vorgegebenen Bauplan nachzubauen.

3.2.2 Video - „Tutorial Arduino IDE“

Diese zwei Videos geben eine kurze Einweisung in die Bedienung der Arduino-IDE. Es gibt zwei Varianten des Videos für die alten Arduino-IDEs mit Version `1.8.*` und die höheren Versionen `2.*`. Je nachdem, welche IDE zum Einsatz kommt, kann die andere Version ausgeblendet werden.

3.2.3 Buch - „Arduino Programmieren“

Diese Bücher leiten die Schüler*innen durch die Erstellung des Arduino-Sketches für die Schaltung, die vorher aufgebaut wurde. Sie bieten aber auch die Möglichkeit die Sketche selbstständig zu schreiben und unterstützen dies mit Hilfestellungen.

3.2.4 Glossar - „Hardware und Arduino-Befehle“

Hier finden sich die behandelten Befehle speziell zur Steuerung der Hardware übersichtlich wieder.

4 Modul 3: Wie kommt der Text ins Kabel?

Abschluss automatisch

4.1 Lernziele

Die Schüler*innen...

- unterscheiden zwischen digitaler und analoger Datenübertragung und kennen die Prinzipien der parallelen und seriellen digitalen Datenübertragung.
- kennen die Bedeutung von Bit und Byte repräsentieren Text mithilfe vom ASCII in Binärdaten.
- implementieren die Umwandlung von Text zu Bits und andersherum mithilfe der Arduino-Befehle `bitRead` und `bitWrite`.

4.2 Elemente

4.2.1 Buch - „Text + Strom = Datenübertragung?“

Das Buch vermittelt Inhalte zu den Grundlagen der Datenübertragung. Die analoge Übertragung wird kurz angeschnitten, um anschließend näher auf digitale Übertragung und die binäre Codierung von Text einzugehen. Außerdem werden die Arduino-Befehle `bitRead` und `bitWrite` eingeführt.

4.2.2 Glossar - „Befehle und Fakten zur Signalübertragung“

Hier werden erneut wichtige Befehle und Begriffe des Moduls zusammengefasst.

4.2.3 Quiz - „bitRead Aufgabe“

Diese Aufgabe für die Gruppe der Sender beschäftigt sich damit, wie die Bits eines `char` ausgelesen und ausgegeben werden. Der hier entstandene Code kann später für die Übertragung verwendet werden.

Mögliche Lösung

```
1 void printBits(char c) {
2     for(int i = 7; i >= 0; i--) {
3         printf("%d", bitRead(c, i));
4     }
5 }
```

4.2.4 Quiz - „bitWrite Aufgabe“

Diese Aufgabe ist an die Gruppe der Empfänger gerichtet und beschäftigt sich damit die Bits eines leeren `char` zu modifizieren, sodass ein Textzeichen dabei herauskommt.

Mögliche Lösung

```
1 void printChar(int arr[]) {
2     char out = '\0';
3     for(int i = 0; i < 8; i++) {
4         bitWrite(out, 7-i, arr[i]);
5     }
6     printf("%c", out);
7 }
```

5 Modul 4: Jetzt bist du dran!

Abschluss manuell

5.1 Lernziele

Die Schüler*innen...

- entwerfen eine Schaltung aus LEDs, Widerständen und einem Arduino, mit dem Daten digital übertragen werden und die Ströme mittels der LEDs sichtbar gemacht werden können. (Sender)
- implementieren einen Sketch, der vorgegebenen Text in Binärdaten umwandelt und die GPIO-Ports des Arduino entsprechend der Daten schaltet. (Sender)
- entwerfen eine Schaltung aus einem Arduino und einem LCD, mit dem digital übertragene Daten eines Senders empfangen und sichtbar gemacht werden können. (Empfänger)
- implementieren einen Sketch, der Text in Form von Binärdaten an den GPIO-Pins des Arduinos empfängt, umwandelt und auf einem LCD ausgibt. (Empfänger)
- erkennen Probleme bei der Datenübertragung und diskutieren Lösungen für diese Probleme.

5.2 Mögliche Lösungen

Sender

```
int ausgabepins[] = {2, 3, 4, 5, 6, 7, 8, 9};
int takt = 1;
char nachricht[] = "Hallo Testnachricht";

void setup() {
  for(int i = 0; i < 8; i++) {
    pinMode(ausgabepins[i], OUTPUT);
  }

  for(int i = 0; i < sizeof(nachricht) - 1; i++) {
    for(int j = 0; j < 8; j++) {
      digitalWrite(ausgabepins[j], bitRead(nachricht[i], j));
    }
    delay(takt * 1000);
  }

  for(int j = 0; j < 8; j++) {
    digitalWrite(ausgabepins[j], LOW);
  }
}

void loop() {
}
```

Empfänger

```
#include <Ardudigit.h>

int eingangspins[] = {2, 3, 4, 5, 6, 7, 8, 9};
int takt = 1;
I2C_LCD lcd(0x27);

void setup() {
  for(int i = 0; i < 8; i++) {
    pinMode(eingangspins[i], INPUT_PULLUP);
  }
  lcd.init();
}

void loop() {
  char in;
  for(int i = 0; i < 8; i++) {
    bitWrite(in, i, digitalRead(eingangspins[i]));
  }
```

```
19
20    if(in != 0) {
21      lcd.print(in);
22      delay(takt*1000);
23    }
24  }
```

5.2.1 Hinweise

Die Hilfen werden über einen Cookie erst 10 Minuten nachdem die Seite das erste Mal aufgerufen wurde, eingeblendet. Sind Cookies deaktiviert, müssen die Hilfen im HTML Editor bearbeitet werden, damit sie standardmäßig eingeblendet sind.

Die Zusatzaufgaben sollen Schüler*innen, die den Kurs abgeschlossen haben zum Weiterdenken anregen und weitere Themen motivieren. Um mehr Lösungsmöglichkeiten offenzuhalten, wird hier auf Lösungsvorschläge verzichtet. Das gibt Ihnen die Möglichkeit vielleicht auch ohne, dass Sie selbst wissen, wie die Lösung am Ende aussieht, mit Ihren Schüler*innen zu verschiedene Ansätze zu diskutieren.