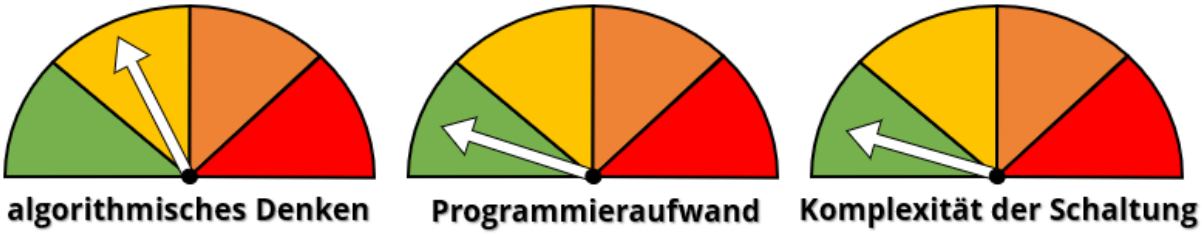
TECHNISCHE UNIVERSITÄT DRESDEN

Arduino Uno

Station 4 | 1+1=10



Professur für Didaktik der Informatik

SYLBER Synergetische Lehrerbildung im exzellenten Rahmen

Das Maßnahmenpaket „TUD-Sylber² – Synergetische Lehrerbildung im exzellenten Rahmen“ wird im Rahmen der gemeinsamen „Qualitätsoffensive Lehrerbildung“ von Bund und Ländern aus Mitteln des Bundesministeriums für Bildung und Forschung gefördert.

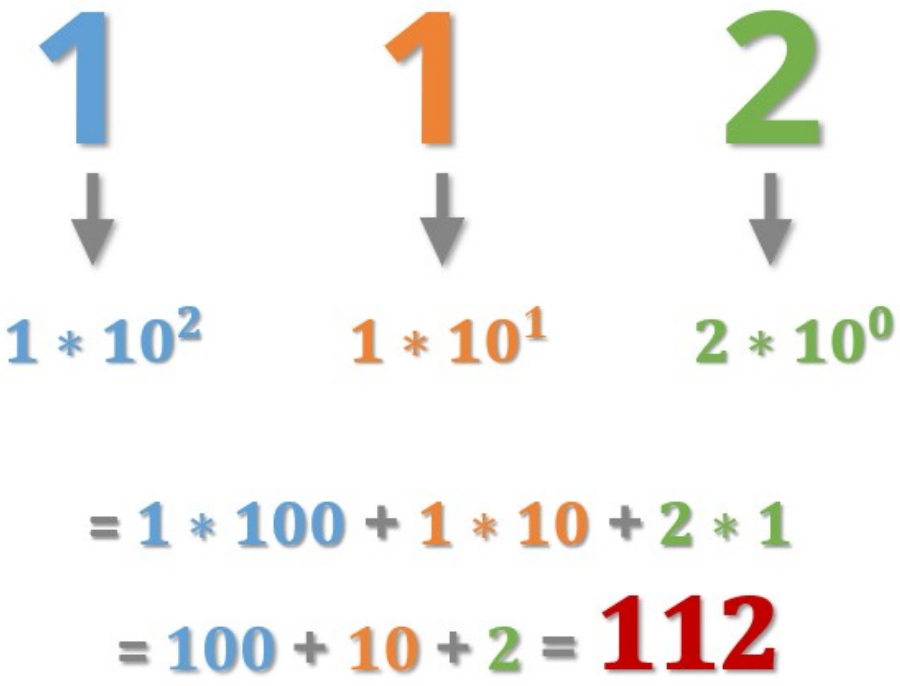
GEFÖRDERT VOM Bundesministerium für Bildung und Forschung

SEIT WANN IST 1+1 DENN 10?

Zugegeben – du weißt, genauso wie wir, dass 1+1 nicht 10 sind. Allerdings ist der Computer vor dir ganz anderer Meinung. Die Thematik dahinter nennt sich Binärzahlen – ein Zahlensystem wo es nur Nullen und Einsen gibt. An dieser Station wirst du dir zunächst einen Überblick darüber verschaffen, wie man Binärzahlen in Dezimalzahlen umwandelt. Anschließend setzt du ein Programm mit dem Arduino um, was dies für eine vierstellige Binärzahl selbstständig erledigen soll.

ÜBERBLICK ÜBER DAS DEZIMALSYSTEM

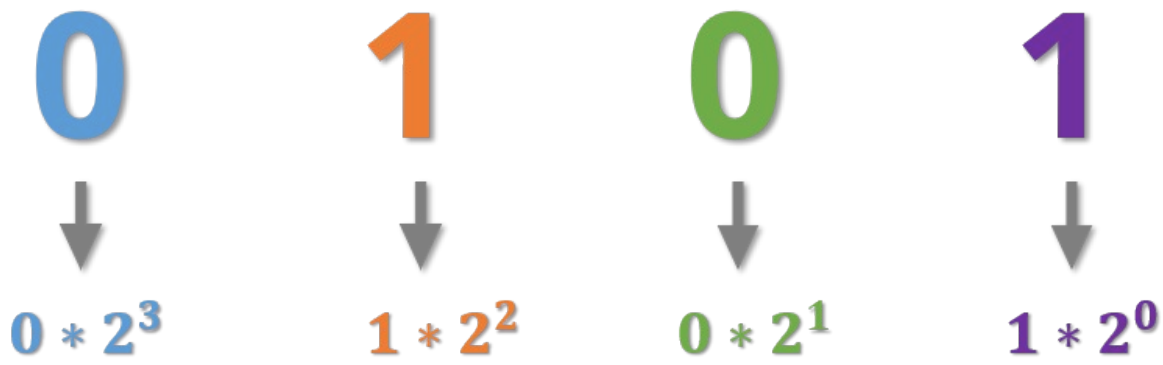
Das Dezimalsystem verwenden wir im Alltag ständig, ohne den grundsätzlichen Aufbau zu hinterfragen, dabei folgt die Struktur des Zahlsystems ganz klaren Regeln. Dies werden wir uns am Beispiel der einfachen dreistelligen Zahl „112“ genauer anschauen.



Wir können jede Zahl einer Zehnerpotenz zuordnen. Beginnend bei 10° wird die Potenz nach links immer um eins erhöht. Die Zahlen werden mit den Zehnerpotenzen multipliziert und anschließend addiert. Für das alltägliche Wissen scheint dies irrelevant zu sein, aber wir benötigen dieses konzeptuelle Wissen, um uns das **Binär**- beziehungsweise **Dualsystem** näher betrachten zu können.

DAS DUALSYSTEM UND DIE UMRECHNUNG INS DEZIMALSYSTEM

Das Dualsystem ist von der Struktur her ähnlich. Aber wir müssen nun die sogenannte Basis ändern. Das Dezimalsystem hat die Basis 10, erkennbar an den Zehnerpotenzen. Im Dualsystem wird dies geändert zur Basis 2, dass heißt wiederum, dass wir jetzt Ausdrücke wie 2^0, 2^1 und 2^2 haben. Auch dies schauen wir uns wieder in einer Übersichtsgrafik an, am Beispiel der Dualzahl „0101“.



$$= 0 * 8 + 1 * 4 + 0 * 2 + 1 * 1$$

$$= 0 + 4 + 0 + 1 = 5$$

Von rechts nach links haben wir also auch hier wieder aufsteigende Zweipotenzen. Wir multiplizieren die Zahlen 0 oder 1 mit der Zehnerpotenz und addieren die einzelnen Ergebnisse. Am Ende steht dann das Ergebnis und wir erkennen, dass die Dualzahl „0101“ der 5 im Dezimalsystem entspricht.

AUFGABE 1 – ZEIT DIE UMRECHNUNG ZU ÜBEN

Damit du das Programm dazu schreiben kannst, damit die Umrechnung für uns der Arduino übernimmt, musst du natürlich das Verfahren selbst erst mal beherrschen.

Rechne folgende Dualzahlen um: *(nutze den Platz auf der nächsten Seite)*

01000011	01101111	01100100	01100101

VON DUALZAHLEN ZU BUCHSTABEN

Es gibt internationale Standards, die die Binärkodierung von Zeichen genau festlegen. Der **ASCII-Code** (Abkürzung für American Standard Code for Information Interchange) wurde vom American National Standards Institute (ANSI) festgelegt. Er sieht (in der ursprünglichen Version) 7 Bits zur Kodierung vor - es lassen sich also 2^7 = 128 Zeichen darstellen.

Nachfolgend ist ein Ausschnitt der Codierungstabelle gezeigt. Versuche damit die in Aufgabe 1 gegeben Dualzahlen in Buchstaben umzuwandeln.

Dezimalzahl	Buchstabe
65	A
66	B
67	C
68	D
69	E

Dezimalzahl	Buchstabe
70	F
71	G
72	H
73	I
74	J

Dezimalzahl	Buchstabe
75	K
76	L
77	M
78	N
79	O

codiertes Wort: _____

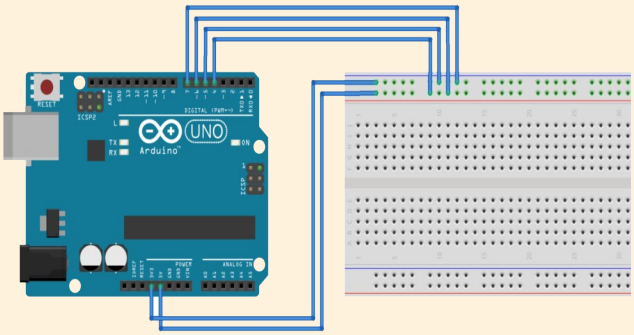
UMSETZUNG – WOHER NEHMEN WIR UNSERE DUALZAHLEN?

Wir wissen ja bereits, dass digitale Pins zwei Zustände einnehmen können, entweder HIGH oder LOW. Diese sind gleichbedeutend mit den Zahlen 1 oder 0. Wir nutzen als unsere digitalen Pins, welche wir frei mit +5V oder GND verbinden und erzeugen damit eine vierstellige Dualzahl.

AUFGABE 2 – SCHALTUNG AUFBAUEN

Verbinde zunächst die äußeren Anschlussleisten des Breadboards. Weiterhin brauchst du 4 Kabel, welche du an den digitalen Pins 4, 5, 6 und 7 ansteckst.

Je nach Position auf dem Breadboard liegt dann jeweils HIGH (1) der LOW (0) an. In der nebenstehenden Abbildung kannst du dir dazu nochmal die schematische Skizze anschauen.



VON DER UMRECHNUNG ZUM ALGORITHMUS

Für eine vorgegebene Anzahl an Pins müssen wir diese von rechts nach links auslesen und überprüfen ob HIGH oder LOW anliegt. Da wir wissen, dass eine Multiplikation mit 0 auch nur 0 ergibt, betrachten wir die Stellen an denen eine 1 ausgelesen wird. An dieser Stelle wird die 1 mit der dazugehörigen Zweierpotenz multipliziert und das Ergebnis der Stellen aufaddiert.

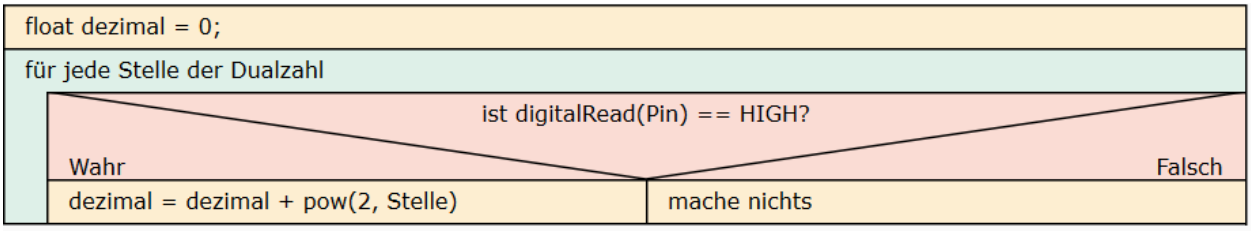
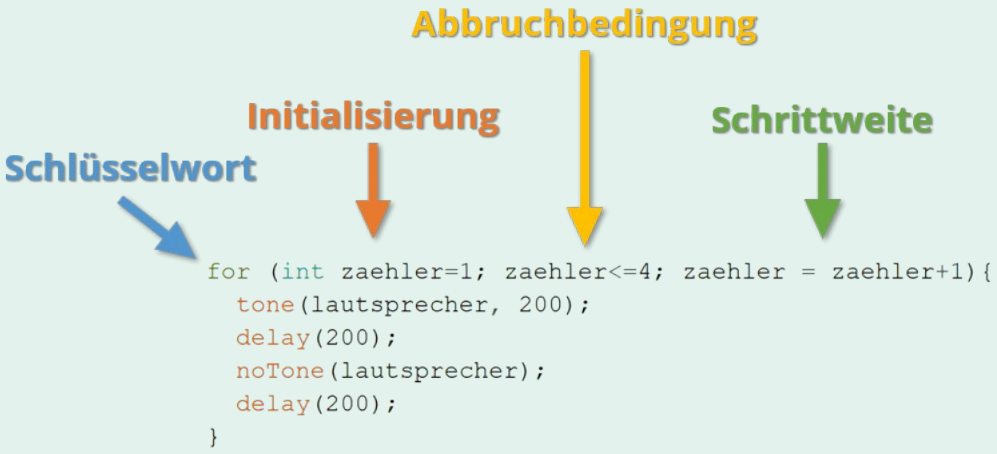


Abbildung 1: Struktogramm

Die Darstellung des Verfahrens im Struktogramm enthält zwei dir derzeit unbekannte Elemente. Zunächst ist die Struktur hinter dem grünen Element eine Zählschleife und weiterhin nutzen wir eine Funktion namens `pow()`. Das schauen wir uns jetzt im Anschluss etwas genauer an!

DIE ZÄHLSCHLEIFE

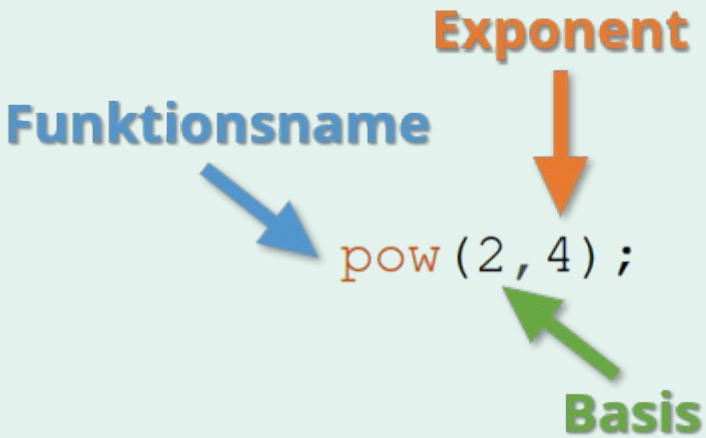
Die Zählschleife wiederholt die in ihr befindlichen Befehle so oft, wie es innerhalb der Schleifendefinition vorgegeben ist. Dazu geben wir einen Startwert (Initialisierung), eine Abbruchbedingung und die Schrittweite vor. Solange diese Bedingung erfüllt ist, werden die Befehle ausgeführt und der Zähler verändert. Beispielhaft sieht das beim Arduino wie folgt aus:



DIE `POW()`-FUNKTION

Die pow()-Funktion ist quasi nichts anderes als eine Exponentenfunktion. Dies kennst du ja bereits, auch aus dem Mathematikunterricht. So kann man beispielsweise 2^4 rechnen über: $2^4 = 2*2*2*2 = 16$

Nicht anderes macht für uns diese Funktion, wie du sie verwenden musst, siehst du in der untenstehenden Abbildung:



ENTWICKELN DES PROGRAMMCODES

Das Programm wird diesmal nur innerhalb der `setup()`-Methode ausgeführt. Warum? Da zum Umstecken der Verbindungen der Arduino sowieso vom Strom getrennt werden soll, genügt es uns die Umrechnung einmal auszuführen.

Zunächst geben wir die Umwandlung nur im seriellen Monitor aus, später dann über einen LCD-Bildschirm. Beschäftigen wir uns aber zuerst mit der Funktionalität der Konvertierung von Dualzahlen zu Dezimalzahlen.

Der erste Schritt ist die Festlegung der Pin-Modi. Wir wollen ja den digitalen Wert der 4 Pins auslesen und müssen diese daher auch als `INPUT` definieren. Zur Erinnerung, der erforderliche Code sieht wie folgt aus:

```
pinMode(4, INPUT);
```

AUFGABE 3 – LEGE DEN MODUS DER PINS FESTGELEGT

Ergänze innerhalb der `setup()`-Methode alle nötigen `pinMode()`-Befehle.

Als nächstes definieren wir uns die **Zählschleife**. Diese muss vom Pin 4 bis zum Pin 7 zählen und dabei jeden Pin einmal ansprechen. Unser Startwert ist also 4, sie wird solang ausgeführt wie der Zähler kleiner gleich 7 ist und erhöht den Zähler immer in Einer-Schritten. Als Code geschrieben ergibt sich daher folgendes:

```
for(int i = 4; i <= 7; i = i+1){
      ...
}
```

Weiterhin können wir auch das Auslesen des digitalen Pins realisieren und wollen zum Testen diesen Wert am Computer ausgeben. Um die Daten an den Computer zu übertragen nutzen wir den sogenannten **seriellen Monitor**, welcher die Informationen, die über die USB-Verbindung übertragen werden, darstellen kann. Dazu benötigst du folgende Befehle:

Befehl	Wo?	Bedeutung
`Serial.begin(9600)`	`setup()`-Methode	Startet die Verbindung zum seriellen Monitor, wobei die 9600 für die Übertragungsrate steht.

Befehl	Wo?	Bedeutung
`Serial.print("Der Arduino... ");`	`loop()`-Methode	Ausgabe einer Zeichenkette. Ohne Anstriche können auch Variablen ausgegeben werden.
`Serial.println(" … ist toll!");`	`loop()`-Methode	Gleiche Funktion wie oben, aber mit einem Zeilenumbruch am Ende.

Mit den Befehlen können wir nun eine Übertragung zum Computer realisieren.

AUFGABE 4 – PIN-MODI AUSLESEN UND AUSGEBEN

Starte zunächst den seriellen Monitor am Anfang der `setup()`-Methode mit dem Befehl: `Serial.begin(9600)` Innerhalb der Zählschleife nutzt du den Befehl `digitalRead()` und die Zählervariable `i` zum Auslesen der Pins. Speichere den Wert in einer neuen Variable und gebe diese im seriellen Monitor aus mit `Serial.println(variable)`.

EIN LETZTER SCHRITT ZUR DEZIMALZAHL

Das funktioniert schon mal? - Sehr gut! Dann fehlt nur noch die Berechnung für die eigentliche Umwandlung der Zahlensysteme.

Als erstes unterscheiden wir ob der ausgelesen Status des Pins HIGH, also 1, ist. Dazu nutzen wir die einfache Verzweigung.

```
if(_____________________________ == 1){
      ...
}
```

Ergänze auf dem Strich deine zuvor initialisierte Variable mit der `digitalRead()` - Funktion!

Anstelle der Punkte fügen wir nun die abschließende Berechnung ein. Dazu brauchen wir eine neue Variable in der wir die Dezimalzahl speichern. **ACHTUNG** – auch wenn wir eine ganze Zahl erwarten, muss der Datentyp float sein. Wenn du wissen willst warum da so ist, kannst du das nachfolgende Hinweiskästchen lesen.

DAS $7.\overline{9999}$ PROBLEM

Die `pow()`-Funktion liefert ein Ergebnis des Datentyps double zurück. Für `pow(2,3)` liefert die funktioniert den Wert $7.\overline{9999}$. Dies wird zwar in der Darstellung aufgerundet, verwenden wir aber von vornherein den Datentyp integer, also eine ganze Zahl, so werden die Nachkommastellen abgeschnitten und das Ergebnis wäre die Zahl 7, was natürlich falsch ist. Aber warum ist das so? Die Wertebereiche der Datentypen sind begrenzt und geben nur in Intervallen Zahlen an. Diese sind zwar sehr klein, aber das korrekte Ergebnis wird immer nur angenähert.

Wir haben nun bereits unsere Verzweigung und die notwendige Verzweigung die überprüft ob der ausgelesene Pin eine 1 repräsentiert. Jetzt setzen wir die Berechnung um. Zu unserer initialisierten Variable addieren wir die dazugehörige Zweierpotenz. Dies funktioniert allerdings nicht ausschließlich über:

```
dezimalzahl = pow(2,i)
```

Denn hierbei weisen wir der Variable nur einen neuen Wert zu. Wollen wir den vorherigen Wert zu unser Addition mit hinzu nehmen, sieht der Code wie folgt aus:

```
dezimalzahl = dezimalzahl + pow(2,i)
```

Jetzt fehlt nur die dazugehörige Ausgabe, dies folgt nach der Zählschleife, wenn die Addition der Zweipotenzen abgeschlossen ist und wird realisiert über:

```
Serial.println(dezimalzahl)
```

AUFGABE 5 - UMRECHNUNG PROGRAMMIEREN

Initialisiere zuerst die Variable für die Speicherung der Dezimalzahl, als Datentyp `double` und belege sie zu Beginn mit dem Wert 0. Ergänze im Anschluss die Verzweigung und die Berechnung innerhalb der Zählschleife, sowie die Ausgabe der Dezimalzahl nach der Schleife.

Teste deine Schaltung mit den folgenden Werten und trage die Ergebnisse deiner Umrechnung ein!

Binärzahl	Dezimalzahl	Ergebnis des Arduino
0010	2	_____
0110	6	_____
1101	13	_____
0101	5	_____
0100	4	_____
1001	9	_____
0001	1	_____

Alles richtig umgerechnet? Sehr gut! – Dann teste doch noch die fehlenden Binärzahlen aus!

DU HAST NOCH NICHT GENUG VON DIESER STATION?

Kein Problem - Frage die Betreuer einfach nach dem Zusatzblatt! Dort wirst du dann die Ausgabe mit Hilfe eines LCD-Bildschirms umsetzen.

Grafik auf dem Deckblatt: *Computer, Clker-Free-Vector-Images, Pixabay License, https://pixabay.com/de/service/license/, https://pixabay.com/de/vectors/computer-notebook-laptop-304146/*
Abbildung 1: *Struktog., Klaus Ramm / Thiemo Leonhardt, MIT © 2019 Didaktik der Informatik der TU Dresden, https://gitlab.com/ddi-tu-dresden/cs-school-tools/struktog/-/blob/master/license.md, https://dditools.inf.tu-dresden.de/struktog/*
Screenshots: *fritzing electronics made by easy und Arduino IDE 1.8.12 (windows)*
Alle weiteren Grafiken: *Patrick Binkert, EduInf@TUD*