Referenz Digitalelektronik

1 Steckplatine

Auf der Steckplatine (engl. *Breadboard*) können elektronische Schaltungen gesteckt werden. Auf der Platine ist ein Arduino-Mikrocontroller vorhanden. **Die Steckplatine wird immer so gedreht, dass sich der Mikrocontroller rechts befindet.**

Die mittleren Buchsen der Steckplatine sind **vertikal** miteinander verbunden, die obersten und untersten zwei Reihen **horizontal**. Die oberste Reihe ist mit der **5 V** Spannungsversorgung verbunden, die unterste mit der **Masse (0 V, GND)**.

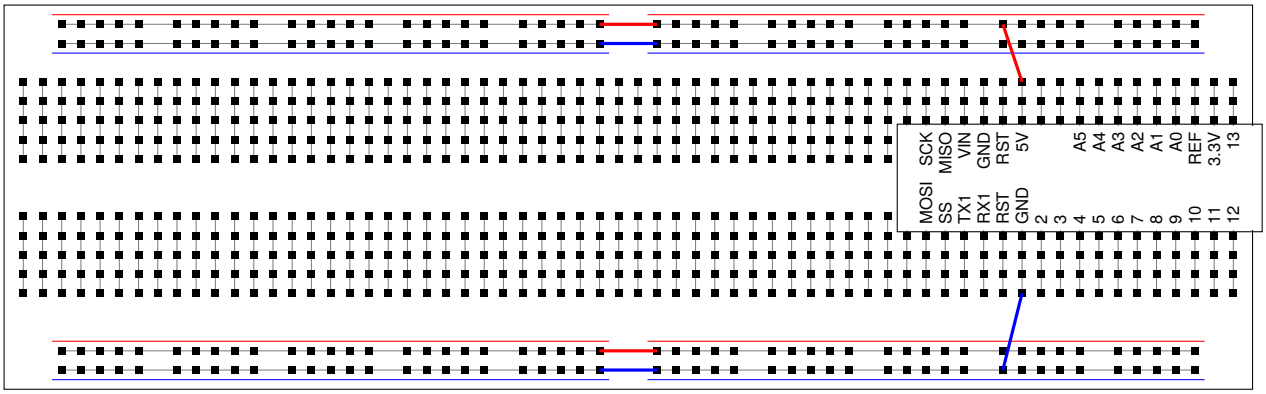


Abbildung 1 Breadboard mit Arduino Micro

2 Widerstände

Widerstände sind elektrische Bauelemente, welche einen bestimmten ohmschen Widerstand in einer Schaltung realisieren.

Abbildung 2 Widerstand

Für Schaltungen auf der Steckplatine werden Widerstände von **220 Ω** verwendet. Für eine gelötete Schaltung sollten Widerstände eingesetzt werden, welche dem benötigten Widerstandswert möglichst genau entsprechen.

2.1 Vorwiderstand

Vorwiderstände werden verwendet, um den Strom durch Leuchtdioden zu begrenzen und damit ihre Beschädigung zu verhindern.

Für eine Leuchtdiode mit einer Betriebsspannung U_L und einem maximalem Strom I an einer Spannungsquelle U berechnet sich der benötigte Vorwiderstand folgendermassen:

$$R_V = \frac{U - U_L}{I}$$

R_V	=	Widerstand	Ohm
U_V	=	Spannung am Widerstand	Volt
I	=	Strom	Ampere
U	=	Gesamtspannung	Volt
U_L	=	Spannung an der Leuchtdiode	Volt

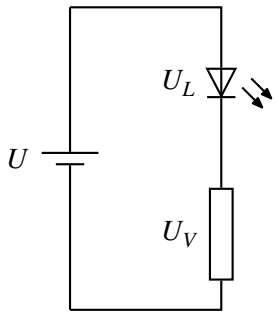


Abbildung 3 Leuchtdiode mit Vorwiderstand

2.2 Farbcodierung des Widerstandswerts

Die Grösse eines Widerstands in Ω wird durch die Farbringe gekennzeichnet. Je nach Anzahl Ringe werden diese anders interpretiert. In der (siehe *Tabelle 1*) wird die Bedeutung für Wiederstände mit fünf bis sechs Ringen zusammengefasst.

Bei Widerständen mit **vier Ringen** gilt folgende Regel:

- Der **erste Ring** gibt die Zehnerstelle b an.
- Der **zweite Ring** gibt die Einerstelle c an.
- Der **dritte Ring** gibt den Exponenten d an.
- Der **vierte Ring** gibt die Fehlertoleranz an.

Der Widerstandswert ist somit $(10b + c) \cdot 10^d$ Ω.

Ring	**1.**	**2.**	**3.**	**4.**	**5.**
Bedeutung	**Wert**	**Wert**	**Wert**	**Mult.**	**Toleranz**
Silber	-	-	-	-	±20%
Gold	-	-	-	×0.01	±10%
Schwarz	-	0	0	×0.1	±5%
Braun	1	1	1	×1	-
Rot	2	2	2	×10	±1%
Orange	3	3	3	×100	±2%
Gelb	4	4	4	×1 k	-
Grün	5	5	5	×10 k	±0.5%
Blau	6	6	6	×100 k	±0.25%
Violett	7	7	7	×1 M	±0.1%
Grau	8	8	8	×10 M	±0.05%
Weiss	9	9	9	×100 M	-

Tabelle 1 Farbcodierung

Bei Widerständen mit **fünf oder sechs Ringen** gilt folgende Regel:

- Der **erste Ring** gibt die Hunderterstelle a an.
- Der **zweite Ring** gibt die Zehnerstelle b an.
- Der **dritte Ring** gibt die Einerstelle c an.
- Der **vierte Ring** gibt den Exponenten d an.
- Der **fünfte Ring** gibt die Fehlertoleranz an.

Somit ist der Widerstandswert $(100a + 10b + c) \cdot 10^d\ \Omega$.

Bei vielen Widerständen erkennt man den Toleranzring daran, dass er sich auf der Verdickung des Widerstandes befindet. Möglich ist auch, dass er in einer etwas grösseren Distanz zu den übrigen Ringen sitzt oder dass er breiter ist. Auf *www.bader-frankfurt.de* kann man den Widerstands-Farbcode online berechnen.

3 Leuchtdioden (LEDs)

Leuchtdioden (kurz LED, engl. *light-emitting diode*) sind Bauelemente, welche Licht abgeben, wenn Strom in einer bestimmten Richtung durch sie fliesst. In der anderen Richtung fliesst kein Strom durch die Leuchtdiode. Der Anschluss, an welchem die Spannung angelegt wird, heisst Anode, der Anschluss, an welchem die Masse angelegt wird, heisst Kathode.

3.1 Einzelne Leuchtdiode (LED)

Für Leuchtdioden muss immer ein Vorwiderstand verwendet werden.

Die hier verwendeten Leuchtdioden benötigen eine Spannung von 1.6 V und dürfen mit einem Strom von maximal 20 mA betrieben werden.

Abbildung 4 Leuchtdiode

Unterscheidungsmöglichkeit von Anode und Kathode: Normalerweise ist der Anschlussdraht der Anode etwas länger als derjenige der Kathode.

3.2 Segmentanzeige

Für Segmentanzeigen muss immer ein Vorwiderstand verwendet werden.

Mit Segmentanzeigen können durch Ansteuerung einzelner oder mehrerer Segmente Ziffern oder Buchstaben dargestellt werden. Wir verwenden eine sogenannte 7-Segmentanzeige, mit welcher die Ziffern so dargestellt werden können:

Abbildung 5 Darstellung der Ziffern mit einer Segementanzeige

Jedes Segment der Anzeige besteht aus einer einzelnen Leuchtdiode. Um Anschlüsse zu sparen, werden Ausgänge (Kathoden) der einzelnen Leuchtdioden zusammengefasst. Die hier benutzte Segmentanzeige besitzt zusätzlich zu den sieben Segmenten für die Ziffern einen Dezimalpunkt.

Abbildung 6 Schaltung der Segmentanzeige

Die Segmentanzeige hat insgesamt zehn Eingänge, je fünf oben und unten. *Abbildung 6* zeigt die Pinbelegung. Die einzelnen Segmente werden üblicherweise mit den Buchstaben **a** bis **g** bezeichnet, **dp** steht für den Dezimalpunkt, **GND** wie üblich für die Masse (gemeinsame Kathode).

4 Integrierte Schaltkreise (ICs)

4.1 Dual in-line Package (DIP)

Der Begriff *Dual in-line Package* oder kurz *DIP* bezeichnet die Gehäuseform für elektronische Bauteile mit zwei Reihen von Anschlussstiften (*Pins*). DIPs besitzen eine Kerbe, welche immer links liegt. Die Pins sind von unten links im Gegenuhrzeigersinn nach oben links durchnummeriert.

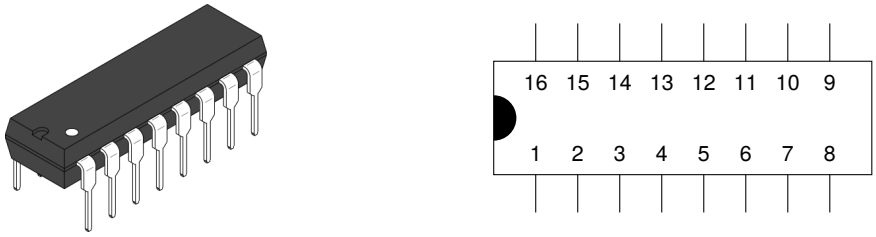
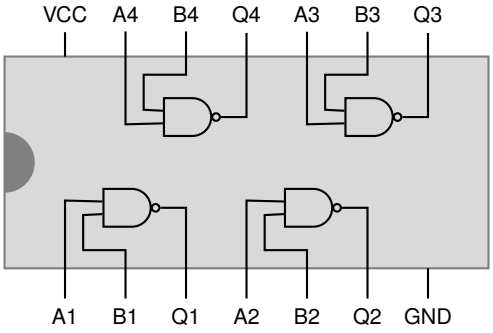


Abbildung 7 Dual in-line Package

4.2 74HC00 NAND

Der 74HC00-Chip enthält vier NAND-Logikgatter. *Abbildung 8* zeigt die Pinbelegung des Chips.

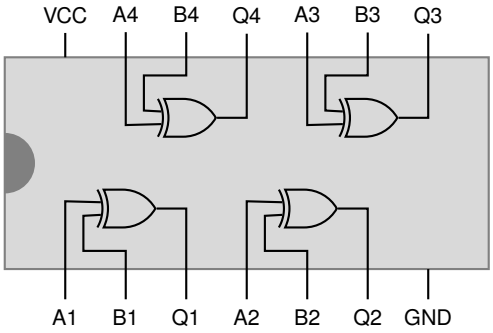


Symbol	Bedeutung
VCC	Versorgungsspannug 5 V
GND	Masse
A1 bis A4	erster Eingang des NAND-Gatters
B1 bis B4	zweiter Eingang des NAND-Gatters
Q1 bis Q4	Ausgang des NAND-Gatters

Abbildung 8 74HC00 NAND-Chip

4.3 74HC86B1 XOR

Der 74HC86B1-Chip enthält vier XOR-Logikgatter. *Abbildung 9* zeigt die Pinbelegung des Chips.

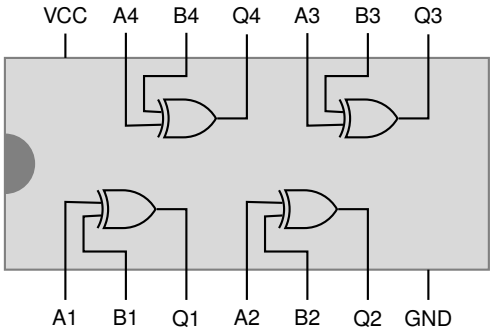


Symbol	Bedeutung
VCC	Versorgungsspannug 5 V
GND	Masse
A1 bis A4	erster Eingang des XOR-Gatters
B1 bis B4	zweiter Eingang des XOR-Gatters
Q1 bis Q4	Ausgang des XOR-Gatters

Abbildung 9 74HC86B1 XOR-Chip

4.4 74HC08B1 AND

Der 74HC08B1-Chip enthält vier AND-Logikgatter. *Abbildung 10* zeigt die Pinbelegung des Chips.



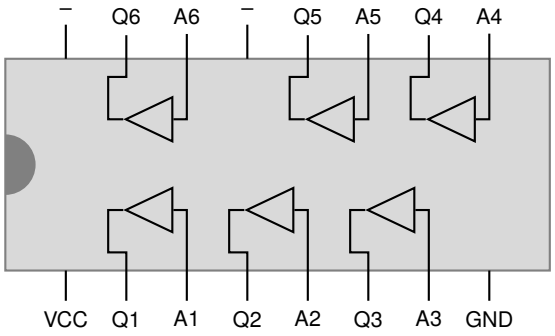
Symbol	Bedeutung
VCC	Versorgungsspannug 5 V
GND	Masse
A1 bis A4	erster Eingang des AND-Gatters
B1 bis B4	zweiter Eingang des AND-Gatters
Q1 bis Q4	Ausgang des AND-Gatters

Abbildung 10 74HC08B1 AND-Chip

4.5 74HC4050 Operationsverstärker

Der 74HC4050-Chip enthält sechs Operationsverstärker. Ein Operationsverstärker gibt ein digitales Eingabesignal identisch aus, jedoch mit der Betriebspannung des Chips. Der Chip hat eine relativ grosse Toleranz bei der Eingabe- und Betriebsspannung.

Somit kann ein Operationsverstärker eingesetzt werden, um eine mit 3.3 V betriebene Komponente (z.B. eine SD-Karte) in eine 5 V-Digitalschaltung zu integrieren.



Symbol	Bedeutung
VCC	Betriebspannung
GND	Masse
A1 bis A6	Eingang des Operationsverstärkers
Q1 bis Q6	Ausgang des Operationsverstärkers

Abbildung 11 74HC4050 Operationsverstärker

4.6 74HC595 Schieberegister

Der 74HC595-Chip enthält ein 8-Bit Schieberegister. Der Chip wird verwendet, um ein serielles Eingabesignal in acht parallele Ausgabesignale umzuwandeln.

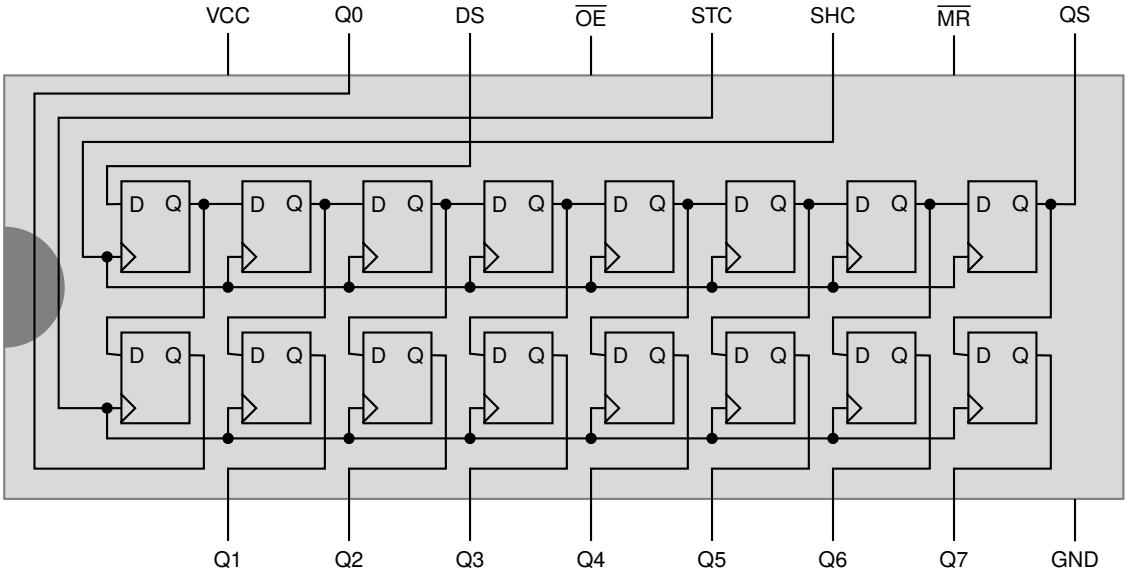


Abbildung 12 74HC595 Schieberegister

Am seriellen Eingang **Q0** wird das Eingabesignal angelegt. Bei einem Spannungswechsel von 0 V auf 5 V am Schiebetakt **SHC** (*shift clock*) wird der Zustand des Eingangs im ersten Schiebepuffer gespeichert. Durch siebenmaliges Wiederholen dieses Vorgangs werden die restlichen sieben Schiebepuffer gefüllt. Durch den Speichertakt **STC** (*storage clock*) werden die acht Bit in den Ausgabepuffer übertragen.

Das inverse Löschsignal **$\overline{\text{MR}}$** muss immer auf die Betriebsspannung 5 V geschaltet sein, die inverse Aktivierung der Ausgabe **$\overline{\text{OE}}$** auf die Masse.

Pin	**Bedeutung**	**Anschluss**
VCC	Betriebspannung 5 V	Betriebspannung 5 V
GND	Masse (*ground*)	Masse
DS	Eingabe (*serial data input*)	
SHC	Schiebetakt (*shift register clock*)	
STC	Speichertakt (*storage register clock*)	
$\overline{\text{MR}}$	Löschsignal (*master reset*)	Betriebspannung 5 V
$\overline{\text{OE}}$	Aktivierung der Ausgabe (*output enable*)	Masse
Q0 bis Q7	Parallele Ausgabe	
QS	Serielle Ausgabe (*serial output*)	

Tabelle 2 74HC595: Pinbelegung

5 Arduino Micro

5.1 Pinbelegung

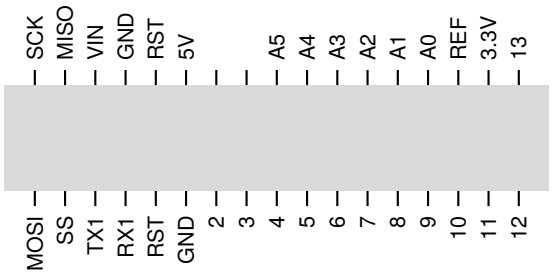


Abbildung 13 Pinbelegung des Arduino Micro

5.2 Programmstruktur

Jedes Arduino-Programm besitzt die folgenden zwei Funktionen:

- **`setup()`** wird einmal beim Einschalten und beim *Reset* ausgeführt.
- **`loop()`** wird anschliessend immer von neuem aufgerufen.

```
void setup() {
}

void loop() {
}
```

5.3 Konstanten

Konstanten werden mit der Präprozessor-Direktive **`#define`** definiert:

```
#define Name Wert
```

Um die Konstante **`MYPIN`** für den Wert **`10`** zu definieren, schreibt man:

```
#define MYPIN 10
```

5.4 Basistypen

Zur Darstellung von ganzen Zahlen können folgende Typen verwendet werden:

Symbol	Bedeutung	Konstante
VIN	Spannungsversorgung 7 V bis 12 V	
GND	Masse (0 V)	
5V	Regulierte Spannungsversorgung 5 V	
3.3V	Regulierte Spannungsversorgung 3.3 V	
RST	Reset	
REF	Analoge Referenzspannung	
RX1	Digitaler Ein-/Ausgang	`0`
TX1	Digitaler Ein-/Ausgang	`1`
2	Digitaler Ein-/Ausgang	`2`
3	Digitaler Ein-/Ausgang	`3`
4	Digitaler Ein-/Ausgang (PWM) / Analoger Eingang	`4` / `A6`
5	Digitaler Ein-/Ausgang (PWM)	`5`
6	Digitaler Ein-/Ausgang (PWM) / Analoger Eingang	`6` / `A7`
7	Digitaler Ein-/Ausgang	`7`
8	Digitaler Ein-/Ausgang / Analoger Eingang	`8` / `A8`
9	Digitaler Ein-/Ausgang (PWM) / Analoger Eingang	`9` / `A9`
10	Digitaler Ein-/Ausgang (PWM) / Analoger Eingang	`10` / `A10`
11	Digitaler Ein-/Ausgang (PWM)	`11`
12	Digitaler Ein-/Ausgang (PWM) / Analoger Eingang	`12` / `A11`
13	Digitaler Ein-/Ausgang (PWM)	`13`
A0	Analoger Eingang	`A0`
A1	Analoger Eingang	`A1`
A2	Analoger Eingang	`A2`
A3	Analoger Eingang	`A3`
A4	Analoger Eingang	`A4`
A5	Analoger Eingang	`A5`
MOSI	SPI-Schnittstelle (*Master Out Slave In*)	
MISO	SPI-Schnittstelle (*Master In Slave Out*)	
SCK	SPI-Schnittstelle	

Tabelle 3 Pinbelegung des Arduino Micro

Typ	Minimum	Maximum
`byte`	0	255
`int`	–32’768	32’767
`unsigned int`	0	65’535
`long`	-2’147’483’648	2’147’483’647
`unsigned long`	0	4’294’967’295

Tabelle 4 Basistypen

5.5 Arrays

Arrays können so definiert und benutzt werden:

```
const int VALUE[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

VALUE[0] = 25;
int val = VALUE[3];
```

Achtung: Die eckigen Klammern stehen hinter dem Namen der Variable, nicht hinter dem Typ wie in Java.

Die Länge eines *Arrays* mit Hilfe des **`sizeof`**-Operators ermittelt. Dieser liefert die Grösse einer Variable in Byte zurück. Die Anzahl Elemente erhält man, indem die Grösse des *Arrays* durch die Grösse eines Elements geteilt wird:

```
const int VALUE_COUNT = sizeof(VALUE) / sizeof(int);
```

5.6 Digitale Ausgabe

Um einen digitalen Anschluss (*Pin*) benutzen zu können, muss erst sein Modus festgelegt werden. Dies geschieht mit folgendem Befehl:

```
pinMode(Pin, OUTPUT);
```

Für **Pin** muss die dem Anschluss entsprechende Konstante (**`2`** bis **`13`**) angegeben werden.

Anschliessend kann die am *Pin* anliegende Spannung mit folgendem Befehl beeinflusst werden:

```
digitalWrite(Pin, HIGH);
// ...
digitalWrite(Pin, LOW);
```

Wird **`HIGH`** angegeben, so wird eine Spannung von 5 V am *Pin* angelegt, bei **`LOW`** eine Spannung von 0 V.

5.7 Warten und Zeitmessung

Die Funktion **`delay()`** kann verwendet werden, um die Ausführung des Programms um eine bestimmte Zeit zu verzögern. Die gewünschte Wartezeit wird in Millisekunden als Parameter übergeben:

```
delay(100);
```

Die Funktion **`millis()`** liefert die vergangene Zeit seit dem Start des Programms in Millisekunden als **`unsigned long`** zurück:

```
unsigned long now = millis();
```

Wenn ein Befehl in einem bestimmten Intervall ausgeführt, der Programmfluss aber nicht unterbrochen werden soll, kann dies mithilfe von **`millis()`** so erreicht werden:

```
unsigned long waitUntil = 0;

void loop() {
    unsigned long now = millis();
    if (waitUntil < now) {
        // Do something
        waitUntil = now + 100;
    }
}
```