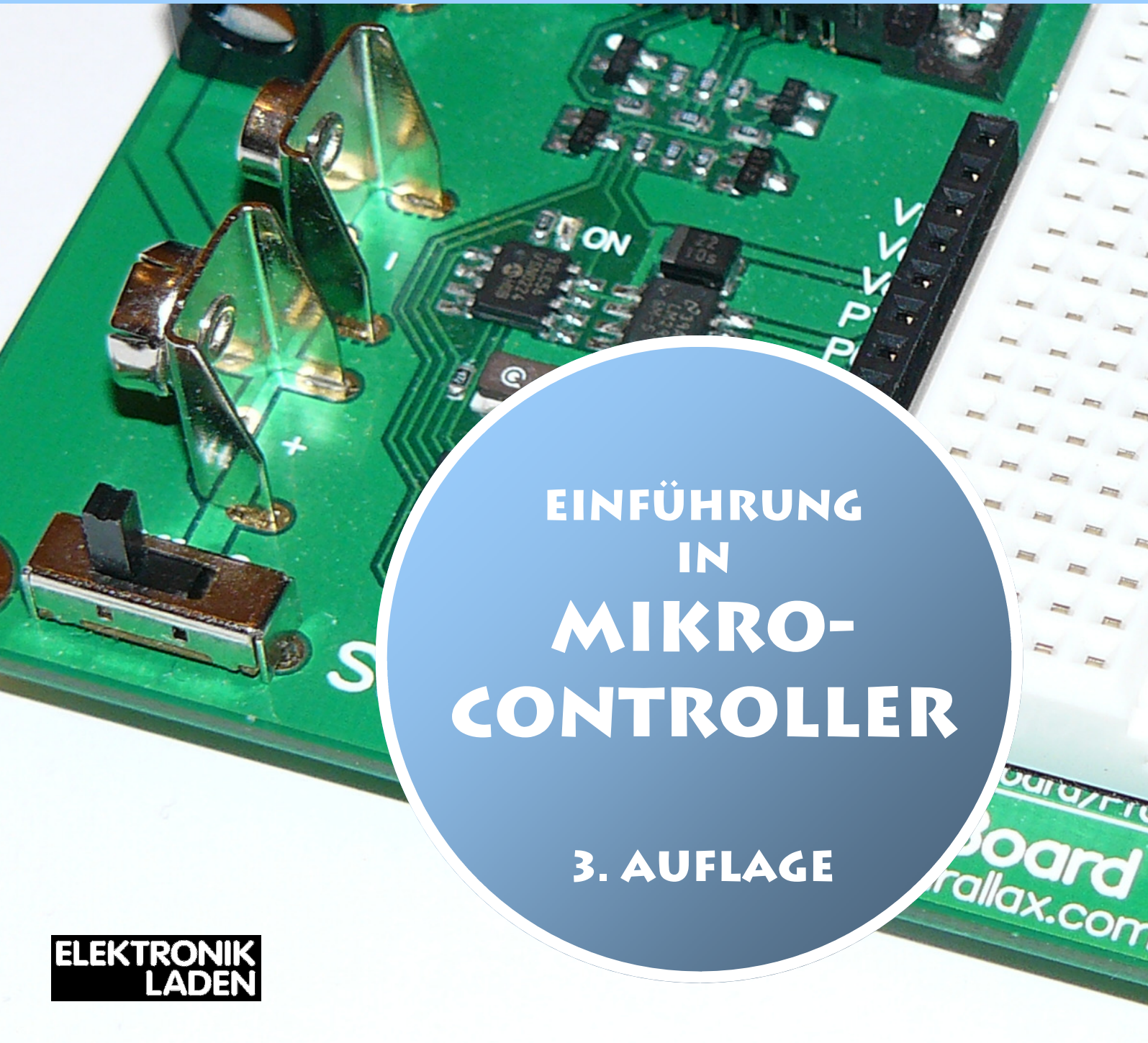
LEHRERHEFT

Fortbildungsreihe
NwT - Technik

NWT
KLASSE 9

EINFÜHRUNGSHEFT FÜR SCHÜLERINNEN UND SCHÜLER



# EINFÜHRUNG IN MIKRO-CONTROLLER

3. AUFLAGE

ELEKTRONIK LADEN

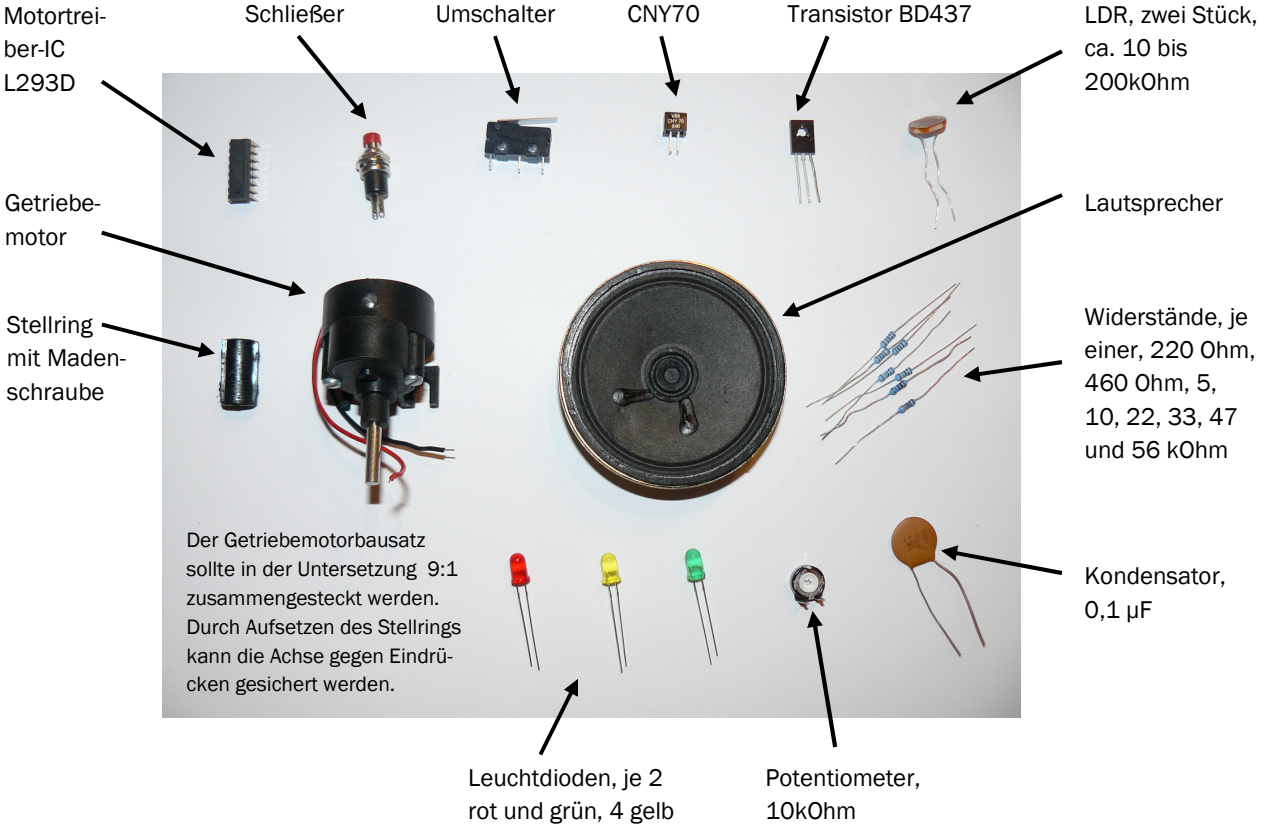
## Liebe Kollegin, lieber Kollege:

Es freut uns, dass Sie sich für Mikrocontroller als Unterrichtsinhalt interessieren. Dieses Heft (auch unter www.lehrerfortbildung-bw.de/faecher/nwt/fb) soll Ihnen die ersten Unterrichtsdurchgänge zu diesem Thema erleichtern. Die vorliegende dritte Auflage dieses Hefts ist als Lehrer– und als Schülerausgabe erhältlich. Einziger Unterschied: Dem Schülerheft fehlt das obere Drittel jeder Seite. In diesem Drittel finden sich Hinweise zu typischen Schülerschwierigkeiten und die Lösungen der Aufgaben.

Das benötigte Material für die Arbeit mit diesem Heft besteht pro Schülerpaar aus dem BS1 Project Board (z.B. von www.elmicro.com/de/bs1-project-board.html), einem RS232-USB-Adapterkabel (je nach Quelle für ca. 5 Euro) sowie etlichen der unten abgebildeten Kleinteile. Diese sind bei www.traudl-riess.de in einer alten Zusammenstellung für die Auflagen 1 und 2 dieses Hefts (Bestellnr. 18.280.0, ca. 12€) und einer neuen Zusammenstellung (Bestellnr. 18.284.0) als Set bestellbar. Diese neue Zusammenstellung verwendet insbesondere einen störungsärmeren Solar-Getriebemotor. Zusätzlich zu diesem Set werden pro Gruppe einige Krokodilklemmenkabel, einfaches Werkzeug, eine 9V-Batterie oder ein Netzteil (6-12 V DC) sowie für das letzte Kapitel ein Modellbauservo (je nach Quelle ca. 5 Euro) benötigt.

Wir danken für die eingegangenen Korrekturvorschläge und freuen uns über weitere Hinweise an: bs1@NwTF.de.

Rainer Kügele & Jochen Wegenast





### Impressum

Rainer Kügele

Rotteck-Gymnasium Freiburg
Lessingstraße 16, D-79100 Freiburg

Jochen Wegenast

Heinrich-von-Zügel-Gymnasium
Rudi-Gehring-Straße 1, D-71540 Murrhardt

**Druck und Verteilung dieser Broschüre mit freundlicher Unterstützung von:**

Elektronikladen | ELMICRO
http://elmicro.com

Dritte Auflage, © August 2010 - Version 3.2




Aus Platzgründen steht in diesem Heft häufig die männliche Form für beide Geschlechter.

## Das Heft im Rahmen einer Unterrichtseinheit

Eine Unterrichtseinheit „Mikrocontroller“ eignet sich ganz hervorragend zu anspruchsvoller Projektarbeit: Der Umgang mit dem Mikrocontroller und elektronischen Bauteilen kann unter Zuhilfenahme dieses Hefts erlernt werden. Dabei - so ist das Heft gestaltet - werden nur Grundlagen erworben, die geradezu danach schreien, in Projektarbeit zu einem Produkt, z.B. einem Fahrzeug, um- und zusammengesetzt zu werden.

Die Arbeit mit diesem Heft kann wie mit einem Schulbuch erfolgen (gut für den Anfang), im Prinzip können sich die Schüler aber auch weitgehend eigenständig in etwa 10 bis 15 Doppelstunden durch das Heft arbeiten. Dabei sind Phasen gemeinsamer Klärung in jeder Doppelstunde vorzusehen. Für die praktischen Aufgaben eignen sich Zweiergruppen. In größeren Gruppen funktioniert die gemeinsame Arbeit vor dem Computer schlecht.

Vor der Arbeit mit dem Heft müssen die Grundlagen der Elektrizität gut verstanden sein: Potential, Spannung, technische Stromrichtung, Stromstärke, Widerstand, Ohmsches Gesetz.

Es hat sich nicht bewährt, den SchülerInnen den genauen Projektauftrag bereits vor der Einarbeitung in die Mikrocontroller zu nennen, da dann insbesondere Jungen dazu tendieren, nur die Teile des Heftes zu bearbeiten, die ihnen für das Projekt direkt nützlich erscheinen. Es ist aber motivierend, die Umsetzung in einem Projekt anzukündigen.



## Inhaltsverzeichnis

Dieses Heft soll euch dabei helfen, den Umgang mit einem Mikrocontroller und anderen Bauteilen der Digitalelektronik zu erlernen. Dazu gehört, Programmieren zu erlernen, elektronische Bauteile zu verstehen und Schaltbilder lesen und erstellen zu können. Ihr werdet feststellen, dass all das bei gründlicher Erarbeitung nicht besonders kompliziert und ziemlich faszinierend ist. Und ihr werdet viel darüber lernen, wie automatische Geräte, vom Aufzug bis zum Roboter, eigentlich funktionieren.

Man kann den Schülern empfehlen, während der Einarbeitung in die Mikrocontroller ein Notizheft zu führen, in das Erkenntnisse notiert werden. Anreiz: Dieses Notizheft, in das nur geschrieben und nichts eingeklebt werden darf, kann dann bei der Klausur verwendet werden.

## Die Mikrocontroller-Platine

Wenn man den Schülern das gesamte Material zu Beginn der Arbeitsphase ausgibt, sollte man darauf hinweisen, dass die LEDs nicht direkt an die 9V-Batterie angeschlossen werden dürfen.

Das BS1 Project Board wird mit einem selbstklebenden Steckbrett und Gummifüßen geliefert. Das Steckbrett sollte, wie es die Grafik unten zeigt, auf das Lochrasterfeld geklebt werden.

Als Unterrichtseinstieg kann man Schüler die Funktionsweise eines „automatischen“ Geräts erläutern lassen, z.B. eine Schiebetüre oder einen Aufzug. Sie können sich die mechanische Umsetzung vorstellen - scheitern aber bei der Beschreibung einer elektrischen Schaltung zur Automatisierung. Das leisten oftmals Mikrocontroller, weshalb es so wichtig ist, diese kennen zu lernen.

Die Basic Stamp verdankt ihren Namen einerseits der Programmiersprache Basic, in der sie sich programmieren lässt, und andererseits ihrer Größe: als Mikrocontrollermodul (ohne das Project Board) ist sie kaum größer als eine Briefmarke (Stamp). Die BS1 ist dabei die kleinste und älteste (1992) Variante einer ganzen Mikrocontrollerfamilie. Weitere heißen BS2, BS2e, BS2sx...

# 1 Einführung in Mikrocontroller


Die kurzen Anschlüsse eines Mikrocontrollers und mancher anderer elektronischer Bauteile bezeichnet man als Pins. Der hier abgebildete Mikrocontroller hat über 60 Pins.

Haben Bauteile lange Anschlüsse, nennt man diese Anschlussdrähte.

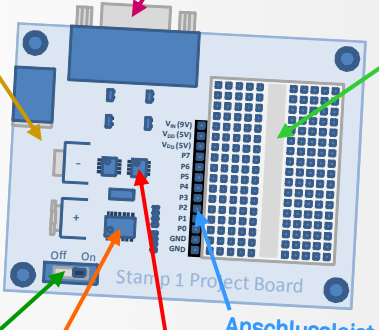
## Die Mikrocontrollerplatine BS1 Project Board

Betrachtet zu Beginn die Mikrocontrollerplatine, genannt auch **Basic Stamp 1 Project Board** oder kurz **BS1**. Der **Mikrocontroller** ist eigentlich nur eines der elektronischen Bauteile auf der Platine. Er benötigt, um zuverlässig zu funktionieren, allerdings einige andere elektronische Bauelemente um sich herum. Einen Mikrocontroller in Verbindung mit all seinen Hilfsbauteilen bezeichnet man auch als „**embedded system**“. Und euer „embedded system“ heißt eben BS1.

Der Mikrocontroller selbst heißt **PIC16** und verfügt über 20 **Pin**s (Anschlüsse), von denen acht als sogenannte **Port**s, also als Ein- oder Ausgänge für elektrische Signale dienen. Die folgende Abbildung stellt euch die BS1 vor:

Anschlüsse für die Energieversorgung: Man darf entweder eine 9V-Blockbatterie oder ein passendes Netzteil anschließen - aber nicht beides zugleich.

Die „serielle Schnittstelle“ für den Anschluss an den PC.




Experimentierplatine, oft auch Steckbrett oder Breadboard genannt. Hier kann man seine eigenen elektronischen Schaltungen aufbauen.

Ein/Ausschalter der gesamten Platine. Während des Anschließens einer Energieversorgung (Batterie oder Netzteil) oder des Computers bitte immer ausschalten. Meistens passiert zwar nichts, aber der Hersteller empfiehlt es so. Wenn man die Platine hier einschaltet, beginnt der Mikrocontroller mit der Abarbeitung des in ihm gespeicherten Programms.

Anschlussleiste für die Ports P0 bis P7 des Mikrocontrollers. Außerdem gibt es hier auch Anschlüsse für Spannungen für eigene Experimente.

Der eigentliche Mikrocontroller, ein Exemplar der sogenannten PIC16-Baureihe. Er arbeitet mit einer Taktfrequenz von 4.0 MHz, die vom darüber liegenden Bauteil erzeugt wird. Mikroprozessoren in Computern arbeiten bis zu 1000 Mal schneller, benötigen aber eine viel größere Zahl an Hilfsbauteilen.

Spannungsregler: Der Mikrocontroller, etliche der Hilfsbauteile und auch viele andere elektronische Komponenten arbeiten nicht bei einer Spannung von 9V, wie sie die angeschlossene Batterie zur Verfügung stellt, sondern bei einer Spannung von 5V. Der Spannungsregler erzeugt aus der Spannung von 9V exakt 5V, kann aber nur maximal 50 mA abgeben. Bei Überlastung wird er heiß und schaltet sich schließlich ab.

µC

Für Mikrocontroller schreibt man oft auch die Abkürzung **µC**. Der erste Buchstabe in dieser Abkürzung ist das griechische kleine m, gesprochen als mü.

## Vorab eine Einführung geben

Aus Gründen zeitlicher Effizienz ist es geschickt, den Schülern im Unterrichtsgespräch zu erklären, was ein Mikrocontroller macht und einmal vorzuführen, wie man ein einfaches BASIC-Programm vom PC auf die Basic Stamp überträgt. Wichtig dabei:

1. Ein Mikrocontroller wird programmiert. Ein Programm ist eine Liste von Anweisungen, die Zeile für Zeile abgearbeitet werden.

2. In der ersten Zeile des Programms muss die Compilerdirektive stehen, die angibt, für welchen Mikrocontrollertyp das Programm gedacht ist. Wenn man auf das Icon der BS1 klickt, wird sie automatisch eingetragen.

3. Ist die Kabelverbindung hergestellt und der Mikrocontroller eingeschaltet, lässt sich das Programm durch einen Klick auf den blauen Pfeil auf die BS1 übertragen.

Wenn die Schüler das einmal gesehen haben, stellt das keine Schwierigkeit mehr dar.

**Hinweis zu 2.2:**
Um zu verdeutlichen, dass das Programm im µC gespeichert wird, können einige der entstandenen Variationen am Lehrercomputer vorgeführt werden. Dazu müssen Sie nur die BS1 der Schüler bei gestartetem Editor am Lehrerrechner in Betrieb nehmen. Man sollte das Übertragungsfenster am Computer aber nicht schließen.

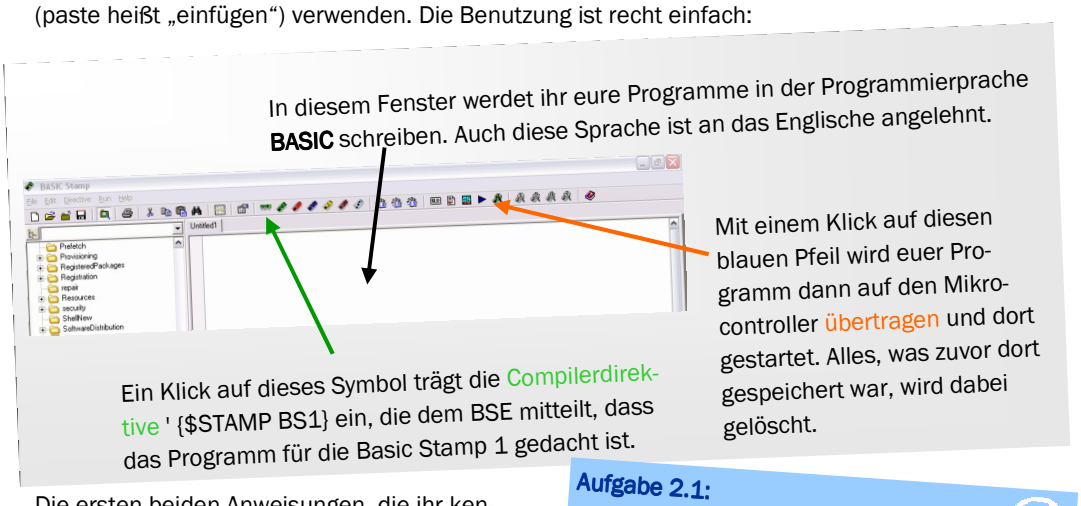
**Lösung 2.3:**
Wird die BS1 aus- und wieder eingeschaltet, wird das Programm erneut ausgeführt und die Texte werden erneut an den Computer gesendet.

# Einführung in Mikrocontroller

2

## Euer erstes Programm

Programme für Mikrocontroller werden in einer sogenannten **Entwicklungsumgebung** auf einem Computer geschrieben und dann auf den Mikrocontroller übertragen. Der **Basic Stamp Editor** (**BSE**) ist die Entwicklungsumgebung für alle Arten von Basic Stamp Mikrocontrollern und lässt sich wie jedes andere Programm bedienen. In den Menüs File und Edit kann man Dateien speichern (save), öffnen (open) und ausdrucken (print) sowie die Funktionen der Zwischenablage (paste heißt „einfügen“) verwenden. Die Benutzung ist recht einfach:




Die ersten beiden Anweisungen, die ihr kennen lernt, heißen **DEBUG** und **PAUSE**:

DEBUG sendet vom Mikrocontroller eine Botschaft an einen angeschlossenen Computer. Dieser Text (Fachbegriff **String**) muss dabei in „doppelte Anführungszeichen“ eingeschlossen werden. Beispiel:

```
DEBUG “Karl ist klug.“
```

Eure zweite Anweisung erzeugt eine Pause. Wie lang diese Pause ist, schreibt man einfach als sog. Parameter dahinter. Beispiel:

```
PAUSE 500
```

lässt den Mikrocontroller 500 Millisekunden warten, bevor er mit der nächsten Zeile fortfährt. Natürlich kann man auch andere (natürliche) Zahlen einsetzen.

**Aufgabe 2.1:**

Was wird dieses Programm machen?

```
' {$STAMP BS1}
DEBUG CR, "PARADOXON"
DEBUG CR, "Es ist dies ein Gedicht,"
DEBUG CR, "das reimt sich "
PAUSE 2000
DEBUG "nicht."
END
```

**Aufgabe 2.2:**

Tippt dieses Programm in den BSE, übertragt es auf den Mikrocontroller und führt es aus. Gerne dürft ihr es kreativ verändern.

**Aufgabe 2.3:**

Wie kann man zeigen, dass tatsächlich der Mikrocontroller das Programm ausführt und nicht der Computer einfach etwas anzeigt?

Tipp:
Wenn man in der DEBUG-Anweisung zunächst CR, dann ein Komma und dann den Text in Anführungszeichen schreibt, beginnt der Text in einer neuen Zeile. Beispiel:
DEBUG CR, “Hallo“

END:
Am Ende eines Programmes schreiben ordentliche Programmierer die Anweisung END. Das Programm würde so oder so enden, schließlich sind keine Anweisungen mehr da - aber END schaltet den Mikrocontroller in einen Stromsparmodus, so dass er beim Nichtstun weniger Energie verbraucht.

Wie lang kann ein Programm sein?
Die BS1 ist ein ziemlich kleiner Mikrocontroller und hat nur einen kleinen Speicher. Obwohl der BASIC-Code vor der Übertragung stark komprimiert wird, können nur etwa 80 - 120 Anweisungen gespeichert werden - ein bisschen hängt es davon ab, welche das sind.

**Lösung 3.1:**
B2 = 24 und B3 = 36

**Lösung 3.2:**
Die BS1 beherrscht keine Punkt-vor-Strich-Regel, sondern rechnet von links nach rechts.

**Lösung 3.3:**
Die BS1 zählt von oben wieder in den Zahlbereich hinein: -1 = 255, -2= 254... Ebenso zählt sie bei Werten über 255 wieder von unten: 256 = 0, 257 = 1...

**Lösung 3.4:**
Die BS1 rundet nicht, sie schneidet einfach ab.

**Lösung 3.5:**
a) Ein Mikrocontroller mit seinen Hilfsbauteilen.
b) Ports sind Pins, die als Signaleingänge oder Ausgänge genutzt werden können.
c) 50mA, der Spannungsregler kann 5V nur bis zu dieser Stromstärke erzeugen - sonst überhitzt er und schaltet ab.
d) BS1 steht für Basic Stamp 1, das kleinste embedded system einer ganzen Baureihe. BSE steht für den Basic Stamp Editor, die Entwicklungsumgebung für alle diese Systeme.
e) DEBUG, PAUSE, END

## Variablen

Den Schülern wird hier nur der Umgang mit Bytevariablen erläutert, der Umgang mit Bitvariablen wird erwähnt. Die BS1 kann auch mit Wordvariablen (W0 bis W6) umgehen, die dann einen Zahlbereich von 0 bis 65535 haben. Der Einsatz einer dieser Variablen belegt aber den Speicherplatz von jeweils zwei Bytevariablen: Wird W0 eingesetzt, kann man B0 und B1 nicht mehr einsetzen, W1 ersetzt B2 und B3...

Ein gutes Beispiel zur Diskussion des Überlaufs ist:

```
B2 = 15 - 30 * 2
B3 = 15 - 30 * 2 + 200
```

Ohne Punkt-vor-Strich-Rechnung erwartet man für B2 ein Ergebnis von -30, für die BS1 also 226. Für B3 erwartet man 170, was sich auch tatsächlich einstellt, da 226+200 in der BS1 wieder 170 ergibt.



Tipp:
Verzichtet man auf die Verwendung der Variablen B0 und B1 stehen 16 sogenannte **Bit-Variablen** mit den Namen BIT0 bis BIT15 zur Verfügung. Bit-Variablen können nur die beiden Werte 0 oder 1 speichern. Beispiel:

```
BIT3 = 1
BIT5 = BIT3
DEBUG BIT5
```

Feinheit:
Während man mit DEBUG B2 auf dem Computer zum Beispiel B2=0 dargestellt bekommt, erhält man mit DEBUG #B2 nur einfach die Zahl 0.

Hinweis:
Die Variablen B0 bis B11 haben nichts mit den Ports P0 bis P7 zu tun!

Korrektur:
Der erste Satz auf dieser Seite ist natürlich falsch: Es ist nicht so, dass ein Mikrocontroller einen Taschenrechner enthält, sondern dass ein Taschenrechner einen Mikrocontroller enthält. Und zwar einen, der mit Kommazahlen umgehen kann.

## Rechnen im Mikrocontroller

Man kann sagen, dass ein Mikrocontroller einen eingebauten Taschenrechner hat. Dieser Rechner hat 12 Speicher mit den Namen B0 bis B11. Man nennt diese auch **Variablen**.

Um zu sehen, welcher Wert gerade in einer Variable gespeichert ist, schreibt man die Anweisung

```
DEBUG B3
```

(ohne Anführungszeichen um B3). Auf dem Bildschirm des angeschlossenen Computers erscheint dann zum Beispiel

```
B3 = 0
```

wenn B3 gerade den Wert 0 enthält.

Um einen Zahlenwert in einer Variable zu speichern, schreibt man

```
B5 =
```

und dann den Zahlenwert.
An Stelle des Zahlenwerts kann auch eine Rechnung stehen - allerdings nur aus den Grundrechenarten -, +, * und /. Wenn ihr das folgende Programmbeispiel betrachtet, versteht ihr sofort, wie man mit den Rechnungen und den Variablen umgeht:

```
' {$STAMP BS1}
B2 = 1
B3 = 5 + 3 + 4
DEBUG B2
DEBUG B3
B2 = B3 * 2
B3 = B3 * 3
DEBUG B3
```

Leider kann die BS1 keine Kommazahlen verarbeiten - und auch keine besonders hohen Zahlen. Jede der Variablen kann nur die Werte von 0 bis 255 speichern - ab dann beginnt sie neu zu zählen. Solche Variablen nennt man **Byte-Variablen**.

### Aufgabe 3.1

Welchen Wert werden die Variablen B2 und B3 am Ende des Programmbeispiels (links unten) haben? Denkt erst nach und probiert es dann aus.

### Aufgabe 3.2

Die Rechnungen in einer Zeile können auch etwas komplizierter aussehen, z.B.

```
B2 = B1 * 2 + 16 / 2
```

Findet heraus, ob die BS1 die Punkt-vor-Strich-Regel beherrscht.

### Aufgabe 3.3

Die BS1 kann nicht mit negativen Zahlen umgehen. Findet heraus, was sie macht, wenn bei einer Rechnung eigentlich etwas Negatives herauskommen müsste?

### Aufgabe 3.4

Rundet die Basic Stamp beim Dividieren richtig?

### 3.5 Klausurwissen

a) Was ist ein embedded system?

b) Was ist das besondere an Ports gegenüber normalen Pins?

c) Wie groß darf die Stromstärke aller Bauteile, die auf dem Basic Stamp Project Board mit 5V versorgt werden müssen, in der Summe maximal betragen? Warum?

d) Was bedeuten die Abkürzungen BS1 und BSE?

e) Legt in eurem Heft eine Seite an, auf der ihr die Basic-Anweisungen sammelt, die ihr hier kennen lernt. Welche drei waren das bislang - außer der Wertzuweisung mit =? Welche Besonderheiten solltet ihr euch am besten vor allem bei DEBUG dazu notieren?

## Vokabeln und Leuchtdioden

Die Schüler müssen sich die etwas kryptischen Bezeichnungen $V_{DD}$ und $V_{SS}$ gut merken. Sie stammen aus der Transistorlogik und haben keine für die Schüler nachvollziehbare Erklärung. Es empfiehlt sich, diese Ausdrücke im Unterricht zu automatisieren. Dazu kann man die Schüler mit dem Finden von Eselsbrücken beauftragen (z.B. „Dagobert Duck ist reich - also $V_{DD}$ positiv“).

Die Marginalie zum Wirkungsgrad von LEDs vergleicht diese nur mit Glühlampen. Der Wirkungsgrad weißer LEDs ist nicht höher als der von weißen Leuchtstoffröhren.

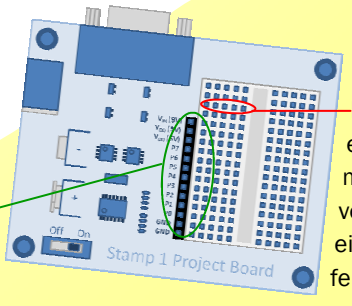
Der Spannungsregler versorgt auch die Betriebs-LED auf dem Project Board. Wenn diese trotz eingeschalteter Stromversorgung nicht leuchtet oder stark flackert, ist der Spannungsregler überlastet. Das deutet auf einen Kurzschluss zwischen $V_{DD}$ und $V_{SS}$ hin.

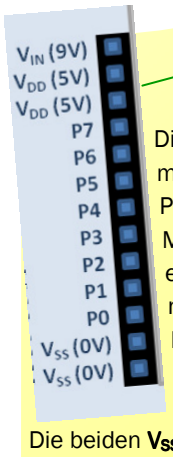
# Einführung in Mikrocontroller



## Leuchtdiode, Widerstand und Anschlussleiste

Hier lernt ihr die ersten elektronischen Bauteile sowie die Anschlussleiste und das Steckbrett näher kennen:



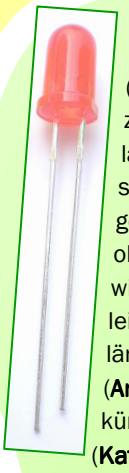

Die Anschlussleiste stellt euch bequemen Zugriff auf nützliche elektrische Potentiale sowie auf die Ports des Mikrocontrollers zur Verfügung. Merkt euch die Bezeichnungen der einzelnen Potentiale gut - ihr werdet sie oft brauchen: **$V_{IN}$** ist mit dem Pluspol der Batterie (oder des Netzteils) verbunden, hat also ein Potential von 9 V. Die beiden **$V_{SS}$**-Buchsen sind mit dem Minuspol der Batterie verbunden, haben also ein Potential von 0V.

Die beiden Buchsen **$V_{DD}$** haben ein Potential von 5V, das vom bereits erwähnten Spannungsregler erzeugt wird - die maximale Stromstärke die aus $V_{DD}$ fließen kann ist also recht begrenzt.

Die Buchsen **P0 bis P7** sind mit den acht Ports des Mikrocontrollers verbunden, die ihr auf den nächsten Seiten näher kennen lernt.

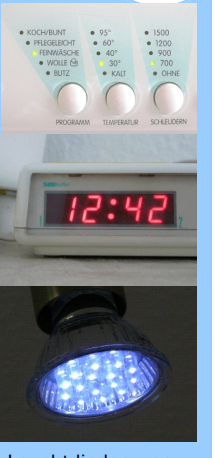
Auf dem Steckbrett sind jeweils die 5 Buchsen jeder Reihe im Inneren miteinander elektrisch verbunden. Möchte man zwei Kabel miteinander verbinden, steckt man also das eine in eine Buchse einer Fünferreihe und das andere in eine andere Buchse der gleichen Fünferreihe.

Ein häufiger Fehler ist, dass man ein Kabel oder ein Bauteil nicht fest genug einsteckt. Dann hat das Bauteil keinen Kontakt zu seiner „Fünferreihe“.



Eine **Leuchtdiode** (**LED**) könnt ihr euch zunächst als eine Glühlampe vorstellen, die sehr schnell kaputt geht, wenn man sie ohne schützenden Vorwiderstand einsetzt. Sie leitet Strom nur vom längeren Anschlussdraht (**Anode** genannt) zum kürzeren Anschlussdraht (**Kathode**) - in technischer Stromrichtung gedacht. Einfache Leuchtdioden sind für eine Stromstärke von 15 bis 17mA gedacht. Diese Stromstärke wird mit Hilfe eines Widerstands eingestellt.

Das Wort **Widerstand** kann in der Elektronik zweierlei bedeuten: Einmal die physikalische Größe Widerstand, abgekürzt mit dem Buchstaben R, gemessen in der Einheit Ohm. Und zum anderen ein Bauteil, das nichts weiter tut, als einen genau festgelegten elektrischen Widerstand zu haben, der durch die farbigen Ringe angegeben ist. Man setzt so einen Widerstand z.B. ein, um die Stromstärke gezielt zu reduzieren.




Leuchtdioden werden an vielen Stellen und in vielen Bauformen und Farben eingesetzt. Auch die leuchtenden Ziffern von Radioweckern sind oft aus (eckigen) Leuchtdioden zusammen gesetzt. Zunehmend ersetzen Leuchtdioden auch herkömmliche Glühlampen, weil sie einen höheren Wirkungsgrad haben, also aus einem Joule elektrischer Energie mehr Licht machen, als andere Lichtquellen.
Die Abkürzung LED kommt von der englischen Bezeichnung der Leuchtiode: Light Emitting Diode, wörtlich übersetzt: Licht aussendende Diode. Es gibt also auch Dioden, die kein Licht aussenden.

## Didaktik

Die Didaktik folgt hier sehr dem Prinzip „Lernen durch Handeln“ und geht dabei curricular vor: Die Schüler begreifen die Funktionsweise erst allmählich; weniger durch möglichst akkurate Erklärungen, als vielmehr durch das Nachahmen, Ausprobieren und Diskutieren mit dem Partner. Insbesondere der rechnerische Teil wird beim eigenständigen Erarbeiten gerne etwas unter den Tisch fallen gelassen, weshalb es sich lohnt, die Lösungen an der Tafel zu besprechen und auch weitere Beispiele zu diskutieren.

Die sehr anwendungsbezogene Grundregel ist: Spannungen in einem Stromkreis addieren sich, die Vorwärtsspannungen von Bauteilen sind hier als negative Spannungen zu werten. Zeigen Sie Ihren Schülern also Beispiele, in denen z.B. auch mehrere Leuchtdioden in Reihe auftreten.

Auch für die Anschlüsse der Leuchtiode wird oft eine Eselsbrücke benötigt, um Unsicherheiten auszuräumen (z.B. **An**ode - l**ang** - „Stromein**gang**“).

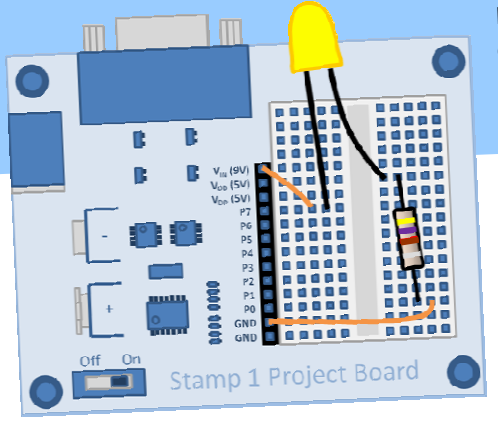
# 5 Einführung in Mikrocontroller

## Leuchtdiode leuchten lassen

Auf dieser Seite baut ihr nun eure erste Schaltung auf dem Steckbrett auf - eine sehr einfache Schaltung, für die man den Mikrocontroller noch gar nicht braucht. Ihr werdet nur die Potentiale benutzen, die auf der Anschlussleiste zur Verfügung gestellt werden.

**Aufgabe 5.1:**

Baut aus einer Leuchtdiode (kurz LED genannt, egal welcher Farbe), einem 470-Ohm-Widerstand (er hat die Farbkombination Gelb - Lila - Braun - Pause - Silber) und ein paar Drähten die abgebildete Schaltung auf. Während des Ein- und Umsteckens schaltet man die Mikrocontrollerplatine üblicherweise ab. Erst wenn alles fertig gesteckt ist, schaltet man sie wieder ein.




Denkt daran, die Leuchtdiode richtig herum einzusetzen (Anode ans höhere Potential) - sonst kann sie von Strom nicht durchflossen werden und auch nicht leuchten.

Die Leuchtdiode, die ihr verwendet, ist eine Standardleuchtdiode, die man idealerweise mit einer Stromstärke von 15 bis 17 mA betreibt und die einen eigenen Widerstand von fast 0 Ohm hat - also so gut wie keinen Widerstand. Wenn ihr nun mit dem Ohmschen Gesetz die Stromstärke I berechnet, die in eurer Schaltung fließt, werdet ihr feststellen, dass der 470 Ohm-Widerstand nicht genau passt:

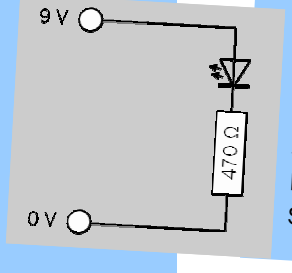
$$I = \frac{U}{R} = \frac{9V}{470\,\Omega} = 0{,}019\,A = 19\,mA$$

Trotzdem wird euch jeder Elektroniker sagen, dass der Widerstand richtig gewählt ist. Der Grund liegt darin, dass eine Leuchtdiode in ihrem Inneren eine Spannung von etwa 1,5V aufhebt. Man sagt auch: Eine LED hat eine **Vorwärtsspannung** von 1,5 V. Um diesen Betrag werden die anliegenden 9V also reduziert. Dann ergibt sich ein guter Wert:

$$I = \frac{U}{R} = \frac{9V - 1{,}5V}{470\Omega} = \frac{7{,}5V}{470\Omega} = 0{,}016A = 16mA$$

**Aufgabe 5.2:**

Techniker und Ingenieure würden die Schaltung als **Schaltbild** darstellen. In einem Schaltbild hat jedes Bauteil ein (üblicherweise europa- oder sogar weltweit) festgelegtes **Schaltsymbol**. Legt in euren Heften eine Doppelseite an, auf der ihr nach und nach jedes elektrische Bauteil mit Namen, Abkürzung und vor allem seinem Schaltsymbol auflistet. Die ersten beiden Schaltsymbole (für LED und Widerstand) könnt ihr herausfinden, in dem ihr dieses Schaltbild mit eurer Schaltung vergleicht.



# Einführung in Mikrocontroller

6

## Widerstandssortiment

Obwohl in der Praxis nur recht wenige Widerstandswerte für die Arbeit mit dem Mikrocontroller benötigt werden, ist es doch pädagogisch wertvoll, ein Sortiment parat zu haben. Die für die Arbeit wichtigen Widerstandswerte, die also in größerer Stückzahl vorhanden sein sollten, sind 220 Ohm, 470 Ohm sowie 10 und 20 kOhm.

Die Widerstände sind eigentlich kleine Heizungen, denn sie geben die elektrische Leistung P=U I als Wärme an die Umgebung ab. Daher gibt es Widerstände in verschiedenen Leistungsklassen. Die hier vorgestellten haben eine Leistung von 0,25W, Widerstände mit höheren Leistungen haben größere, thermisch günstiger geformte und vor allem temperaturstabilere Gehäuse, z.B. aus Keramik.

Die Zahlencodes kann man sich recht gut merken: Erst die dunklen (schwarz und braun), dann den Regenbogen entlang (rot bis lila), und dann die hellen (grau und weiß) Farben. Auf die Beschriftungsfarben für Widerstände kleiner als 10 Ohm wurde hier verzichtet (Kurzreferat?)

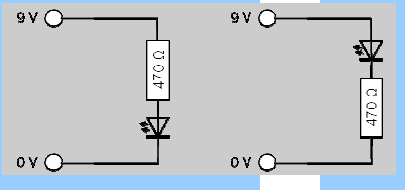
**Lösung 6.1:**
Es spielt natürlich keine Rolle.

**Lösung 6.2:**
a) 46200 Ohm bei einer Toleranz von 0,25%.
b) Orange-Grün-Braun-Lücke-Gold

**Lösung 6.3:**
220 Ohm, also Rot-Rot-Braun.

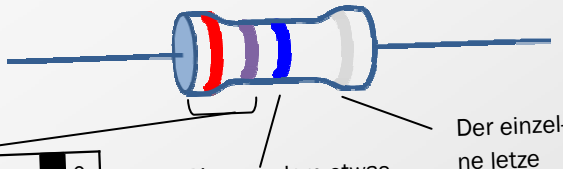
### Aufgabe 6.1:

Den Widerstand, den man beim Einsatz einer LED immer benötigt, bezeichnet man als **Vorwiderstand** der LED. Spielt es eine Rolle, ob man den Widerstand „vor“ der Anode oder „nach“ der Kathode in den Stromkreis einsetzt? Probiert es aus, wenn ihr euch nicht sicher seid.




## Farbcodes von Widerständen

Der Farbcode auf Widerständen besteht aus vier oder fünf Ringen. Damit man weiß, wo man mit dem Ablesen anfangen muss, hat der letzte Ring einen etwas größeren Abstand von den anderen.



Die ersten zwei oder drei Ringe geben jeweils Ziffern an. Jeder Ring steht für eine Ziffer der Zahl gemäß diesem Farbcode. Hier also: 27

| | |
|---|---|
| Schwarz | 0 |
| Braun | 1 |
| Rot | 2 |
| Orange | 3 |
| Gelb | 4 |
| Grün | 5 |
| Blau | 6 |
| Lila | 7 |
| Grau | 8 |
| Weiß | 9 |

Der Ring vor dem etwas größeren Abstand gibt an, wie viele Nullen angefügt werden müssen. Die Anzahl ergibt sich ebenfalls aus der linken Farbtabelle.

Hier müssen 6 Nullen angehängt werden. Der Widerstand hat also einen Wert von 27000000 Ω, meist abgekürzt als 27 MΩ (Mega-Ohm). Das ist ziemlich viel.

Der einzelne letzte Ring gibt an, wie genau der Widerstand hergestellt ist. Je genauer, desto teurer ist ein Widerstand auch.

| | |
|---|---|
| Grau | 0,05 % |
| Lila | 0,1 % |
| Blau | 0,25 % |
| Grün | 0,5 % |
| Braun | 1 % |
| Rot | 2 % |
| Gold | 5 % |
| Silber | 10 % |
| keiner | 20 % |

Dieser Widerstand ist auf 10% genau gefertigt.

Günstige Widerstände haben **Toleranzen** (Genauigkeiten) von 10% oder 20%. Es gibt also keinen Sinn, Widerstände mit nah beieinanderliegenden Werten zu produzieren, wenn sie ohnehin so ungenau sind. Daher sind günstige Widerstände nur in folgenden Werten üblich:

| | | | |
|---|---|---|---|
| 10 | 100 | 1000 | ... |
| 12 | 120 | 1200 | ... |
| 15 | 150 | 1500 | ... |
| 18 | 180 | 1800 | ... |
| 22 | 220 | 2200 | ... |
| 27 | 270 | 2700 | ... |
| 33 | 330 | 3300 | ... |
| 39 | 390 | 3900 | ... |
| 47 | 470 | 4700 | ... |
| 56 | 560 | 5600 | ... |
| 68 | 680 | 6800 | ... |
| 82 | 820 | 8200 | ... |

Hinweis:
Die Widerstands-Farbcodes braucht man nicht auswendig zu wissen. Bei Bedarf schlägt man sie nach - oder misst den Widerstand eines Widerstands mit dem Multimeter.

### Aufgabe 6.2:

a) Welchen Wert hat dieser Widerstand?

b) Welchen Farbcode hätte ein Widerstand mit 350 Ω und einer Toleranz von 5%?

### Aufgabe 6.3:

Die Leuchtdiode aus Aufgabe 5.1 soll nun an einer Spannung von 5V, also zwischen $V_{DD}$ und $V_{SS}$ betrieben werden. Berechnet den passenden Vorwiderstandswert, ermittelt die Farbkombination und baut die Schaltung auf.

## Falsch verstandene Pause

Unter den Schülern befinden sich mit hoher Regelmäßigkeit welche, die hier zunächst nicht verstehen, dass Anweisungen nacheinander abgearbeitet werden und die Zustände der PIN-Variablen sich nur dann ändern, wenn es dazu eine Anweisung gibt. Sie glauben, dass die Anweisungen PIN6=1 und PAUSE 500 irgendwie zusammen gehören und der Port nur 500 Millisekunden lang high bleibt, weil der Pausenbefehl direkt dahinter steht. Es lohnt sich, Beispiele an der Tafel zu diskutieren und Laufzeiten von Programmen abzuschätzen:

Die meisten Anweisungen dauern etwa 1/2000 s, ein DEBUG etwas länger und eine PAUSE je nach Wert des Parameters natürlich wesentlich länger. Auch PIN1=1 ist nach 1/2000s abgearbeitet, der Zustand der Variable PIN1 und des Ports bleibt allerdings dauerhaft gespeichert.

Bei der Betrachtung der kleineren SMD-Bauteile stellt sich die Frage, wie groß Widerstände wirklich sind. Hier kann man die Leistungswerte von Widerständen thematisieren.

# 7 Einführung in Mikrocontroller

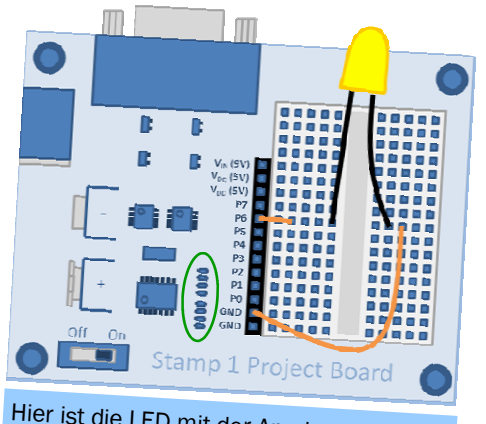
## Blinkende Leuchtdiode

Jetzt kommt der Mikrocontroller zum Einsatz, um eine LED blinken zu lassen. Dazu wird sie mit ihrer Anode (längerer Anschlussdraht) nicht mehr an $V_{IN}$ oder $V_{DD}$ angeschlossen, sondern an einen der Ports des Mikrocontrollers. Die Kathode bleibt an $V_{SS}$, also an den Minuspol der Batterie, angeschlossen.

Der Mikrocontroller muss dann so programmiert werden, dass der Port abwechselnd ein Potential von 5V und ein Potential von 0V annimmt. Immer wenn das Potential 5V beträgt, wird ein Strom durch die Leuchtdiode fließen; immer wenn es 0V beträgt, existiert keine Potentialdifferenz zwischen Anode und Kathode und es fließt kein Strom.

Wenn ein Port das hohe Potential (5V) hat, sagt man, der Port sei **high**. Wenn er das niedrige Potential hat (0V), so ist er **low**.

Das folgende Beispiel zeigt ein Programm, das eine Leuchtdiode blinken lässt, die an Port 6 angeschlossen wurde. Es ist nicht schwer verständlich, wenn ihr die Erläuterungen lest:




Hier ist die LED mit der Anode an Port 6 angeschlossen. Der Port wird dann abwechselnd high (5V) und low (0V), so dass die LED blinkt.

Auf einen Vorwiderstand kann man dabei verzichten, weil auf der Platine bereits vor jeder Anschlussbuchse ein passender Widerstand mit 220 Ohm eingebaut ist. Die Widerstände sind jeweils zu viert in einem kleinen, schwarzen sog. **SMD**-Gehäuse untergebracht (grüne Ellipse).

```
' {$STAMP BS1}
OUTPUT 6

PIN6 = 1
PAUSE 500
PIN6 = 0
PAUSE 500
PIN6 = 1
PAUSE 500
PIN6 = 0
PAUSE 500
END
```

Es ist eine Besonderheit der BS1, dass jeder der Ports des Mikrocontrollers als Ausgang oder als Eingang für elektrische Signale verwendet werden kann. Hier soll Port 6 als Ausgang fungieren. Die Anweisung OUTPUT 6 sorgt dafür, dass Port 6 ein Ausgang ist.

Für jeden der Ports gibt es eine Bit-Variable (kann nur 0 oder 1 speichern), die festlegt, ob der Port high (5V) oder low (0V) ist. Für die acht Ports P0 bis P7 heißen diese Variablen PIN0 bis PIN7. Wenn eine Variable den Wert 1 hat, ist der Port high (5V) und bleibt so, bis sich der Wert der Variablen ändert. Ist der Wert der Variablen 0, ist der Port low (0V). Hier soll Port 6 high werden - also wird der Variable PIN6 der Wert 1 zugewiesen.

Hier soll Port 6 low werden, damit die LED ausgeht. Deshalb wird der Variable PIN6 der Wert 0 zugewiesen.

**Warum steht hinter PIN6 ein Gleichheitszeichen und hinter Pause nicht?**
PIN6 ist eine Variable, so wie B0, B1 und die anderen auch. Um einen Port high oder low zu schalten, muss man der Variable einen neuen Wert geben. Dazu benötigt man das Gleichheitszeichen, wie wenn man B2=2+5 schreibt.

PAUSE hingegen ist eine Anweisung. Sie kann keinen Wert speichern, man muss ihr aber dennoch sagen, wie lang die Pause sein soll. Das schreibt man ohne Gleichheitszeichen, so wie man bei DEBUG den zu sendenden Text auch ohne Gleichheitszeichen angibt.

**SMD?**
SMD (surface mounted device) ist eine sehr kompakte Gehäuseform für elektronische Bauelemente, die insbesondere von Robotern sehr schnell auf Platinen aufgelötet werden kann. Für eigene Basteleien sind diese Gehäuse unpraktisch.

## Endlosschleife

In Beispielen an der Tafel, auch wenn Schüler sie dort notieren, sollte auf das Einrücken hingearbeitet werden. Ob das GOTO ebenfalls eingerückt werden soll oder nicht, ist eine Geschmacksfrage. Größere Programme werden aber mit eingerücktem GOTO (und später dann RETURN) sehr viel übersichtlicher.

Für Aufgabe 8.6 kann man auch kleine Ampel-LED-Bausteine besorgen, mit denen das noch mehr Spaß macht (z.B. Reichelt MEN 1881.8720 für ca. 1,25€). Teilaufgabe für eine schöne GFS ist es, ein Kreuzungsmodell zu fertigen.

**Lösung 8.1:**
ähnlich wie das Beispiel auf Seite 7.

**Lösung 8.2:**
Es sind dies die vier Pins oben rechts und die vier Pins unten rechts. Man erkennt es auf der Platine und begreift dabei die in Reihe geschalteten Widerstände. Dazu ist die Aufgabe da. Allerdings sind zwei Verbindungen auf die Rückseite durchkontaktiert.

**Lösung 8.4:**
Es können die vier Zeilen über dem GOTO und das END gestrichen werden.

**Lösung 8.5:**
Die Ausschaltzeit der LED zwischen LOW3 und HIGH3 ist so kurz, dass man sie nicht wahrnimmt.

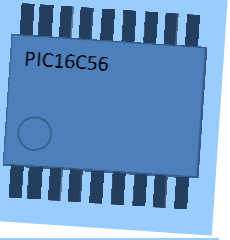


**Aufgabe 8.1:**

Schließt eine LED an Port 5 an und schreibt ein Programm, das diese LED im Sekundenrhythmus viermal blinken lässt. Gerne könnt ihr das auch variieren.

**Aufgabe 8.2:**

Der eigentliche Mikrocontroller ist ja eines der schwarzen SMD-Bauteile auf der Platine. Er hat 20 Pins, von denen acht die Ports sind. Könnt ihr auf der Platine erkennen, welche acht das sind?




## Endloses Blinken

Um eine Leuchtdiode häufiger blinken zu lassen, könnte man das high- und low-Schalten des Ports immer wieder hintereinander in das Programm schreiben. Es gibt aber eine bessere Lösung: Die sogenannte **Endlosschleife**.

Sie besteht aus einem sogenannten „unbedingten Sprung“ mit Hilfe der Anweisung GOTO. Dazu sind zwei Schritte notwendig, die das folgende Programmbeispiel erklärt:

```
' {$STAMP BS1}
OUTPUT 6

Otto:
  PIN6 = 1
  PAUSE 500
  PIN6 = 0
  PAUSE 500
  PIN6 = 1
  PAUSE 500
  PIN6 = 0
  PAUSE 500
  GOTO Otto

END
```

1. Eine Zeile des Programms wird mit einem sogenannten Label markiert. Das ist ein beliebiges Wort (ohne Umlaute, Sonderzeichen und Leerzeichen), das mit einem Doppelpunkt endet.

2. Wenn der Mikrocontroller bei der Bearbeitung des Programms auf die Anweisung GOTO stößt, setzt er die weitere Abarbeitung am angegebenen Label fort.

**Aufgabe 8.3:**

Realisiert ein Dauerblinklicht und - anschließend daran - ein Dauerblitzlicht (LED immer nur kurz an).

Probiert auch aus, die Anweisungen PIN4=1 und PIN4=0 durch HIGH 4 und LOW 4 zu ersetzen. Das geht auch - und ist manchmal einfacher zu lesen.

**Aufgabe 8.4:**

Das Programmbeispiel links lässt sich um fünf Anweisungen kürzen, ohne dass sich etwas am Ergebnis ändert. Welche?

**Aufgabe 8.5:**

Mit diesem Programm blinkt die an Port 3 angeschlossene LED nicht. Warum?

```
' {$STAMP BS1}
OUTPUT 3
Anfang:
  HIGH 3
  PAUSE 500
  LOW 3
  GOTO Anfang
```

**Aufgabe 8.6:**

Besorgt euch eine grüne, eine gelbe und eine rote LED und programmiert eine Ampel: grün, gelb, rot, rot+gelb, grün...und so weiter.

Tipp:
Erfahrene Programmierer wissen, dass eine Übersichtlichkeit des Programmcodes sehr dabei hilft, Fehler zu vermeiden.

Deshalb werden üblicherweise die Programmzeilen, die innerhalb eines mehrfach wiederholten Programmteiles liegen, um zwei Leerzeichen eingerückt.

So seht ihr es auch im Programmbeispiel links unten auf dieser Seite. Und selbst solltet ihr es auch so machen.

„unbedingter Sprung“
Im Text heißt es, GOTO sei die Anweisung für einen „unbedingten Sprung“. „unbedingt“ bedeutet: Bei GOTO wird das Programm ohne jede weitere Bedingung beim Label fortgesetzt.

Es gibt auch einen Sprungbefehl, der mit einer Bedingung verknüpft ist. Er heißt IF...THEN - ihr lernt ihn später in diesem Heft kennen.

**Lösung 91:**
B1=1
B1=2
...
B1=20.

**Lösung 9.2:**
die Fünferreihe von 5 bis 50.

**Lösung 9.3:**
Das Programm erzeugt ein Lauf-Blitzlicht durch die 8 LEDs. Dies ist der entscheidende Unterschied zwischen den HIGH/LOW-Anweisungen und der Schreibweise PIN1=... Mit dieser hätte man das Lauflicht nicht so elegant programmieren können.

## For-Next-Problem

Während einige Schülergruppen die FOR-NEXT-Schleife schnell kapieren, gibt es auch immer Gruppen, für die das zu kompliziert ist. Tafelbeispiele, die sich dann auch auf die FOR ... TO ...STEP-Erweiterung mit negativem STEP oder Verschachtelungen ausweiten sind für niemanden langweilig und helfen den schwächeren Gruppen.

**Beispiel:**

```
' {$STAMP BS1}
FOR B1=100 TO 80 STEP -2
  DEBUG B1
  NEXT B1
```

gibt die Zahlen 100,98,96,94... bis 84,82 und 80 aus.

Für Aufgabe 9.3 sollte man eine größere Zahl gelber Leuchtdioden bereithalten. Wenn Schüler das Programm variieren, werden sie schnell feststellen, dass der Spannungsregler beim gleichzeitigen Einschalten von zu vielen LEDs abschaltet. Dies äußert sich in einem unlogischen und sehr unzuverlässigen Verhalten der BS1.

9

# Einführung in Mikrocontroller

## Die For-Next-Schleife

Auf der vorigen Seite habt ihr mit der GOTO-Anweisung gelernt, wie man sogenannte Endlosschleifen programmiert. Manchmal ist es aber auch günstig, etwas nur exakt 20 oder 30 Mal ausführen zu lassen, bevor das Programm weiter ablaufen soll. Dazu dienen sogenannte Zählschleifen, die in einer Variable mitzählen, wie oft sie schon durchlaufen wurden. In der BS1 heißt die Zählschleife FOR-NEXT-Schleife:

```
' {$STAMP BS1}
OUTPUT 2

FOR B3=5 TO 25
  PIN2=1
  PAUSE 500
  PIN2=0
  PAUSE 500
  NEXT B3

DEBUG "Fertig"
END
```

In dieser FOR-Zeile steht, dass die Variable B3 am Anfang auf den Wert 5 gesetzt werden soll. Außerdem steht hier, dass B3 höchstens 25 werden darf.

Diese Zeilen werden in diesem Beispiel solange wiederholt, bis B3 einen Wert von 25 überschreitet.

Die Anweisung NEXT sorgt dafür, dass nun die nächste Wiederholung der Schleife dran ist. Davor wird aber der Wert von B3 um 1 erhöht.
Wenn das nicht mehr möglich ist, weil B3 bereits den Höchstwert von 25 erreicht hat, setzt der Mikrocontroller das Programm einfach mit der nächsten Anweisung fort. In diesem Fall wird dann das Wort „Fertig“ an den Computer gesendet.

Tipp:
In der FOR-Zeile kann mit dem Wort STEP angegeben werden, wie viel der Zählvariable bei jeder Runde hinzuzuzählen ist:

Beispielsweise zählt FOR B2=0 TO 10 STEP 2 in Zweierschritten nach oben. FOR B2=10 TO 0 STEP -1 zählt rückwärts und verlässt die Schleife, sobald die 0 unterschritten wird.

### Aufgabe 9.1:

Was wird die folgende For-Next-Schleife in einem Programm machen? Probiert es ggf. aus!

```
' {$STAMP BS1}
FOR B1=1 TO 20
  DEBUG B1
  NEXT B1
```

### Aufgabe 9.2:

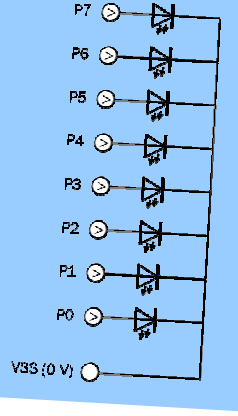
Was zeigt dieses Programm an?

```
' {$STAMP BS1}
FOR B2=1 TO 10
  B3=B2*5
  DEBUG CR, #B3
  NEXT B2
END
```

Tipp:
Die Angabe der Variable hinter dem Wort NEXT ist - wie man so schön sagt - optional. Das heißt: Man darf die Variable auch weglassen, weil sie im FOR ja bereits angegeben ist.

### Aufgabe 9.3:

Was macht das folgende Programm, wenn man an jeden der acht Ports eine LED anschließt? Überlegt erst - und probiert es dann.




```
' {$STAMP BS1}
OUTPUT 0
OUTPUT 1
OUTPUT 2
OUTPUT 3
OUTPUT 4
OUTPUT 5
OUTPUT 6
OUTPUT 7

Start:
  FOR B1=0 TO 7
    HIGH B1
    PAUSE 100
    LOW B1
    NEXT B1
GOTO Start
```

## Lautsprecher

Die Erklärung des Aufbaus des Lautsprechers ist nicht ganz präzise. Von den Schülern können jedoch noch keine vertieften Kenntnisse über den Elektromagnetismus erwartet werden. Auch auf die Akustik und andere Bauformen von Lautsprechern wird hier nicht näher eingegangen.

Hinweis: Professionell würde man den Lautsprecher noch über einen Kondensator puffern (vgl. BSE Hilfe „Sound“). Da es in der Praxis auch ohne diesen Kondensator ordentlich funktioniert und die Schüler ihn kaum verstehen könnten, wird auf hierauf verzichtet.

Der Sound-Befehl hat eine Erweiterung: Man kann ganze Melodien in einer Zeile ausdrücken, in dem man weitere Töne jeweils durch Komma getrennt anfügt: SOUND 1, (Tonhöhe, Tondauer, Tonhöhe, Tondauer, Tonhöhe, Tondauer). Auf diese Erweiterung wird im Schülertext verzichtet, da das einzelne Niederschreiben der Sound-Befehle das Gefühl für die zeilenweise Struktur eines Programmes verstärkt.

**Lösung 10.2:**
255 x 12 ms = 3,06 s.

**Lösung 10.3:**

```
' {$STAMP BS1}
OUTPUT 3
Start:
 FOR B1=50 TO 100 STEP 5
  SOUND 3,(B1,20)
 NEXT B1
 FOR B1=100 TO 50 STEP -5
  SOUND 3,(B1,20)
 NEXT B1

 GOTO Start
```

## Töne

Auf dieser Seite lernt ihr, wie man mit dem Mikrocontroller Töne erzeugen kann. Dazu benötigt ihr lediglich noch einen Lautsprecher.

Ein **Lautsprecher** ist im wesentlichen ein Elektromagnet (1) mit einer Membran (2). Immer wenn ein Strom durch den Elektromagnet fließt, zieht er die Membran nach hinten. Fließt kein Strom, schwingt die Membran wieder nach vorne.

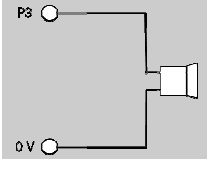

**Aufgabe 10.1:**

Schließt den Lautsprecher an Port 3 und an $V_{SS}$ (GND) an. Schreibt ein Programm, das „Alle meine Entchen“ oder eine andere bekannte Melodie vorspielt.

Der Lautsprecher wird nur sehr leise klingen - dies liegt u.a. am internen Vorwiderstand auf der Platine.

Geschieht beides abwechselnd mit einer hinreichend hohen Frequenz, erklingt ein für den Menschen hörbarer Ton. Um einen Ton zu erzeugen kann man also den Lautsprecher an einen Port und an $V_{SS}$ (0V) anschließen und ein Programm wie das folgende starten:

```
' {$STAMP BS1}
OUTPUT 3
Mozart:
  HIGH 3
  LOW 3
  GOTO Mozart
```




Es geht aber auch einfacher, weil es eine spezielle BASIC-Anweisung zur Tonerzeugung gibt:

```
SOUND 3, (101,50)
```

Diese Anweisung erzeugt ein schnell wechselndes Signal auf Port 3. Die Zahl 50 gibt die Dauer des Tones in einer Einheit von 12ms an. Erst wenn der Ton fertig ist, kann die nächste Anweisung des Programms ausgeführt werden.

Die Zahl 101 ist ein Code für die Tonhöhe. Man kann hier Zahlen von 0 bis 255 einsetzen. Von 0 bis 127 stehen sie für gewöhnliche Töne, dann folgen verschiedene Geräusche.

In einer kurzen Programmieranleitung würde die Sound-Anweisung einfach so dargestellt sein:

```
SOUND Port, (Tonhöhe, Tondauer)
```

**Aufgabe 10.2:**

Wie lang dauert der längste Ton, der sich mit einer einzigen Sound-Anweisung erzeugen lässt?

**Aufgabe 10.3:**

Programmiert eine Sirene, also einen Ton, der beständig langsam höher und dann wieder tiefer wird.

Vermutlich benötigt ihr hierzu zwei For-Next-Schleifen mit dem STEP-Zusatz.

**Aufgabe 10.4:**

Ist eure Liste aller BASIC-Anweisungen und die Liste aller Bauteile und ihrer Schaltsymbole auf dem aktuellen Stand? Mindestens acht Anweisungen und drei Symbole...

| | | | |
|---|---|---|---|
| | | E | 110 |
| 109 | Dis | D | 108 |
| 106 | Cis | C | 105 |
| | | H | 104 |
| 102 | B | A | 101 |
| 99 | Gis | G | 97 |
| 96 | Fis | F | 94 |
| | | E | 92 |
| 89 | Dis | D | 87 |
| 85 | Cis | C | 82 |
| | | H | 79 |
| 76 | B | A | 73 |
| 70 | Gis | G | 67 |
| 63 | Fis | F | 59 |
| | | E | 53 |

Die Klaviatur zeigt die Tonhöhencodes für einige Töne.

**Software** und **Hardware**:
Jedes Gerät, das einen Mikrocontroller enthält, besteht im Prinzip aus zwei Teilen, in die jemand viel Grips investiert hat: Einmal ist das der Aufbau der Elektronik aus vielen Teilen, die sogenannte Hardware. Und zum anderen das Programm, ohne das es ja auch nicht funktionieren würde: die Software.

Hinweis:
Die Sound-Anweisung setzt einen Pin abwechselnd auf high und low. Am Ende der Anweisung bleibt der Pin auf high. Mit LOW kann man es wieder abschalten.

**Lösung 11.2:**
a) Light Emitting Diode
b) ca. 656 Ohm, man wählt also den nächst größeren Widerstand mit einem Wert von 680 Ohm.
d) high: in der Software eine 1, am Pin über 1,4V. low: in der Software eine 0, am Pin unter 1,4V.
e) 200 Sterne

## Potentiometer

In der Physiksammlung finden sich üblicherweise große Schiebepotentiometer, an denen der Aufbau gut verdeutlicht werden kann.

Es bietet sich an dieser Stelle an, das Ohmsche Gesetz und das Berechnen von Vorwiderständen zu wiederholen.

Hier wird das Potentiometer als regelbarer Widerstand verwendet—daher werden nur zwei der drei Anschlüsse benötigt. In der Praxis wird dann der dritte Anschluss mit dem Mittelabgriff verbunden - der Einfachheit halber wird darauf hier nicht eingegangen.



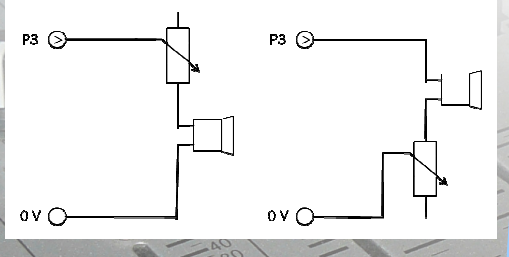
## Lautstärkeregelung mit einem Potentiometer

An vielen elektronischen Geräten findet man Regler, an denen man drehen oder schieben kann. Dahinter stecken meistens **Potentiometer**, kurz „**Poti**“ genannt. Das sind regelbare Widerstände mit zumeist drei Anschlüssen.

Üblicherweise ist der linke Anschluss an einem Ende einer Kohleschicht befestigt, der rechte Anschluss am anderen Ende. Der mittlere Anschluss, genannt **Mittelabgriff**, ist mit einem Kontakt verbunden, der auf der Kohleschicht schleift und beim Drehen oder Schieben verschoben wird.

Da die Kohlebahn relativ schlecht leitet, fließt ein Strom leichter durch ein kurzes dünnes Stück Kohle als durch ein langes Stück. Beim Drehen des Potentiometers wird also der elektrische Widerstand zwischen dem Mittelabgriff und jedem der anderen Anschlüsse verändert. Wenn man Potentiometer kauft, ist der Maximalwiderstand angegeben.

Ihr sollt hier ein Potentiometer verwenden, um die Lautstärke des ohnehin recht leisen Tons noch leiser und feiner einstellen zu können - bis der Ton schließlich unhörbar leise wird. Dazu muss die Stromstärke im Lautsprecher nur einfach weiter verringert werden - der zusätzliche veränderbare Widerstand muss also zwischen den Port und den Lautsprecher oder - das ist genau so gut - zwischen den Lautsprecher und $V_{SS}$ geschaltet werden:




### Aufgabe 11.1:

Baut in eure Sirene oder eine andere Tonausgabe eine Lautstärkeregelung mit einem Potentiometer ein. Geeignet ist ein Potentiometer mit einem Maximalwiderstand von 5000 bis 10000 Ω.

### 11.2 Klausurwissen:

a) Wofür stehen die Buchstaben LED? [S. 7]

b) Berechnet einen geeigneten Vorwiderstand, um eine Leuchtdiode an einer Spannung von 12 V zu betreiben. [S. 8]

c) Warum werden billige Widerstände nicht mit allen Werten hergestellt? [S. 9]

d) Was bedeuten die Begriffe high und low?

e) Wie viele Sterne werden hier ausgegeben?

```
' {$STAMP BS1}
FOR B1=1 TO 10
  FOR B2=1 TO 20
    DEBUG "*"
  NEXT B2
NEXT B1
```

f) Was sind Hardware und Software?

Die Leistungsberechnung ist stark vereinfacht, in Anbetracht der verwirrenden Leistungsangaben auf Multimediageräten (Schülerreferat?) aber spielt das kaum eine Rolle:

**Nennleistung** nach DIN 45324 ist die einzig wirklich seriöse Angabe: Hier wird bei einem Verstärker die elektrische Leistung bei der Abgabe einer großen Zahl verschiedener Frequenzen („rosa Rauschen“) bestimmt, bei Lautsprechern wird die elektrische Maximalleistung angegeben, bei der der Lautsprecher das „rosa Rauschen“ noch dauerhaft wiedergeben kann.

**Sinusleistung, RMS**: Wie die Nennleistung, aber mit einer einzigen Frequenz gemessen. Liegt etwa 1,5 Mal höher als die Nennleistung.

**Musikleistung, PMPO**: Eine nicht geschützte Bezeichnung, bei jeder einfach irgendetwas misst. Zitat: „Alleine das Vorhandensein einer PMPO-Angabe [ist] ein Merkmal für minderwertige Multimedia-Geräte.“ Die Nennleistung kann um das hundertfache oder tausendfache geringer sein.

**Schalldruckpegel**, z.B. 95dB: seriöse Lautstärkemessung, 1m vor dem Lautsprecher; gibt an, „wie viel Schall“ erzeugt wird und berücksichtigt den Wirkungsgrad des Lautsprechers.

**Lösung 12.1:**
a) 4,2 W bei 6 Ohm
b) 13,5 W bei 6 Ohm



## Lautstärkeverstärkung

Bevor ihr gleich einen sogenannten Transistor als Verstärker verwendet, um die Tonausgabe lauter zu machen, müsst ihr verstehen, warum die Tonausgabe denn überhaupt so leise ist:

Das liegt daran, dass die elektrische Leistung, mit der der Magnet im Lautsprecher die Membran anziehen kann, sehr gering ist. Für die elektrische Leistung gilt bekanntlich P=U·I. Die Spannung U beträgt beim Betrieb an einem Port 5V, die Stromstärke ergibt sich aus der Spannung sowie dem gesamten Widerstand in diesem Stromkreis. Er setzt sich aus 220 Ω auf der Platine und grob geschätzten 6 Ω im Lautsprecher zusammen. Es ist also die Stromstärke

$$I = \frac{U}{R} = \frac{5V}{220\Omega + 6\Omega} = 0{,}022A = 22mA$$

Daraus ergibt sich eine Leistung von

$$P = U \cdot I = 5V \cdot 0{,}022A = 0{,}11W$$

Das ist weniger als ein Zehntel dessen, wofür der Lautsprecher ausgelegt ist. Die Membran wird also nur schwach zurück gezogen, die an die Luft übertragene „Bewegung“ ist also gering.

Die Leistung wäre schon erheblich besser, wenn ihr den Lautsprecher an 5V Spannung ohne den 220 Ω-Widerstand betreiben könntet - und noch besser an den 9V zwischen $V_{IN}$ und $V_{SS}$. Allerdings ist $V_{IN}$ eben leider kein Port, lässt sich also nicht mit Hilfe eines Programms (per Software) schnell ein- und ausschalten. Aber genau dabei wird der Transistor helfen.

Leistungsangaben auf Lautsprechern:
Ihr berechnet hier die elektrische Leistung, mit der euer Lautsprecher arbeitet, wenn die Membran gerade zurückgezogen wird.
Für die Leistungsmessung von Lautsprechern existieren in der Praxis bessere Verfahren.

**Aufgabe 12.1:**

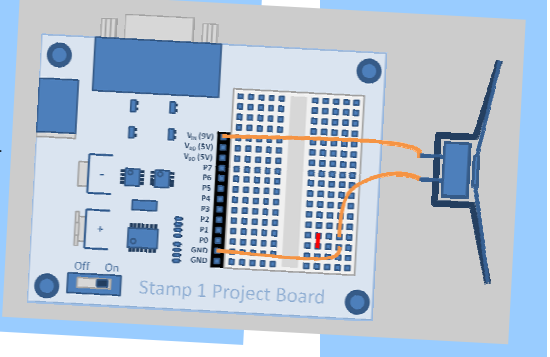
a) Berechnet die elektrische Leistung, die euer Lautsprecher bei direktem Betrieb an einer Spannung von 5V hätte.

b) Berechnet die Leistung beim direkten Betrieb , des Lautsprechers zwischen $V_{IN}$ und $V_{SS}$.

**Aufgabe 12.2:**

Mit einem kleinen Experiment kann man leicht feststellen, dass der Ton bei einem Betrieb an 9V viel lauter werden würde: Schließt den Lautsprecher wie abgebildet direkt zwischen $V_{IN}$ und $V_{SS}$ an. Wenn ihr nun das rote Kabelstück herauszieht und hereinsteckt, hört ihr ein Knacksen - viel lauter als der bisherige Ton.

Macht diesen Aufbau ruhig so umständlich, wie es die Abbildung zeigt: Das rote Kabelstück wird auf der kommenden Seite durch einen Transistor ersetzt, der diese „rote“ Verbindung mit hoher Frequenz trennen und wieder schließen wird - gesteuert von einem der Ports.



**Lösung 13.1:**
Stelle 1: Basis; Stelle 2: Collector; Stelle 3: Emitter

Interessant:
Das Experimentieren mit der Schaltung aus Aufgabe 14.1 ist spannend, wenn man dort den Widerstand um eine LED ergänzt. Sie leuchtet auch dann, wenn man nur eine Tischplatte oder eine Schülerkette zwischen $V_{IN}$ und die Basis von T1 bringt.

## Transistoren

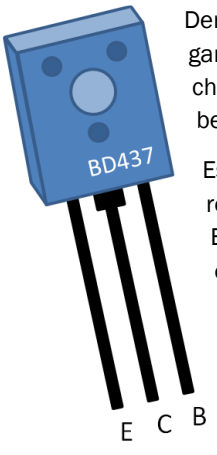
Die Schülerinnen und Schüler lernen hier nur NPN-Transistoren kennen, weil diese etwas leichter zu durchdenken sind. Der innere Aufbau von Transistoren ist typischer Lerninhalt des Faches Physik in der neunten oder zehnten Klasse, muss in NwT also nicht behandelt werden.

Der Transistor ist unter anderem deshalb die am häufigsten Hergestellte technische Einheit, weil moderne Mikroprozessoren bereits mehr als 1 Milliarde Transistoren beinhalten. Die Anzahl der Transistoren ist dabei eines der Kriterien für die Leistungsfähigkeit eines Mikrocontrollers oder Prozessors. Das sog. „Mooresche Gesetz“ (Schülerreferat?) - ausgesprochen bereits 1965 und korrigiert 1975 - besagt, dass sich in unserem digitalen Zeitalter die Anzahl der Transistoren auf einem Mikroprozessor etwa alle 18-24 Monate verdoppeln wird. Eine bislang sehr genaue Abschätzung. Eine historische Liste der Mikroprozessoren und ihrer Transistorenzahlen findet sich bei Wikipedia unter dem Stichwort „Mikroprozessor“. Der PIC-Mikrocontroller liegt im Bereich von ca. 100000 Transistoren.

# 13 Einführung in Mikrocontroller

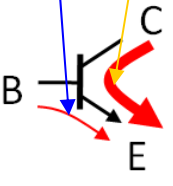
Rekord:
Der Transistor ist eines der wichtigsten Bauteile der Welt - und die mit Abstand am häufigsten hergestellte technische Einheit.

## Transistor




Der **Transistor** ist ein elektronisches Bauelement mit drei Anschlüssen: Ein Eingang für starke Ströme (ca. 1-2A) namens **Collector** (C), ein Eingang für schwache Ströme namens **Basis** (B) (ca. 0-50 mA) und ein gemeinsamer Ausgang für beide Ströme namens **Emitter** (E).

Es können also zwei Ströme gleichzeitig durch einen Transistor fließen: Der relativ schwache sogenannte BE-Strom fließt in die Basis hinein und aus dem Emitter heraus, der relativ starke CE-Strom fließt in den Collector hinein und ebenfalls aus dem Emitter heraus.

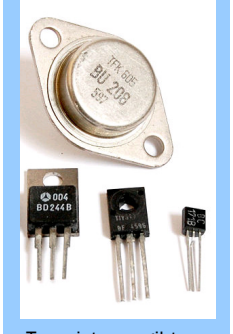



Das besondere an Transistoren ist nun, dass die maximale Stromstärke für den starken CE-Strom durch den schwächeren BE-Strom festlegt wird. Genauer gesagt gilt für den hier verwendeten Transistortyp BD437: Der stärkere CE-Strom kann genau 130 Mal stärker sein, als der schwächere BE-Strom. Mehr lässt der Transistor einfach nicht zu.

Beträgt der BE-Strom also z.B. 1 mA, so kann auf der CE-Strecke eine Stromstärke von bis zu 130 mA fließen, beträgt der BE-Strom 2 mA, werden 260 mA zugelassen... Das geht so weiter bis zur maximalen Stromstärke, ab der der Transistor als ganzes kaputt gehen würde. Sie beträgt für diesen Transistortyp etwa 2A.

Ein Fall ist noch besonders wichtig: Wenn der BE-Strom 0 mA beträgt, kann auch auf der CE-Strecke kein Strom fließen.

Und das ist die Idee unseres **Transistorverstärkers**: Als BE-Strom wird ein Strom vom Port des Mikrocontrollers zu $V_{SS}$ verwendet. Ist der Port low, fließt kein BE-Strom, also kann auch kein CE-Strom fließen. Ist der Port high, fließt ein BE-Strom, also kann auch ein CE-Strom fließen.

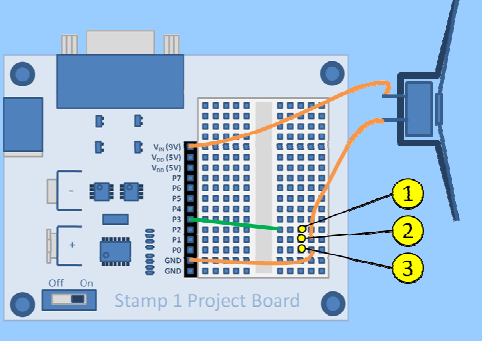



Transistoren gibt es in vielen verschiedenen Bauformen und Arten. Sie unterscheiden sich in der Maximalstromstärke, dem **Transistorfaktor** aber auch in vielen anderen Eigenschaften. Vor allem sind aber auch die Anschlüsse oft in unterschiedlichen Reihenfolgen angeordnet.

### Aufgabe 13.1:

Transistoren lassen sich leicht zerstören, wenn man sie falsch anschließt.

Überlegt also zunächst, welcher Transistoranschluss an welche der Stellen 1, 2 oder 3 angeschlossen werden muss, damit hier das Signal aus Port 3 den starken Strom durch den Lautsprecher steuert. Die Aufgabe ist nicht leicht. Setzt den Transistor erst ein, wenn ihr euch sicher seid.



## Kompilieren

Die Feststellung, dass weder SYMBOL noch Kommentare Speicherplatz auf der BS1 kosten, kann zum Anlass genommen werden, über den Vorgang der Datenübertragung und der Datenspeicherung auf der BS1 zu sprechen: Das BASIC-Programm wird auf dem Computer in einen Bytecode umgerechnet, den man sich im Menu des BSE unter Run - MemoryMap anschauen kann. Diesen Vorgang bezeichnet man als Kompilieren. Jedes eingegebene Wort wird also in ein bis drei Bytes umgewandelt, wobei es auf die Funktionalität und nicht auf das Wort ankommt.

Für den Bytecode hat die BS1 nur 256 Bytes an Platz. Zusätzlich stehen noch 768 Bytes bereit, die die MemoryMap nicht anzeigt. Hier werden zum Beispiel die Texte von DEBUG-Anweisungen gespeichert. Im Heft wird zwar immer davon gesprochen, dass die Daten „auf den Mikrocontroller“ übertragen werden, dies ist aber nicht präzise: Sie werden in einem EEPROM (auf der BS1 das 8-Pin-IC neben dem Spannungsregler) gespeichert und dort Stück für Stück vom µC ausgelesen.

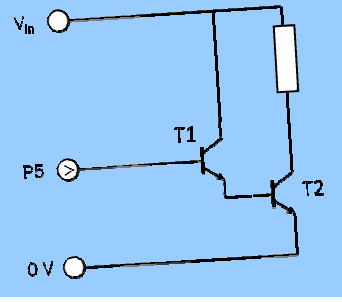
**Lösung 14.1:**
130*130=16900
**Lösung 14.2:**
Zunächst die maximal erlaubte CE-Stromstärke berechnen:
$I_{BE}$=U/R=(5 V-0,7 V)/220 Ohm=19,5 mA.
$I_{CE}$=130*IBE
=130*19,5 mA=2,5 A.
Diese liegt über der Stromstärke, die tatsächlich fließt, weil bei 9V und ca. 6 Ohm Widerstand des Lautsprechers kein so großer Strom fließen kann.

# Einführung in Mikrocontroller

14

**Aufgabe 14.1:**

Wie groß ist der Verstärkungsfaktor in dieser sogenannten **Darlington-Schaltung** zwischen dem BE-Strom des Transistors T1 und dem CE-Strom des Transistors T2, wenn beide Transistoren jeweils einen Verstärkungsfaktor von 130 haben?




**Aufgabe 14.2:**

Berechnet, wie groß die maximale CE-Stromstärke durch den Transistor in Aufgabe 13.1 sein dürfte, wenn Port 3 high ist. Zu berücksichtigen ist dabei neben dem 220 Ohm Widerstand auf der Platine auch, dass in der BE-Verbindung des Transistors eine Vorwärtsspannung abfällt - wie bei einer LED. Allerdings beträgt sie nur etwa 0,7 V.

## Bessere Programmübersicht

Zwei Techniken helfen, auch bei längeren und komplexeren Programmen eine gute Übersicht über den Code zu haben: Zum einen die Anweisung SYMBOL, zum anderen Kommentare, die sich besonders leicht erklären lassen. Beides zeigt das rechts abgebildete Programm.

**Kommentare**: Beginnt eine Zeile mit einem Hochkomma (Shift + #), wird sie bei der Übertragung in den Mikrocontroller ignoriert. Man kann sich also hinter das Hochkomma Kommentare in das Programm schreiben. Auch die Compilerdirektive in der ersten Zeile beginnt mit einem Hochkomma: hier macht sich der BSE sozusagen selbst eine Notiz, die nicht an den µC übertragen wird.

**Symbole** werden üblicherweise am Anfang eines Programms mit Hilfe des Begriffs SYMBOL definiert. Dabei geht es nur darum, Variablen wie z.B. B3 oder PIN5 mit anderen Namen bezeichnen zu können. Der Gewinn an Übersichtlichkeit wird beim rechts dargestellten Ampelprogramm schnell deutlich. Die Anweisung SYMBOL kostet übrigens keinen Speicherplatz auf dem Mikrocontroller.

```
' {$STAMP BS1}
SYMBOL Rot = 0
SYMBOL Gruen = 1
SYMBOL Gelb = 2
OUTPUT Rot
OUTPUT Gelb
OUTPUT Gruen

Start:
  ' grünes Licht einschalten
  HIGH Gruen
  PAUSE 3000
  LOW Gruen
  ' gelbes Licht einschalten
  HIGH Gelb
  PAUSE 1000
  LOW Gelb
  ' rotes Licht einschalten
  HIGH Rot
  PAUSE 2000
  ' gelbes Licht dazu
  HIGH Gelb
  PAUSE 1000
  LOW Rot
  LOW Gelb
  GOTO Start
```

**Lösung 15.:**
b) Es darf nicht der Eindruck entstehen, dass der Wert stabil ist, wenn man die Schaltung nicht berührt.
c) Auch die Verbindung über große Widerstände wie 10 oder 20 kOhm sorgt für einen stabiles Potential am Eingang.

## Eingänge

Es ist sinnvoll, zunächst den Begriff des Messeingangs zu klären, in dem man zum Beispiel auf die Spannungsmessung an einem Oszilloskop oder Multimeter eingeht.

Auch hilft es den Schülerinnen und Schülern, wenn man mit ihnen das Ausmessen der Anschlüsse des verwendeten Schalters vorab bespricht oder gemeinsam durchführt.

Die Darstellung, dass nur eine vernachlässigbar kleine Stromstärke in INPUT-Ports hineinfließt, ist für viele Mikrocontroller richtig, für die BS1 aber nicht so ganz. Die Stromstärke kann durchaus einige Milliampere betragen, für das Detektieren, ob das Pin high oder low ist, ist allerdings das Potential entscheidend. Aus Gründen der Einfachheit wird hier auf die Betrachtung der Ströme an Eingängen verzichtet.

15

# Einführung in Mikrocontroller

## Eingänge

Tipp:
Bei Schaltern mit drei Anschlüssen ist nicht unbedingt der mittlere Anschluss auch der, der wahlweise mit dem einen oder anderen der beiden anderen verbunden wird. Wenn nicht ein Schaltbild auf den Schalter aufgedruckt ist, kann man mit einem Multimeter in der Betriebsart Widerstandsmessung herausfinden, welcher Anschluss welche Bedeutung hat. Oftmals ist der einzeln stehende Anschluss der Mittelanschluss.

Alle Ports des µC lassen sich sowohl als Ausgänge als auch als Eingänge nutzen. So, wie ihr einen Port mit dem Befehl OUTPUT als Ausgang deklariert habt, lässt er sich mit dem Befehl INPUT zum Eingang machen. Eingang bedeutet dabei allerdings nicht, dass Strom in den µC hinein fließt: Ein **Eingang** ist ein Messeingang für Potentiale. Er lässt keine bzw. nur eine vernachlässigbar kleine Stromstärke hineinfließen.

Ist z.B. P2 als Eingang konfiguriert, misst der µC fortlaufend das Potential, das an P2 anliegt und weist der Variable PIN2 den Wert 0 oder 1 zu. 0 immer dann, wenn das Potential unter 1,4V liegt, 1 immer dann, wenn es über 1,4V liegt. Potentiale von ungefähr 1,4V führen zu einem zufälligen Ergebnis - ebenso wie die Situation, wenn kein Potential angeschlossen ist.

So ein Mikrocontrollereingang, der nur die beiden Stufen high und low unterscheidet, wird als digitaler Eingang bezeichnet. Digitale Eingänge werden dazu benutzt, um Programmabläufe mit Hilfe von Schaltern, Tastern, Lichtschranken, Wassermeldern und vielen anderen Sensoren zu beeinflussen.

Der einfachste Sensor dabei ist ein **Umschalter** oder ein **Wechseltaster**: Es gibt drei Anschlüsse, von denen einer (**Mittelkontakt**) bei nicht betätigtem Taster mit einem der beiden anderen, bei betätigtem Taster mit dem anderen der beiden anderen verbunden wird. Das Schaltbild rechts macht das deutlich.

**Aufgabe 15.1:**

Dieses Programm schaltet Port 2 auf Input und sendet den Wert der Variablen PIN2 laufend an den Computer.

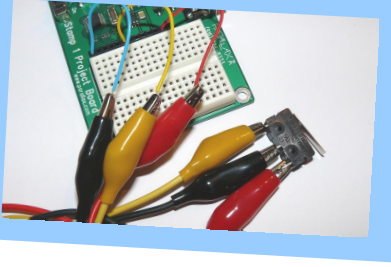
```
' {$STAMP BS1}
INPUT 2
Start:
  DEBUG PIN2
  GOTO Start
```

a) Wenn ihr nun in Port 2 ein Kabel einsteckt und es mit den verschiedenen Potentialen am Anschlussbrett verbindet, seht ihr, wie sich der Wert verändert.

b) Probiert auch aus, was geschieht, wenn das Kabel nicht mit einem Potential verbunden ist sondern offen in die Luft steht. Vor allem wenn man die Schaltung nun bewegt oder das Kabel berührt, wird der Wert sich zufällig ändern.

c) Testet, ob sich der Wert ändert, wenn Port 2 über einen hohen Widerstand (z.B. 10000 Ω oder 50 kΩ) mit den Potentialen $V_{DD}$ oder $V_{SS}$ verbunden wird.

d) Schließt nun einen Wechseltaster so an Port 2, $V_{SS}$ und $V_{DD}$ an, dass sich bequem zwischen dem Wert 0 (low) und dem Wert 1 (high) umschalten lässt.



## Spannungsteiler

Diese Seite stellt vor allem eine gute Übung in das Lesen und Verstehen eines abstrakten technischen Textes dar. Als Lehrer sollte man also gut darauf achten, dass diese Aufgaben auch bearbeitet werden und ausreichend Zeit zur Besprechung einplanen.

Ferner bietet es sich an, in Absprache mit dem Physikunterricht diese einfache Formel zum Spannungsteiler herzuleiten und den Spannungsteiler auf Ketten von Widerständen zu erweitern. Dieser Text kann das in dieser Kürze nicht leisten - und versucht das eben auch gar nicht.

**Lösung 16.1.:**

a) Entweder mit der Formel, oder: Zunächst die Gesamtstromstärke berechnen: I=U/R=5V/(10000 Ohm+5000 Ohm)=0,0003 A.. Also ergibt sich eine Spannung über dem unteren 5kOhm-Widerstand von $U_{down}$=RI=5000 Ohm * 0,0003 A = 1,66V. Das Pin wird also knapp high sein.

c) Nein, es ist gerade umgekehrt.

**Lösung 16.2:**

b) Bei geöffnetem Schalter ist das Potential high, der Spannungsteiler existiert ja nicht. Bei geschlossenen Schalter mit Schalterwiderstand 1 Ohm ist das Potential $U_{down}$=1 Ohm*5V/100010hm = 0,5 mV.

# Einführung in Mikrocontroller

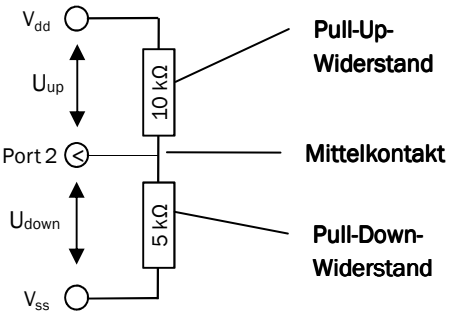
16

## Widerstände und Spannungsteiler

Was geschieht, wenn man einen Eingangsport sowohl an ein Potential von 5V als auch an ein Potential von 0V anschließt? Wird der Port dann high (Wert der Pin-Variable 1) oder low (0) sein?

Wenn man einen Port ganz direkt sowohl mit 5 V als auch mit 0 V verbindet, erzeugt man einen Kurzschluss. Das ist keine gute Idee: Der Spannungsregler wird überhitzen und sich hoffentlich rechtzeitig abschalten, bevor etwas kaputt geht.

Interessanter wird es, wenn man die beiden Verbindungen vom Port zu $V_{DD}$ bzw. vom Port zu $V_{SS}$ jeweils mit Widerständen erzeugt. So entsteht kein Kurzschluss, da nur ein schwacher Strom durch die beiden Widerstände am Mikrocontroller vorbei fließt. Diese Schaltung bezeichnet man als Spannungsteiler mit einem Mittelkontakt.




Der Pull-Up-Widerstand $R_{up}$ und der Pull-Down-Widerstand $R_{down}$ teilen sich die Gesamtspannung entsprechend ihrer Anteile am Gesamtwiderstand in die Spannung $U_{up}$ und $U_{down}$ auf. Das Potential am Port des Mikrocontrollers entspricht $U_{down}$ und lässt sich aus den Widerstandswerten und der Gesamtspannung berechnen:

$$U_{down} = U_{gesamt} \cdot \frac{R_{down}}{R_{gesamt}}$$

Herleitung:
Die angegebene Formel zur Berechung des Potentials des Mittelkontakts hat eine recht einfache Herleitung aus dem Ohmschen Gesetz:
Durch beide Widerstände muss die gleiche Stromstärke fließen - sie beträgt $I_{gesamt}=U_{gesamt}/R_{gesamt}$.
Dabei ist $R_{gesamt}=R_{up}+R_{down}$.

Wenn man nun nur den unteren Widerstand betrachtet, so muss die Spannung nur an diesem Widerstand so groß sein, dass sie diese Stromstärke durch diesen Widerstand treiben würde. Also $U_{down}=R_{down}$ $I_{down}$
Da $I_{down}=I_{gesamt}$ ist, ergibt sich durch Einsetzen von $I_{gesamt}$ die angegebene Formel.

**Aufgabe 16.1:**

a) Berechne das Potential am Mittelkontakt (also $U_{down}$) mit den Widerstands- und Spannungswerten der Abbildung oben rechts. Würde der Input-Port high (PIN2=1) oder low (PIN2=0) sein?

b) Probiere aus, wie sich diese Kombination von Widerständen in der Praxis an einem Port des Mikrocontrollers verhält.

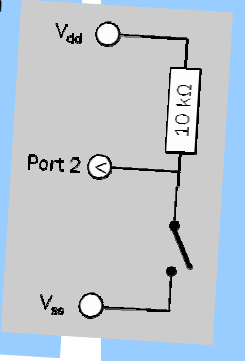
c) Ist die folgende Aussage richtig? Je kleiner $R_{down}$ im Verhältnis zu $R_{up}$ ist, desto eher ist der Eingabe-Port high.

**Aufgabe 16.2:**

Mit Hilfe eines Spannungsteilers kann man auch normale Taster oder Schalter an den Eingängen eines Mikrocontrollers einsetzen. Setzt als Pull-Up-Widerstand einen Widerstand von 10 kΩ ein, an Stelle des Pull-Down-Widerstands einen Schalter. Er hat in geöffnetem Zustand einen Widerstand von nahezu unendlich (über 100 MΩ), im geschlossenen Zustand einen Widerstand von unter 1Ω.

a) Testet, wie sich der Zustand eines Eingangs-Ports verändert, wenn der Schalter geöffnet bzw. geschlossen wird.

b) „Berechnet“ das Potential am Port bei geöffnetem und geschlossenem Schalter.



**Lösung 17.1:**
Die Schaltung wird bei größeren Pull-Up-Widerständen empfindlicher.

## CNY70

Der Einsatz des CNY70 ist nicht ganz einfach - ein klärendes Gespräch zwischen dem Lesen der Seite und der Realisierung der Schaltung ist hilfreich.

Im Schülertext ist angegeben, dass die Infrarot-LED des CNY70 nur mit 15mA betrieben werden sollte. Das ist nicht ganz richtig: Der CNY70 lässt sich mit bis zu 30mA bei einer Vorwärtsspannung von 1,25V betreiben - dann ist der Spannungsregler auf der Platine aber am Ende seiner Leistungsfähigkeit.

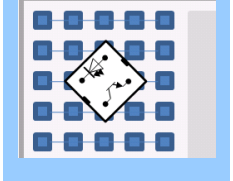
Der CNY70 eignet sich in Schülerprojekten einerseits als Sensor beim Bau von Fahrzeugen, die einem am Boden befindlichen schwarzen Linie folgen sollen. Andererseits eignet er sich auch als Positionssensor, wenn sich ein Motor nur genau um einen festen Winkel drehen soll: Der Motor wird dann mit einer schwarz-weißen Winkel-Rasterscheibe verbunden, neben der sich ein CNY70 befindet. So kann er mitzählen, wie weit sich der Motor gedreht hat und diesen wieder rechtzeitig ausschalten.

# 17 Einführung in Mikrocontroller

CNY70?
Die Bezeichnung CNY70 folgt keiner tieferen Systematik. Der Reflexoptokoppler wurde von seiner Heilbronner Herstellerfirma Vishay (früher Telefunken) so benannt. Auch andere Hersteller, die ein identisches Bauteil anbieten, verwenden dann diesen Namen.

Tipps:
1. Das Infrarot-Licht der sendenden Leuchtdiode ist zwar für das menschliche Auge nicht sichtbar, die meisten Handykameras und einfache Digitalkameras erkennen das Licht aber als weißes Leuchten. So könnt ihr leicht überprüfen, ob die LED auch leuchtet.

2. Da zwei der vier Anschlüsse mit $V_{SS}$ verbunden werden müssen, kann man den CNY70 einfach diagonal auf das Steckbrett setzen:

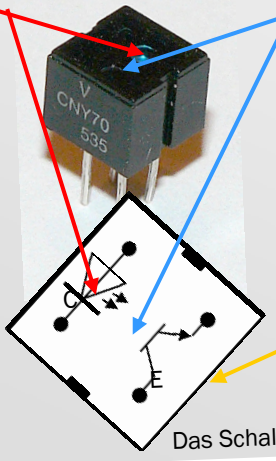


## Reflexoptokoppler

Ähnlich wie den einfachen Schalter oder Taster in Aufgabe 16.2 kann man auch eine Lichtschranke an einen Eingangsport des Mikrocontrollers anschließen.

Jede Lichtschranke besteht aus einem Sender und einem Empfänger. Hier verwenden wir die Kurzstrecken-Reflexlichtschranke CNY70, bei der sich Sender und Empfänger im gleichen Gehäuse befinden - das ausgesendete Licht muss also reflektiert oder zurückgestreut werden, damit es den Empfänger trifft. Dazu genügt ein helles Objekt, das sich in wenigen mm Abstand vor der Reflexlichtschranke befindet.

Der **Sender** ist eine LED, die für das menschliche Auge unsichtbares Licht (Infrarot-Licht) aussendet. Ihre Vorwärtsspannung beträgt 1,6 V, die Idealstromstärke 15mA.



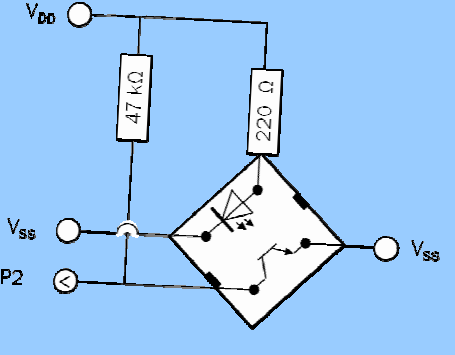

Der **Empfänger** ist ein sogenannter **Phototransistor**. Er hat nur Collector (C) und Emitter (E), die Basis ist durch eine lichtempfindliche Schicht ersetzt. Je mehr Licht auf diese Fläche fällt, desto stärker kann ein Strom von C nach E fließen.

Auf einer Seite des CNY70 ist die Beschriftung aufgedruckt. Auf dieser **Beschriftungsseite** befindet sich im Inneren der Phototransistor.

Das Schaltbild zeigt den CNY70 von oben.

**Aufgabe 17.1:**

Testet den CNY70 an einem Eingangsport des Mikrocontrollers, in dem ihr ihn an Stelle des Pull-Down-Widerstands in einen Spannungsteiler einsetzt.




Als helle Fläche für das Zurückstreuen des Lichts eignen sich ein weißes Blatt oder möglicherweise auch ein Finger. Das hängt davon ab, wie groß ihr den Pull-Up-Widerstand wählt: Probiert Werte zwischen 10 kΩ und ca. 60 kΩ aus.

Wird die Schaltung bei größeren oder kleineren Pull-Up-Widerständen empfindlicher? Warum?

Wo in dieser Schaltung befindet sich nun der Spannungsteiler?

## IF-THEN

Im Schülertext ist die Verknüpfung mehrerer Bedingungen mit Hilfe der logischen Operatoren AND und OR nur angerissen. Dies kann vertieft werden, auch weil diese logischen Verknüpfungen eine allgemein große Bedeutung haben.

Auch bietet es sich an, komplizierte Beispiele für die Verknüpfung der Resultate von zwei Ports an der Tafel „programmieren“ zu lassen.

Wenn die erste Gruppe mit 18.1 fertig ist, lohnt es sich, diese Schaltung in der nächsten Anfangsrunde zu präsentieren um die Uneindeutigkeit eines offenen Eingangs hervorzuheben.



## IF-THEN

An einem ziemlich einfachen Beispiel lernt ihr hier kennen, wie man den Ablauf eines Programmes verändern kann, wenn sich der Wert eines Inputs bzw. der dazu gehörenden Pin-Variable verändert.

Das Beispiel verwendet als Eingang den Port 2, an den z.B. einfach nur ein Kabel angeschlossen ist, das sich leicht mit $V_{DD}$ oder $V_{SS}$ verbinden lässt. Man könnte hier auch einen Wechseltaster oder Umschalter verwenden. Wenn Port 2 high ist, also PIN2=1, soll der Text „Port 2 high“ gesendet werden, sonst wird immer wieder der Satz „Port 2 low“ gesendet. Dabei hilft eine neue und besonders wichtige Anweisung: `IF Bedingung THEN Stelle`

Wenn die **Bedingung**, zum Beispiel eine Gleichung, erfüllt ist, wird das Programm an der angegebenen Stelle fortgesetzt. Ansonsten geht es einfach mit der nächsten Zeile weiter. Dies zeigt nun auch das Beispiel:

Wenn Port 2 high ist, ist die Bedingung , dass Pin2 0 sein soll, nicht wahr. Also wird das Programm einfach mit der nächsten Zeile fortgesetzt und nicht an die Stelle „weiter“ gesprungen.

Dort wird der Text „Port 2 high“ ausgegeben. Dann wird das Programm wieder bei Start fortgesetzt, damit die weiteren Zeilen nicht auch ausgeführt werden.

```
' {$STAMP BS1}
INPUT 2
OUTPUT 5
OUTPUT 6

Start:
  If PIN2=0 THEN Weiter
    DEBUG “Port 2 high“
    GOTO Start
Weiter:
    DEBUG “Port 2 low“
    GOTO Start
```

Wenn Port 2 low ist, ist die Bedingung wahr. Also wird das Programm am Label „Weiter“ fortgesetzt.

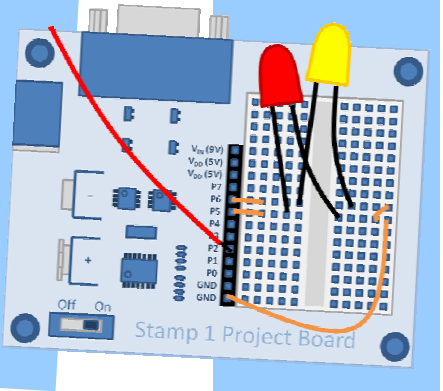
Dort wird „Port 2 high“ gesendet, bevor der Programmablauf wieder an den Anfang zu „Start“ zurück springt.

**Aufgabe 18.1:**

a) Überlegt euch genau, wann welche Zeilen in obigem Beispiel ausgeführt werden und wie das Programm bei PIN2 high bzw. low abläuft.

b) Verwendet die Lichtschranke am Eingang Port 2 und schließt zwei Leuchtdioden an zwei weitere Ports an. Schreibt ein Programm, das immer, wenn Port 2 high ist, nur die eine Leuchtdiode leuchten lässt, immer wenn Port 2 low ist, nur die andere.

c) Wenn ihr nur ein einfaches, offen stehendes Kabel an Stelle der Lichtschranke an Port 2 verwendet, ist es zufällig, welche LED leuchtet. In der Nähe (nicht anschließen!) von Wechselstromkabeln ändert sich der Wert laufend. Es scheint dann so als würden beide LEDs leuchten. So funktionieren auch Kabelfinder für Bauarbeiter und Handwerker.




Beispiele für Bedingungen:
Als Bedingungen kann man auch Ungleichungen mit >, <, >= (größer oder gleich), <=, <> (ungleich) verwenden. Wenn man eine Variable verwendet, sollte sie links stehen. Bsp:
IF B3=100 THEN
IF B2>20 THEN
IF PIN4<>0 THEN

Man kann auch zwei Bedingungen mit Hilfe von AND und OR verknüpfen. Beispiele:

IF PIN1=1 AND PIN2=0 THEN ... springt nur, wenn sowohl der erste als auch der zweite Teil wahr sind. Es muss also PIN1 1 und PIN2 0 sein.

IF PIN1=1 OR PIN2=0 THEN ... springt, wenn einer der beiden Teile erfüllt ist - oder sogar beide.

**Lösung 19.1:**
15 Sound-Anweisungen

**Lösung 19.2:**
Wenn beide Input-Pins high sind, spielt das Programm eine riesen_Fanfare, ist nur Pin 1 high, spielt das Programm die hohe, bei Pin 2 die tiefe Fanfare. Ist keines der Pins high, spielt das Programm keine Fanfare.

## GOSUB und RETURN

Diese beiden Anweisungen ermöglichen eine wenigstens einigermaßen strukturierte Programmierung in Form von Unterprogrammen. Daher sind sie von höchster Wichtigkeit, damit die Schülerinnen und Schüler später eigene, kompliziertere Programme schreiben können. Erstens lässt sich nur so eine Übersicht halten, zweitens können die Schüler mit Hilfe dieser Anweisungen z.B. erst die Drehung eines Motors programmieren und testen und diese dann später in ein Unterprogramm packen und dann erst mit der Programmlogik verbinden. Man bezeichnet dies als Bottom-Up-Programmierung. Diese ist typisch für das sehr Programmieren von Systemen, an deren Hardware gleichzeitig entwickelt wird. Hingegen werden reine Softwareprogramme oft Top -Down entwickelt: Hier lässt man die Unterprogramme leer und schreibt zunächst das Hauptprogramm. Später kann man dann die vielen kleinen Unterprogramme mit Inhalt füllen.



## Unterprogramme

Ein hervorragende Methode, komplexe Programme zu strukturieren, bietet das Anweisungspaar GOSUB und RETURN.

GOSUB funktioniert ähnlich wie GOTO: Man gibt eine als Label (Wort mit Doppelpunkt) markierte Stelle an, an der das Programm fortgesetzt werden soll. Zusätzlich merkt sich der Mikrocontroller aber die Stelle, von der der Sprung ausging. Stößt der Programmablauf nun auf die Anweisung RETURN, wird das Programm in der Zeile nach dem GOSUB fortgesetzt. Der Mikrocontroller hat also mitten im Programm ein sog. „**Unterprogramm**“ ausgeführt.

**Aufgabe 19.1:**

Wie viele einzelne Sound-Anweisungen werden in diesem Programm insgesamt ausgeführt?

```
' {$STAMP BS1}
OUTPUT 3
GOSUB hohe_Fanfare
GOSUB tiefe_Fanfare
GOSUB hohe_Fanfare
GOSUB riesen_Fanfare
END

hohe_Fanfare:
  SOUND 3, (80, 100)
  SOUND 3, (90, 100)
  SOUND 3, (100, 100)
  RETURN

tiefe_Fanfare:
  SOUND 3, (30, 100)
  SOUND 3, (50, 100)
  SOUND 3, (70, 100)
  RETURN

riesen_Fanfare:
  GOSUB tiefe_Fanfare
  GOSUB hohe_Fanfare
  RETURN
```

**Aufgabe 19.2:**

Dieses Programm verwendet die gleichen Unterprogramme wie das Programm aus Aufgabe 19.1. Was macht dieses Programm?

```
' {$STAMP BS1}
INPUT 1
INPUT 2
OUTPUT 3
Start:
      IF PIN1<>1 OR PIN2<>1 THEN Weiter
            GOSUB riesen_Fanfare
            GOTO Start
      Weiter:
      IF PIN1<>1 THEN Weiter2
            GOSUB hohe_Fanfare
            GOTO Start
      Weiter2:
      IF PIN2=0 THEN Start
            GOSUB tiefe_Fanfare
            GOTO Start
```

Analysiere dieses Programm für jeden der Fälle in einer solchen Tabelle:

| INPUT 1 | INPUT 2 | Resultat |
|---|---|---|
| low | low | |
| high | low | |
| low | high | |
| high | high | |

Begrenzung:
Die Basic Stamp kann nicht beliebig viele GOSUBs zurückverfolgen. In einem Programm dürfen maximal 16 GOSUBs vorkommen, die bis zu vier Ebenen tief verschachtelt sein können.

## Kondensator

Hinweis: Elektrolyt- oder Tantalkondensatoren können bei falscher Polung platzen. daher hier nur Folien- oder Keramikkondensatoren mit einer passenden Spannungssicherheit verwenden.

Die Art und Weise, wie die Messung hier funktioniert, ist als solche bereits interessant: Zunächst schaltet die BS1 den Port als Ausgang und lädt den Kondensator über den Widerstand einige ms lang auf. Er ist nun nahezu voll.

Dann wird der Port zum Input gewandelt und eine interne Stoppuhr gestartet. Zunächst liegt am Port ein hohes Potential an, weil der Kondensator geladen ist. Da auch in einen Input bei der BS1 ein Strom hineinfließen kann, entlädt sich der Kondensator, so dass das Potential sinkt. Wenn es die Schwelle von 1,4V erreicht wird der Port low und die Stoppuhr gestoppt.

Aus der Zeit (gezählt in einer Word-Variable) lässt sich auf den Widerstand schließen, der den Entladestrom ja bremst. Die Zeit wird dazu noch durch den Skalierungsfaktor geteilt.

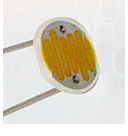
**Lösung 20.2:**

```
' {$STAMP BS1}
OUTPUT 1

Start:
  POT 3, 100, B0
  IF B0>50 THEN Lichtaus
    HIGH 1
    GOTO Start

Lichtaus:
  LOW 1
  GOTO Start
```



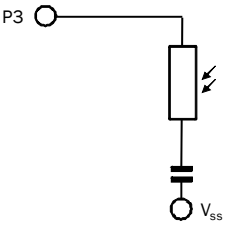
## Analoge Messungen

Die Eingänge der Basic Stamp kennen nur die beiden Werte 0 und 1 - zu wenig um z.B. Temperatur oder Helligkeit zu messen. Manche Mikrocontroller haben sogenannte **A/D-Ports**, die ein analoges Potential (also stufenlos) z.B. in eine Bytezahl umwandeln können. Das ist nicht ganz stufenlos, aber schon besser. Die Basic Stamp verfügt nicht über solche A/D-Ports, kann aber mit einem Trick und der Anweisung POT zumindest Widerstände als Bytewert messen. Das ist nicht so schlecht, denn es gibt für einige Größen Sensoren, die ihren Widerstand verändern:



Für Licht gibt es **LDR**s (Light Dependent Resistor). Je heller es ist, desto geringer ist der Widerstand. Für Temperatur gibt es **NTC**s: Je höher die Temperatur, desto geringer der Widerstand. Und für Winkel kann man die euch bekannten Potentiometer verwenden, um z.B. zu erfassen, wie weit sich etwas gedreht hat.

Um den Widerstand z.B. eines LDR zu messen, schließt man ihn an den Port und an einen sog. Kondensator an, dessen anderes Ende an $V_{SS}$ angeschlossen ist....und dann ruft man die Anweisung POT auf:




```
POT Port, Faktor, Bytevariable
```

Die Anweisung liest nun den Wert des Widerstands aus, wofür sie ungefähr 0,02 Sekunden benötigt. Dann rechnet sie den Wert mit Hilfe des Umrechnungsfaktors auf eine Zahl zwischen 0 und 255 um und speichert das Ergebnis in der angegebenen Bytevariable, also B0 bis B11.

Das funktioniert nur, wenn der Kondensator die richtige „Größe“ hat. Die passende Größe in der Einheit Farad (F) zeigt die Tabelle:

| max. Widerstand | Kondensator | Beschriftung |
|---|---|---|
| 5 kΩ - 20 kΩ | 0,1 µF | 104 |
| 20 kΩ - 50 kΩ | 0,01 µF | 103 |

### Aufgabe 20.1:

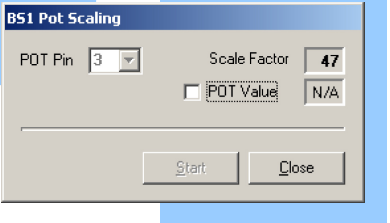
Schließt ein Poti oder einen LDR mit dem passenden Kondensator an Port 3 an und führt das folgende Programm mit einem geeigneten Skalierungsfaktor aus. Es zeigt nun laufend den abgelesenen Wert:

```
' {$STAMP BS1}
Start:
  POT 3, 127, B0
  Debug B0
  GOTO Start
```

### Aufgabe 20.2: Babylicht

Schreibt ein Programm, das ein Licht (LED) einschaltet, sobald der LDR sich in Dunkelheit befindet. Dazu benötigt ihr so etwas wie

```
IF B0<50 THEN ...
```




**Kondensatoren** sind im Prinzip winzig kleine Akkus, die sich nur kurzzeitig laden lassen. Hier setzt ihr Keramik- oder Folienkondensatoren ein, die nicht mit den gefährlichen Elektrolytkondensatoren verwechselt werden dürfen.

**Skalierung:**
Um den passenden Skalierungsfaktor zu ermitteln, findet ihr im Menu des BSE unter Run das Programm POT-Scaling. Man startet es, nachdem man die Schaltung aufgebaut hat, und gibt den Port ein, an den der veränderliche Widerstand angeschlossen ist. Dann verändert man den Widerstand auf seinen höchstmöglichen Wert - bei einem LDR also: gut abdunkeln. Der dann angezeigte „Scale Factor“ ist der geeignete.

## ICs

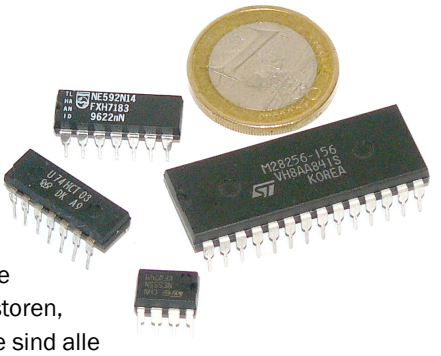
Um die Bandbreite an ICs zu veranschaulichen, kann man hier Referate verteilen, die jeweils ein IC vorstellen. Es eignen sich dazu alle Digital-ICs, also zum Beispiel aus der 74xxx-Baureihe oder der NE555.

Die Schülerinnen und Schüler entdecken dabei, dass sich einfache Schaltungen auch ohne einen Mikrocontroller realisieren lassen - spannend!

Häufigste Schülerfehler beim Hardwareaufbau mit dem Motortreiber-IC sind nicht richtig angedrückte ICs sowie vergessene GND-Verbindungen. Es müssen alle entsprechend gekennzeichneten Pins mit GND ($V_{SS}$) verbunden werden. Den Schülern kann empfohlen werden, sich einen möglichst einfachen Steckplan im Notizheft einzuzeichnen: Dieser kann z.B. die GND-Pins zunächst alle untereinander verbinden und dann eine Verbindung zu $V_{SS}$ herstellen. Dies erhöht zusammen mit einer sinnvollen Farbwahl bei den Kabeln die Übersichtlichkeit enorm.



## ICs, Integrierte Schaltkreise



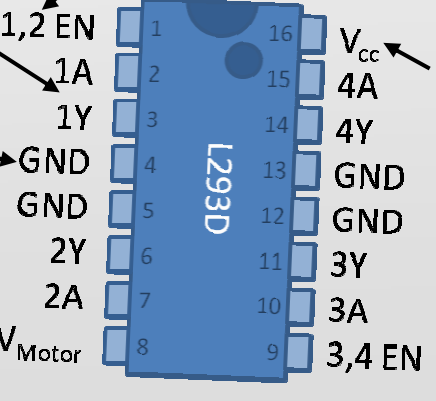
Auf dieser Seite lernt ihr **IC**s (integrated circuit = **integrierter Schaltkreis**) kennen - eigentlich kennt ihr schon recht viele: Die kleinen schwarzen Hilfsbauteile (z.B. der Spannungsregler), die sich zusammen mit dem Mikrocontroller auf dem BS1 Project Board befinden, sind größtenteils ICs. Jedes IC erfüllt dabei eine spezielle Funktion, die man ansonsten nur mit Hilfe vieler Transistoren, Widerstände und Dioden erfüllen könnte. Diese Bauteile sind alle in das kleine Gehäuse eines ICs integriert - daher der Name „integrierter Schaltkreis“. Nur die Anschlüsse ragen als Pins aus dem schwarzen Gehäuse heraus.

Auf der folgenden Seite werdet ihr ein IC verwenden, um einen Elektromotor vom Mikrocontroller aus anzusteuern. Schon kleine Elektromotoren benötigen recht große Stromstärken von 0,5 bis 1,5 A, so dass man sie nicht direkt an die Ports des µC anschließen kann. Das sogenannte **Motortreiber-IC** mit dem Namen L293D beinhaltet eine Schaltung, die ähnlich wie ein Transistor funktioniert, aber sogar gleich zwei Motoren vorwärts und rückwärts antreiben kann - gesteuert von den Ports des Mikrocontrollers. Sie besteht eigentlich aus über 10 Transistoren, aber für den Einsatz des ICs braucht man nur zu wissen, wo man was anzuschließen hat:

1A, 2A, 3A und 4A sind Eingänge des ICs und werden mit Ports des Mikrocontrollers verbunden. 1Y, 2Y... sind Ausgänge, an die ein Motor angeschlossen werden kann.

1,2EN und 3,4EN sind sogenannte **Enable-Pins**. Ist 1,2EN mit einem Potential unter 1,4 V verbunden (low), werden die Eingänge 1A, 2A sowie die Ausgänge 1Y und 2Y in einen Stromsparbetrieb geschaltet und funktionieren nicht. Diese Pins sollten mit etwas verbunden sein, das high ist, z.B. $V_{DD}$.

**GND** steht für **Ground**, eine andere Bezeichnung für $V_{SS}$. Diese Pins müssen also einfach mit $V_{SS}$ verbunden werden, wenn man das IC verwenden möchte.




Hier muss das Betriebspotential für die Motoren angelegt werden. Auf eurem Projekt Board gibt es dafür nur eine Quelle: $V_{IN}$.

$V_{CC}$ muss mit $V_{DD}$ auf der µC-Platine verbunden werden. Es ist die 5V-Energieversorgung für das IC.

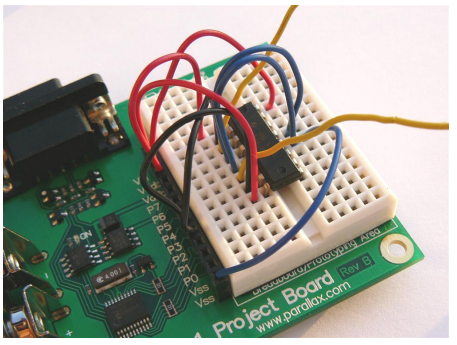
Pins an einem IC:
Die Pins an ICs haben eine einheitliche Nummerierung: Man zählt von links oben nach unten und auf der anderen Seite wieder herauf. Damit man weiß, wo oben ist, ist dieses Ende entweder mit einem eingestanzten Halbkreis oder Kreis gekennzeichnet.

An der Ausrichtung des Textaufdruckes darf man sich nicht orientieren.

Datenblätter:
Es gibt viele tausend Typen von ICs, eigentlich immer mindestens eines zu jedem häufigeren Anwendungsfall - manchmal auch mehrere konkurrierende Modelle von verschiedenen Hersteller. Zu jedem IC gibt es ein Datenblatt, das die Bedeutung jedes Pins genau erklärt.

Es gibt im Internet verschiedene Seiten, die die Originaldatenblätter der Hersteller (Englisch) kostenlos zum Download anbieten,

## Motoransteuerung




Die Motoransteuerung steht am Ende dieses Hefts, weil sie die höchste Hard- und Software-Komplexität aufweist - und weil damit offensichtlich ist, was man alles an tollen Projekten mit Mikrocontrollern realisieren kann! Der ideale Zeitpunkt um mit der Projektarbeit zu starten.

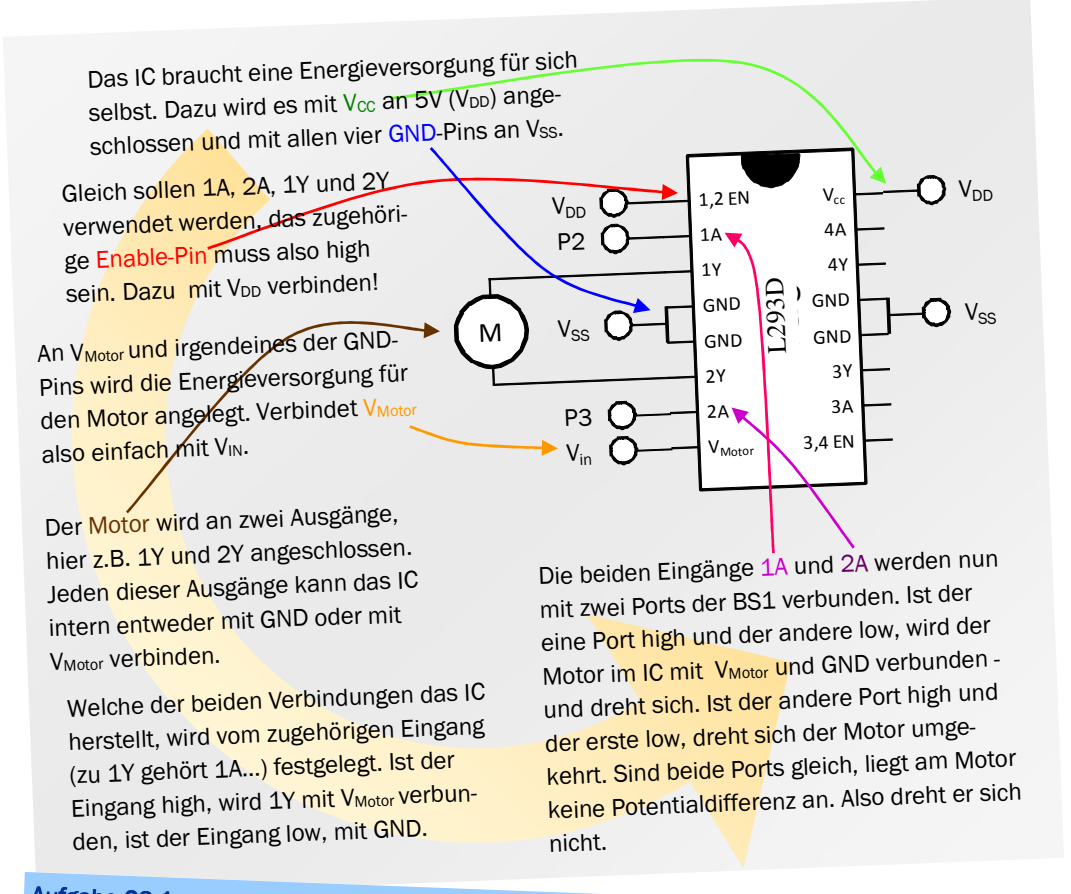
Hinweis: Bei der Verwendung mancher USB-RS232-Adapter kommt es mit dem Motor des alten Sets (siehe Einleitung) zu Abstürzen des PCs, wenn der Motor während der Übertragung angeschlossen ist. Es empfiehlt sich, den Motor vor jeder Programmänderung abzutrennen.

Hinweis: Beim Zusammenbau des Getriebemotors des neuen Sets ist das Übersetzungsverhältnis 1:9 empfehlenswert. Dabei darf das Ritzel nicht zu weit auf die Achse geschoben werden.



## Motorsteuerung

Ein einfacher **Elektromotor** hat zwei Anschlüsse. Wenn eine Spannung anliegt, dreht sich der Motor in die eine Richtung, kehrt man die Spannung um, dreht sich der Motor in die andere Richtung. Welches Pin man dazu wo am L293D anschließen muss und wie man die Drehrichtung vom Mikrocontroller aus steuern kann, wird hier erklärt:



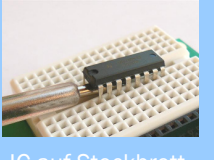

**Aufgabe 22.1:**

Schreibt euch ein Programm in einer Endlosschleife, das die Pins 2 und 3 auf Ausgang schaltet und dann Pin2 auf high und Pin3 auf low setzt. Nach einer Pause von einigen Sekunden soll das Programm die beiden Pins umgekehrt setzen. Und so weiter.

Schließt nun an diesem Testprogramm zunächst den L293D und dann den Motor mit allen notwendigen Verbindungen an. Der Motor sollte sich nun abwechselnd in eine und dann in die andere Richtung drehen!

Hinweis:
Auch wenn man nur einen Motor am L293D betreibt, müssen alle Ground-Pins an VSS angeschlossen werden.

TTL
Es ist schön, dass das „high“ und das „low“ des Mikrocontrollers so gut zu den Anforderungen des high und low des Motortreiber-ICs passt. Es liegt daran, dass beide Bauteile der gleichen internationalen Festlegung, nämlich der TTL-Norm folgen.



IC auf Steckbrett
Das IC passt genau über den mittleren freien Bereich des Steckbretts. ICs müssen auf Steckbrettern richtig fest aufgedrückt werden, damit sie Kontakt haben. Zum Lösen eines ICs benutzt man entweder eine Spezialzange oder einen Schraubenzieher. Beim Herausziehen mit der Hand verbiegen sich die Pins oder brechen ab.

**Lösung 23.1:**
Der Servo erreicht irgendwann seine Zielstellung—dann verändert sich ja nichts mehr. Das IC im Servo erkennt, dass die richtige Stellung bereits erreicht ist und steuert den Motor nicht mehr weiter.

## Servo

Das Thema Servo ist erst in der zweiten Auflage dieses Heftes hinzugenommen worden. Servos bieten den Schülern viele Freiheiten bei der Realisation technischer Vorhaben und lassen sich sehr einfach ansteuern.

Modellbauservos sind üblicherweise für Spannungen von 4,8 bis 6V ausgelegt, bei nicht zu hoher Belastung lassen sich günstige Servos (ca. 5€) aber auch bei 9V einsetzen. Ansonsten ist es auch möglich, die BS1 komplett mit nur 6V zu betreiben. Auch kann man den Servo an einer anderen Stromquelle als die BS1 betreiben (Plus an rot, Minus an schwarz). Zusätzlich müssen dann die Minuspole der Stromquelle und der BS1 miteinander verbunden werden.

Ein (defekter?) Servo mit vorsichtig aufgesägter Seitenwand ist ein interessantes Exponat für den Unterricht.

**Lösung 23.2:**

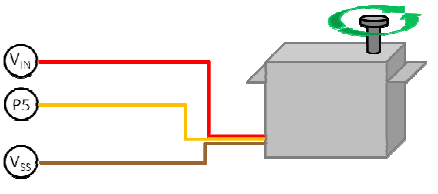
```
Start:
  FOR  B0=0 TO 30
    LOW 5
    PULSOUT 5, 100
    PAUSE 20
    NEXT B0
  FOR  B0=0 TO 30
    LOW 5
    PULSOUT 5, 200
    PAUSE 20
    NEXT B0

    GOTO Start
```

# 23 Einführung in Mikrocontroller

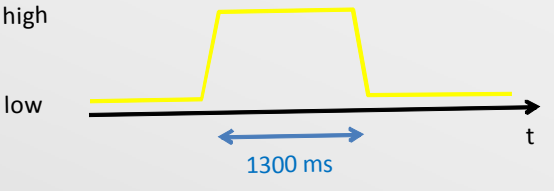
## Servo

Hier lernt ihr einen sogenannten "**Servoantrieb**" kennen - genauer gesagt einen **Modellbauservo**. Er wird zum Beispiel für die Lenkung von Modellfahrzeugen oder für die Bewegung der Steuerruder in Modellflugzeugen verwendet. Ein Servo kann Winkel gezielt und relativ schnell ansteuern - dafür kann sich seine Achse aber nicht wie beim Motor mehrfach im Kreis herum drehen.

Ein typischer Modellbauservo verfügt über drei Anschlüsse. Zwei davon sind für die Stromversorgung (rot für 6V, schwarz oder braun für $V_{SS}$), über den dritten (meist gelben oder orangenen) Anschluss erhält der Servo die Information, auf welchen Winkel er sich einstellen soll. Dieser Anschluss kann direkt mit einem Port des µC verbunden werden. Die Informationsübertragung über nur einen Pin erfolgt durch die sogenannte **Pulsweitenmodulation**, kurz: **PWM**.




Ein spezielles IC im Servo "lauscht" auf der gelben Leitung, bis der Mikrocontroller diese Leitung auf "high" setzt. Ist die Leitung "high", misst es die Zeit, bis die Leitung wieder "low" wird. Dauerte dieser positive **Puls** 1ms (1000µs), soll das Servo möglichst nach ganz rechts gedreht werden, ein Puls von 2ms steht für den linken Anschlag. Pulslängen zwischen 1 und 2 ms entsprechen den Winkeln dazwischen.




Nach jedem Puls dreht der Servo seine Achse etwa 20ms lang in Richtung des Zielwinkels. Unabhängig davon, ob der Zielwinkel erreicht ist, endet die Bewegung nach etwa 20ms – ein erneuter Puls ist nötig, um die Bewegung fortzusetzen.

Um mit der BS1 einen solchen Puls zu erzeugen, setzt man den Port zunächst auf low und verwendet dann die Anweisung

`PULSOUT Port, Dauer.`

Die Dauer wird dabei in Einheiten von jeweils 10µs gemessen: `PULSOUT 5,130` erzeugt also einen 1300µs-Puls auf Port 5.

**Aufgabe 23.1:**
Ein Servo ist an $V_{IN}$, $V_{SS}$ und Pin 5 angeschlossen. Zunächst bewegt er sich, dann tut sich - auch wenn man die BS1 neu einschaltet - nichts mehr. Warum?

```
' {$STAMP BS1}
FOR B0=0 To 30
  LOW 5
  PULSOUT 5, 175
  PAUSE 20
NEXT B0
```

**Aufgabe 23.2:**
Schreibe ein Programm, das einen Servo abwechselnd immer nach ganz links, dann nach ganz rechts, wieder nach ganz links usw. dreht. Warum sind For-Next-Schleifen hier so praktisch?

Hinweis:
Modellbauservos sind für die im Modellbau üblichen Spannungen von 4,8 bis 6 V ausgelegt. Beim Betrieb an 9V sollte man die Achse nicht zu stark belasten.

Was ist ein Servo?
Der Ausdruck Servo steht für ein technisches Teilsystem, in dem ein Antrieb mit einer Ansteuerelektronik fest verbunden ist und als Komponente fix und fertig eingesetzt werden kann.

Wie funktioniert ein Modellservo?
Im Inneren eines Servos befinden sich ein Elektromotor mit einem Getriebe, ein spezielles Motortreiber IC und ein Potentiometer. Das Poti befindet sich auf der Achse und ist auch der Grund, weshalb ein Servo keine Drehungen von mehr als 300° ausführen kann.

Mit dem Poti misst das IC, auf welchem Winkel die Achse steht und kann dann den Puls des Mikrocontrollers entsprechend umsetzen.

## Klassenarbeit?

Als Aufgaben für eine Klassenarbeit bieten sich Abfragen, Berechnungen und auch kombinierende Aufgaben an: Abfragen zu einzelnen Fachbegriffen, Rechnungen zu Vorwiderständen und rund um die Transistoren, kombinierende Aufgaben, in denen Schülerinnen und Schüler selbst eine Hard- und Software zu einer Problemstellung auf dem Papier entwickeln, darstellen und deren Funktionsweise kurz beschreiben sollen.

Beispiel: Beschreibe, wie man aus zwei gegenüber einander angeordneten CNY70 eine Lichtschranke konstruieren könnte, mit der sich der Durchgang einer Maus durch ein Mauseloch detektieren ließe. Zeichne ein Schaltbild und schreibe ein Programm, mit dem die Maus erkannt werden kann und das die Maus dann durch einen möglichst lauten (!) Ton vertreibt.

Es ist günstig, derartige Aufgaben (als Hausaufgabe) bereits ein oder zwei Mal während des Unterrichtsgangs „Mikrocontroller“ gestellt zu haben.



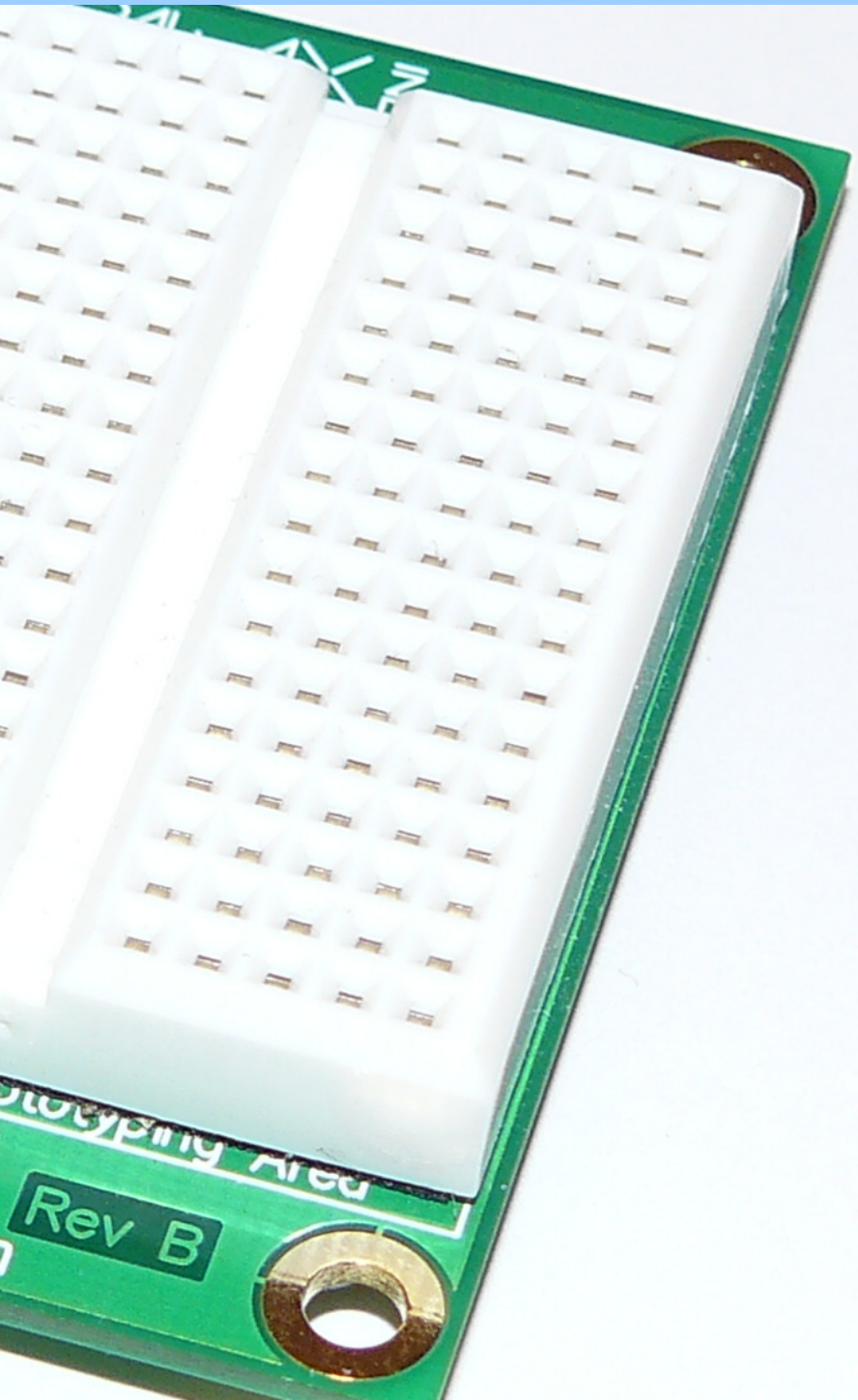


## Index



Tipp:
Dieser Index hilft einerseits, schnell die Seite zu einem vergessenen Fachbegriff zu finden. Andererseits ist er auch eine gute Abfrageliste, mit der ihr euch vor einer Klassenarbeit über das Thema Mikrocontroller auf euer Wissen testen könnt:

Kennt ihr die Bedeutung all dieser Begriffe und Abkürzungen?


Rev B