



Inhaltsverzeichnis

Was ist ein Mikrokontroller?.....	1
Hardware und Software.....	1
Das Seeeduinoboard.....	1
Die Programmiersoftware BASCOM.....	2
Die Anschlüsse des Mikroprozessors.....	7
Schnittstelle zur Außenwelt.....	8
Das Prototype-Shield.....	8
LEDs als Statusanzeige.....	9
Bargraph Anzeige (Balkenanzeige).....	11
7-Segment-Anzeige.....	12
Taster als Eingabegerät.....	15
Wie rechnet eigentlich ein Mikrocomputer?.....	19
Befehlsliste.....	22



Arduino Projekt

→ Mikrocontroller Einführung und Start

Name

Blatt
1/23

Was ist ein Mikrokontroller?

Das sind programmierbarer Chips, die wie ein Minicomputer (Prozessoren) arbeiten und heute in allen Handys, Taschenrechnern, Waschmaschinen, Fahrkartenautomaten, aber auch in Autos und Maschinen etc. eingebaut sind. In diesen Geräten wird ein Mikrokontroller programmiert, um z.B. zu erkennen, wann eine Taste gedrückt wird, oder um eine Digitalanzeige anzusteuern. Mikrokontroller werden dafür programmiert, Sensordaten zu lesen, die Daten zu verarbeiten, um dann Entscheidungen zu treffen, indem Texte auf Displays auszugeben werden. Ebenso kann aber auch eine LED oder ein Motor angesteuert werden. Wie das geht werden wir in den nächsten Wochen lernen. Als Hardware werden wir dazu das Open Source Projekt *Arduino* benutzen, für das es je nach Anwendung weitere Aufbaumodule gibt.

Hardware und Software

Mit der Programmierung eines Arduino-Projekts anzufangen ist ähnlich, als wenn Du einen neuen PC oder Laptop in Betrieb nehmen willst. Das erste, was die meisten Leute tun, wenn sie einen neuen Computer haben, ist ihn auszupacken, einzustecken, die Software zu installieren und auszuprobieren und vielleicht sogar ein paar eigene Programmzeilen in einer Programmiersprache zu schreiben. Wenn Du den Arduino zum ersten Mal einsetzt, wirst Du etwa dieselben Dinge tun. Im Folgenden wird Dir gezeigt, wie Du ein Arduino zum Laufen bringst:

- Installieren der Programmierungs – Software (**BASCOM**) und der Software für den Programmieradapter,
- Anschluss des Arduino Moduls an den PC für die Programmierung,
- Schreiben eines ersten BASIC Programms,
- Anschluss des Arduino an die (Batterie-) Stromversorgung,
- Start des ersten Programms.

Das Seeeduino board

Das Seeeduino-Board ist vergleichbar mit einem Motherboard bei einem PC (s. Abb. 1.1).

- Auf ihm sitzt der Prozessor, eine kleiner unabhängiger Rechner, der programmiert werden kann **[1]**.
- Dieses geschieht über einen Programmieradapter der am ICSP-Stecker **[2]** angeschlossen wird.
- Weiterhin besitzt der Seeeduino einen USB Anschluss **[3]**, mit dem er Daten mit einem PC oder Notebook austauschen kann.
- Die Verbindung zur "Außenwelt" funktioniert über die Steckerleisten **[4]** und **[5]**, indem einzelne Ports abgefragt oder angesteuert werden. Das geschieht z.B. indem man weitere Shields (Boards) z.B. für ein Display oder eine Tastatur auf das Motherboard aufsteckt.

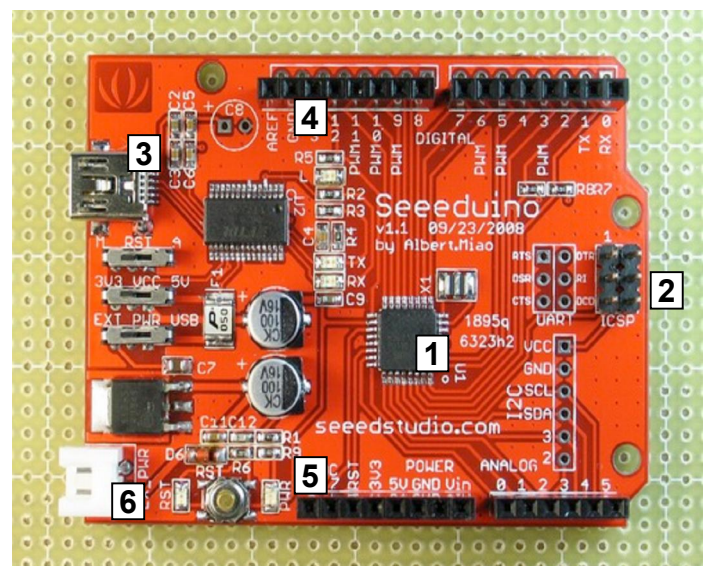


Abb. 1.1: Das Seeeduino Board

- Am Anschluss **[6]** kann eine externe Stromversorgung wie eine Batterie angeschlossen werden, damit das Arduino-Projekt eigenständig ohne PC läuft

Die Programmiersoftware BASCOM

BASCOM ist die Software mit dem Dein Arduino-Projekt in der Programmiersprache *Basic* programmiert wird (Alternative Sprachen sind C++, Assembler, etc.). Diese Software ermöglicht es Dir, auf Deinem Computer Programme zu schreiben und diese in den Mikrokontroller auf den Seeduino zu laden. Dieses Programm ist bereits auf den Rechnern installiert oder kann von folgender Website als Demo heruntergeladen werden ➔ <http://www.mcselec.com>



Bevor aber ein eigenes Projekt gestartet wird, müssen einige Einstellungen vorgenommen werden.

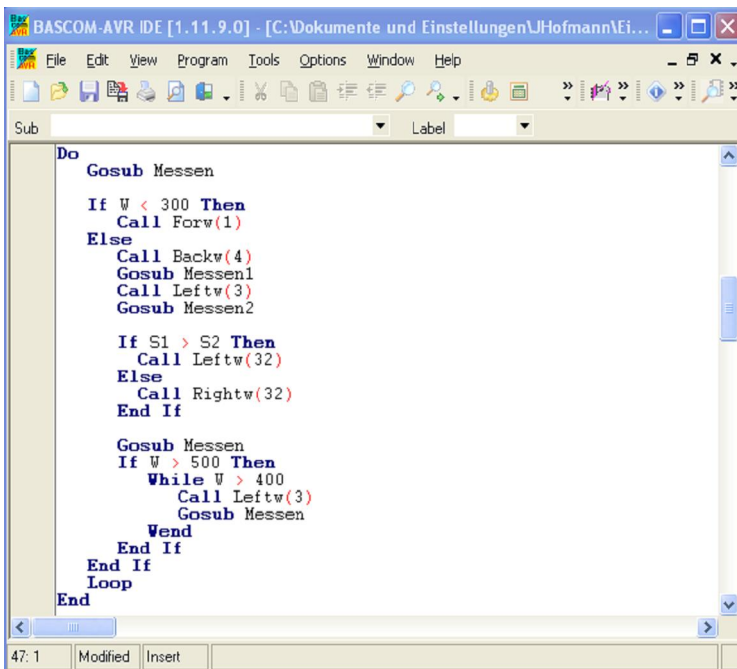


Abb. 1.2: Bascom Programmeditor

Dazu auf ➔ Options ➔ Compiler

➔ Chip

Zuerst muss dem Programm erklärt werden, welcher Chip programmiert werden soll (s. Abb. 1.2).

Bei einem Seeduino ist es der Prozessor **M 328** (m328pdef)

➔ LCD

Ebenfalls kennt der Controller noch nicht, welches Display angeschlossen ist und welche Ports (EN, RS, DB4-DB7) dazu verwendet werden. Außerdem muss der LCD-Typ eingestellt werden. (Anzahl Zeichen und Zeilen)



Alle Einstellungen müssen nach den Vorgaben der Abbildungen eingestellt werden

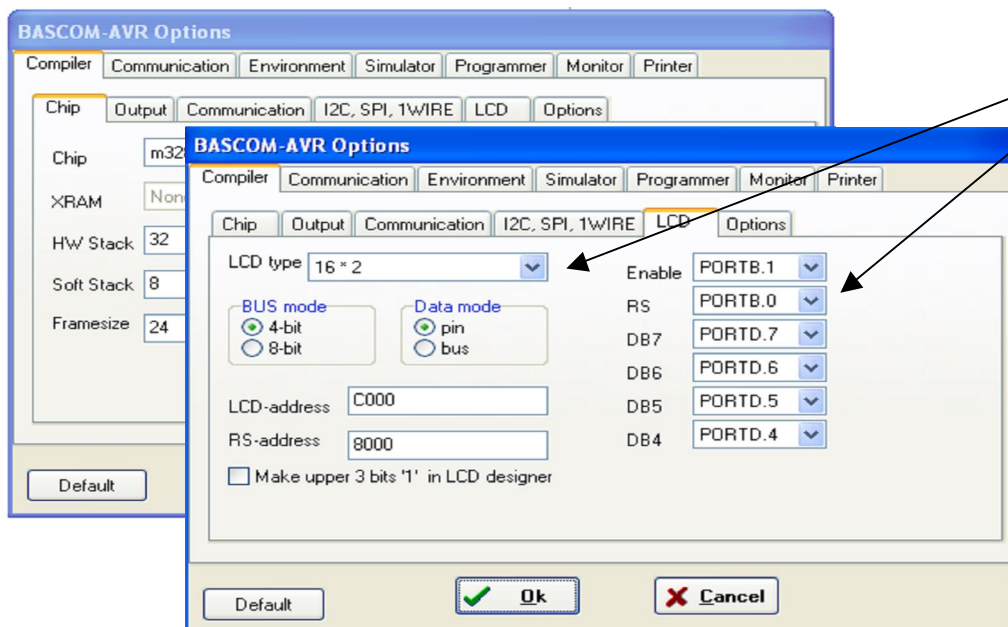



Abb. 1.3: Einstellung von Bascom AVR Options

	Arduino Projekt ➔ Mikrocontroller Einführung und Start	Name
		Blatt 3/23

Programmieradapter

Um den Mikrocontroller zu programmieren wird ein Programmieradapter benötigt, der das Programm auf den Mikrocontroller "brennt". Die Treiber für den Programmieradapter werden nach Herstellerangaben installiert.

Damit BASCOM nun weiß, welcher Programmieradapter verwendet wird, müssen auch hier Einstellungen vorgenommen werden.

Dazu wird als Programmer der STK500 gewählt, wobei die Software ebenfalls noch installiert werden muss. Diese findet man in Google, wenn man unter "stk500.exe" sucht und sich das Programm z.B. bei *AVR Freaks* herunterlädt und installiert.

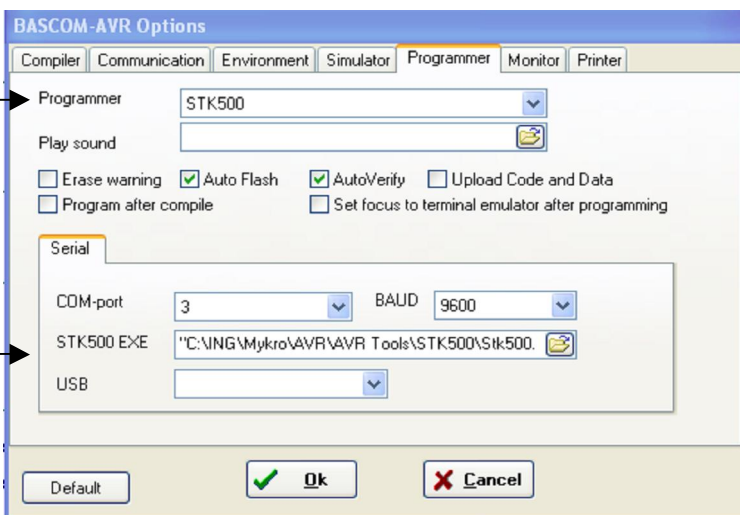


Abb. 1.4: Auswahl des Programmieradapters

Danach muss angegeben werden, an welcher Stelle die Software für den Stk500 installiert wurde.

Zum Schluss muss der Programmer nur noch über den USB Anschluss mit dem PC oder Notebook und dem Seeeduino verbunden werden.



Abb. 1.5: Anschluss des Programmieradapters

So jetzt hast Du eine Menge über die Installation der Programme und verschiedene Softwareeinstellungen, die notwendig sind, erfahren. Klingt kompliziert, ist aber einfacher als Du Dir das jetzt vielleicht vorstellst.

Im Übrigen habe ich für Euch schon sämtliche notwendigen Programme installiert. Falls du aber auch zu Hause mal Programme schreiben willst, dann könnten diese Informationen sehr nützlich sein.

Viel Spass beim Programmieren !!!



Arduino Projekt

→ Das erste Testprogramm

Name

Blatt
4/23

Testprogramm "Hallo World"

Das erste Programm, das wir schreiben und auf den Mikrokontroller übertragen werden, soll uns zeigen, ob der Mikrokontroller einsatzbereit ist und ob die Übertragung durch den Programmieradapter funktioniert. Dazu nutzen wir eine winzig kleine LED, die auf dem Seeduino zu finden ist, die wir mit dem Mikrokontroller ansteuern (s. Abb. 2.1).

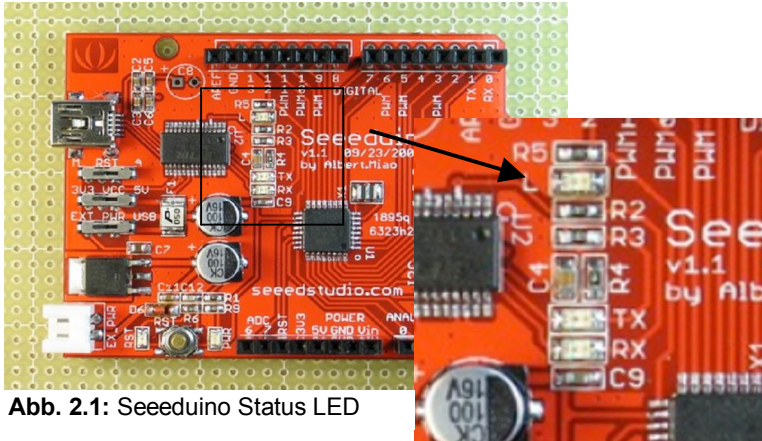


Abb. 2.1: Seeduino Status LED

Dazu müssen man ein kleines Programm schreiben, dass aus einzelnen Befehlen besteht, die der Mikrokontroller versteht. Dazu verwendet man die Programmiersprache *BASIC*, eine von mehreren Sprachen, die der Mikrokontroller verstehen kann.

Ganz praktisch bei unserem kleinen Testprogramm ist, dass wir auch schon ein paar ganz einfache Befehle kennenlernen (s. Abb. 2.2).



Zuerst schreibt man ein Programm in einem sogenannten **Programmeditor**



Wenn das Programm fertig geschrieben ist, wird es von einem **Compiler** übersetzt, damit es der Microprozessor auch verstehen kann. (Hier Button drücken)



Damit das Programm auf den Mikrokontroller übertragen wird, muss dieser Button abschließend gedrückt werden.

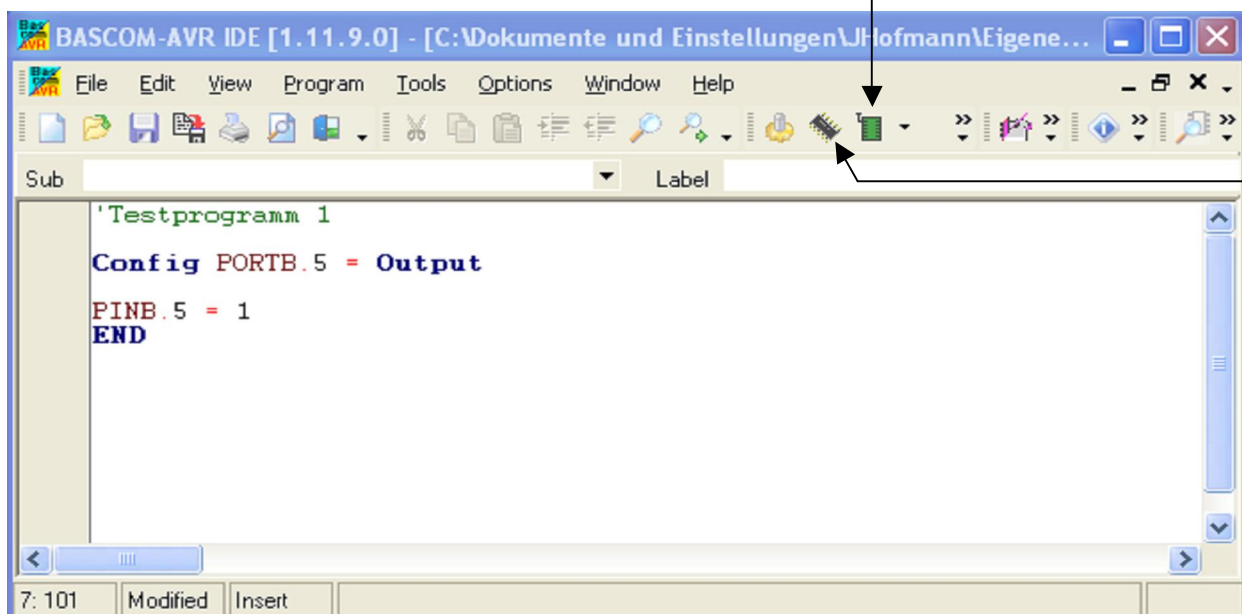


Abb. 2.2: Schreiben eines Testprogramms "Hallo World"

Los geht es !!!



Arduino Projekt

→ Das erste Testprogramm

Name

Blatt
5/23

Aufgabe



Schreibe folgendes Programm

Programm 1.a

```
Config Portb.5 = Output  
Pinb.5 = 1  
END
```



Was beobachtest du auf dem Seeduino?



Erweitere das Programm um zusätzliche Befehle

Programm 1.b

```
Config Portb.5 = Output  
Pinb.5 = 0  
Waitms 2000  
Pinb.5 = 1  
END
```



Was beobachtest du nun auf dem Seeduino?



Was könnte der Befehl **Waitms** bewirken und welche Bedeutung hat die **Zahl** dahinter?



Erweitere das Programm und rücke die Befehle zwischen **DO** und **LOOP** mit der Tabulatortaste ein.

Programm 1.c

```
Config Portb.5 = Output  
DO  
  Waitms 1000  
  Pinb.5 = 1  
  Waitms 1000  
  Pinb.5 = 0  
LOOP  
END
```



Was geschieht nun mit der LED auf dem Seeduino?



Was bewirkt der Befehl **DO** und **LOOP** und wann kann man ihn beim Programmieren einsetzen?



Jetzt kannst Du selber ein wenig experimentieren, indem Du in den Programmzeilen neue Werte ausprobierst.



Was hast Du rausgefunden?

 Arduino Projekt → Befehlsliste	Name
	Blatt 6/23

Befehlsliste in Bascom

Um einen Microprozessor zu programmieren, hat Bascom mittlerweile über 80 Befehle. Natürlich können hier nur eine Auswahl von Befehlen behandelt werden. Sämtliche Befehle (statements) sind aber in der Hilfe von Bascom zu finden und werden in englischer Sprache erklärt.



Zu finden unter: Help → Index → Index (hier gesuchten Befehl eingeben)

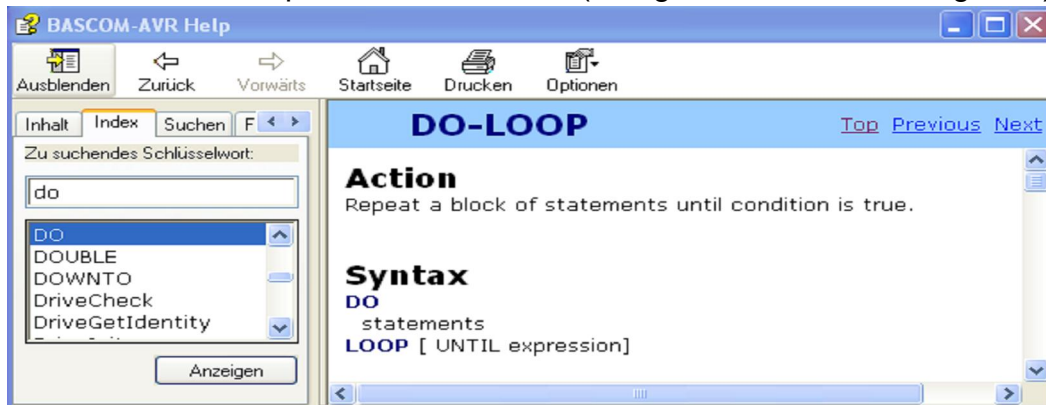


Abb. 3.1: BASCOM Befehlsliste



Es ist sinnvoll sich für jeden Befehl (statement) sein eigene Befehlsliste anzulegen, um sich so sein eigenes Nachschlagewerk zu konstruieren.

Als Beispiel wird dieses hier einmal an zwei Befehlen, die Ihr schon kennen gelernt habt gezeigt.


	DO-LOOP
	<p>Action Wiederholung von Befehlen in einer Schleife angefangen von DO bis LOOP</p> <p>Syntax DO statement 1 statement 2 ... LOOP</p>

Abb. 3.2: Befehlssyntax Do-Loop


	Waitms x
	<p>Action Warten von x Millisekunden bis der nächste Befehl ausgeführt wird.</p> <p>Syntax Waitms 1000</p> <p>Verwandte Befehle Wait, Waitus, Delay</p>

Abb. 3.3: Befehlssyntax Waitms



Die Anschlüsse des Mikroprozessors

Ein Mikroprozessor hat Anschlüsse, mit denen er entweder

- Dinge wie eine LED **steuern** oder
- Informationen wie z.B. von einem Temperatursensor **empfangen** kann.

Einen einzelnen Anschluss nennt man bei einem Mikroprozessor **Pin**. In der Abbildung rechts erkennst Du Bezeichnungen wie PB0 bis PB5, PC0 bis PC5 und PD0 bis PD7. Der zweite Buchstabe bei der Bezeichnung (B,C oder D) gibt an, welchem **Port** der Anschluss angehört.

- Bei einem Port werden nämlich mehrer Pins zusammengefasst und durchnummeriert.

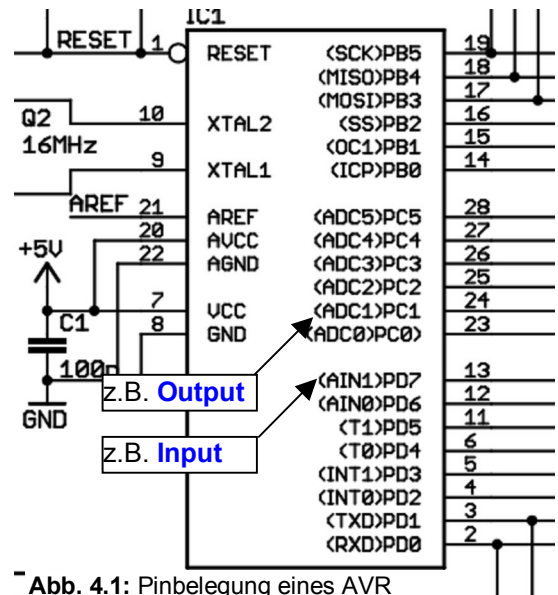


Abb. 4.1: Pinbelegung eines AVR ATMeaa 8

Je nachdem, ob ein ganzer Port oder nur ein Pin Eingang oder Ausgang sein soll, muss er im Programm vorher festgelegt werden.



Config Portx.y = [Output] oder [Input]

Mit dem Befehl **Config** wird bei der Programmierung immer etwas festgelegt (konfiguriert). In diesem Fall wird dem Prozessor mitgeteilt, ob einer seiner Anschlüsse ein **Output** (Ausgang) oder ein **Input** (Eingang) ist.

Der Anschlussname ist immer eine Adresse in der Form Pinx.y, wobei

- x den Portnamen angibt (B,C oder D) und
- y die Zugehörige Nummer des Pin (1,2,3 ...)

Beispiel:

```

Config Portc.0 = Output      ' Pinc.0 ist ein Output
Config Portd.6 = Input       ' Pind.6 ist ein Output

```

Abb. 4.2: Konfiguration eines Ports

Befehle:



Im BASCOM-Programmeditor werden **Befehle** immer automatisch **blau** markiert.

Kommentare:



Neben dem eigentlichen Programm, bei dem man sich ganz genau an die Schreibweise halten muss, damit es vom Prozessor "verstanden" wird, kann man auch eigene Kommentare einfügen. Diese müssen aber durch ein " ' " vor dem Kommentar gekennzeichnet werden.



Arduino Projekt

→ Anschlüsse des Mikroprozessors (Ports und Pins)

Name

Blatt
8/23

Schnittstelle zur Außenwelt

Die Ports des Mikrokontroller sind die Schnittstelle zur Außenwelt und werden auf dem Seeeduino an drei Steckerleisten herausgeführt (s. Abb. 4.3). Zusätzlich gibt es noch eine Steckerleiste für die Spannungsversorgung (Power).

Die Entwickler dieses Mikrokontroller Projekts (Arduino Projekt) haben genau festgelegt, an welcher Stelle sich die Steckerleisten befinden, so dass auf die Steckerleisten weitere Platinen (Shields) aufgesteckt werden können.



Abb. 4.3: Ports des Seeeduino

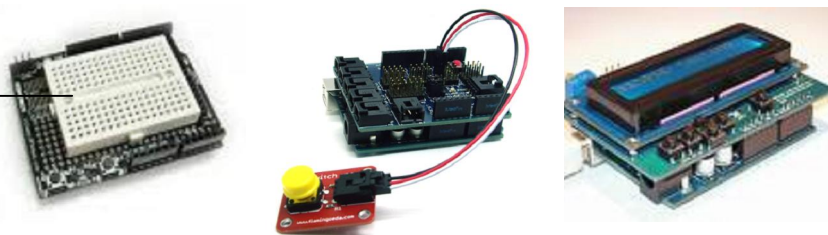


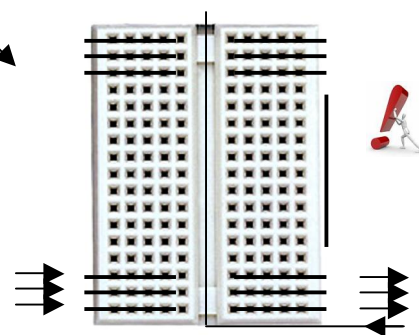
Abb. 4.4: Prototype-Shield, Sensor-Shield, LCD-Shield



Beim Aufstecken der Shields muss man unbedingt darauf achten, dass die Stecker nicht verbogen werden!

Das Prototype-Shield

Für eigenen Entwicklungen ist das sogenannte Prototype-Shield sehr interessant. Hier kann man eigene Schaltungen auf einem kleinen Steckbrett entwickeln und über die Ports mit Hilfe von kleinen Kabeln direkt mit dem Mikrokontroller verbinden.



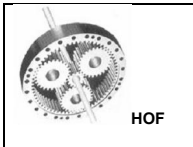
Damit das Verbinden der elektronischen Bauelemente sehr einfach geht, sind jeweils die fünf Buchsen in einer Zeile - sowohl auf der rechten als auch auf der linken Seite - miteinander verbunden.

verbunden
hier getrennt

Abb. 4.5: Verdrahtung des Steckbrettes



Jede aufgebaute Schaltung wird vor der Inbetriebnahme vom Lehrer überprüft!



LEDs als Statusanzeige

Leuchtdioden (s. Abb. 4.6) werden in vielen Geräten der Unterhaltungselektronik eingesetzt. Ihr Aufleuchten zeigt dabei den Betrieb oder die Betriebsbereitschaft des jeweiligen Gerätes an. Der erste Teil ihres Namens ist damit klar : sie "leuchten" - geben Licht ab.

Leuchtdioden sind ebenfalls Dioden wie andere Halbleiter- oder Röhrendioden auch. Dioden sind so etwas wie ein "Ventil" für Elektronen: so wie das Ventil an einem Fahrradreifen, die Luftmoleküle nur hinein- aber nicht wieder herauslassen, lassen auch Dioden die Elektronen nur in einer Richtung passieren.

Aus diesem Grund muss man den Aufbau einer LED kennen, um diese richtig in die Schaltung einzusetzen. In Abb. 4.7. ist eine typische Standard-LED mit ihren Anschlussmerkmalen dargestellt.



Der **lange** Anschlussdraht ist die **Anode** und zeigt zum **Port** des Mikrokontrollers.

Der **kurze** Anschlussdraht ist die **Kathode** und wird mit **GND** verbunden.

LEDs werden immer mit einem Vorwiderstand betrieben (s. Abb. 4.8), da die Versorgungsspannung fast immer über der LED-Spannung liegt, was ohne Vorwiderstand zur Folge hat, dass ein zu hoher Strom fließt, der die LED zerstören würde.



LEDs dürfen nie ohne Widerstand betrieben werden, da sonst die **LED zerstört** werden könnte!

Als Vorwiderstand für eine **LED an einem Mikrokontroller** (s. Abb. 4.9) wählt man einen Wert zwischen $100\ \Omega$ und $470\ \Omega$, um den **Mikrokontroller nicht zu zerstören**.



Abb. 4.6: Versch. LED -Typen

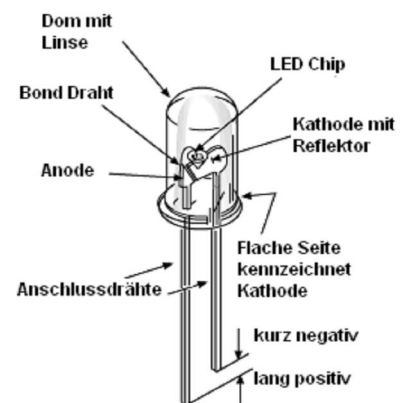


Abb. 4.7: Aufbau einer LED

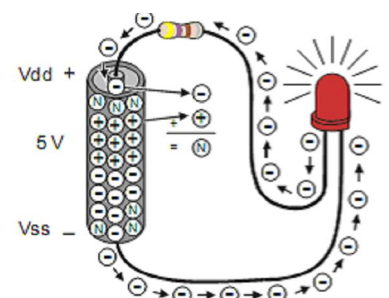


Abb. 4.8: LED mit Vorwiderstand an einer Batterie

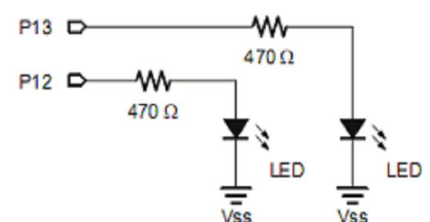


Abb. 4.9: LED an einem Port (Schaltplan)



Arduino Projekt

→ Anschlüsse des Mikroprozessors (Ports und Pins)

Name

Blatt
10/23

Aufgabe



Baue die LED Schaltung nach Vorgabe des Fotos auf!

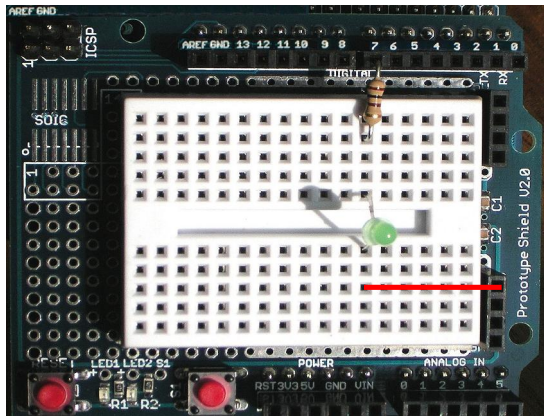


Abb. 4.10: Aufbau einer LED-Schaltung

Material:

- 1x LED
- 1x Widerstand 100 Ω
- 1x Kabel

Verwendete Pins:

Port D: D7



LEDs dürfen nie ohne Widerstand betrieben werden, da sonst der **Mikrokontroller zerstört** wird!



Schreibe das folgende Programm!

Programm 2.a

```

Config Portd.7 = Output
Portd.7 = 1
END

```



Was beobachtest du?



Schließe eine weiteren LED an einen anderen Pin an und erweitere das Programm so, dass beide LEDs leuchten!

Programm 2.b

```

Config Pind.7 = Output
...
Portd.7 = 1
...
END

```



Was beobachtest du?



Erweitere das Programm so, dass die LEDs abwechselnd blinken!

Programm 2.c

```

Config Portd.7 = Output
...
DO
...
LOOP
END

```



Zähle auf, welche Befehle du verwendet hast?



Arduino Projekt

→ Anschlüsse des Mikroprozessors (Ports und Pins)

Name

Blatt
11/23

Bargraph Anzeige (Balkenanzeige)

Eine Bargraph-Anzeige ist ein Anzeigeelement, bei dem sich die Länge eines leuchtenden Balkens (engl.: *bar* für Stange) mit der Signalgröße verändert. In der Tontechnik z.B. benutzt man eine Bargraph-Anzeige, um den Lautstärkepegel von Boxen anzuzeigen.

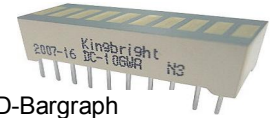


Abb. 4.11: LED-Bargraph

Aufgabe



Baue die Bargraph Schaltung nach Vorgabe des Fotos auf!!

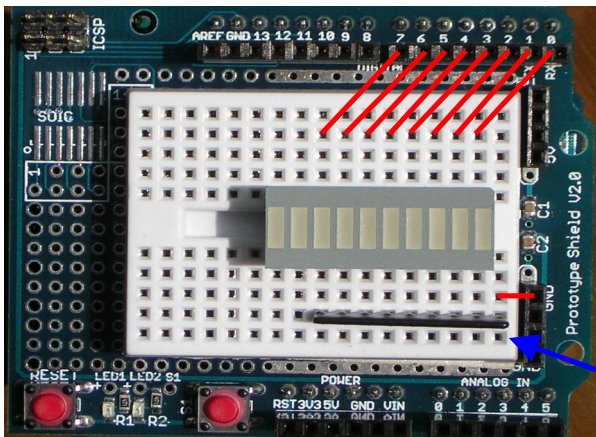


Abb. 4.12: Aufbau einer Bargraph-Schaltung

Material:

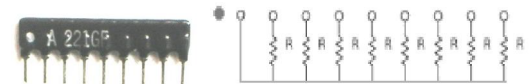
- 1x Bargraph Anzeige
- 1x Widerstands-Array 100 Ω
- 9x Kabel

Verwendete Pins:

Port D: D0, D1, D2, D3, D4, D5, , D6, D7



Der weiße Punkt muss auf die rechte Seite!



Schreibe das folgende Programm!

Programm 3.a

```
Config portd = Output
```

```
Do
```

```
Portd = &B00000001
```

```
Waitms 500
```

```
Portd = &B00000010
```

```
Waitms 500
```

```
Portd = &B00000100
```

```
Waitms 500
```

```
LOOP
```

```
END
```



Was beobachtest du?



Was bewirkt der Befehl

Portd = &B00000001 ?

Trage den Befehl in deine Befehlsliste ein!



Kann man auch zwei LED gleichzeitig ansteuern? (Erklärung)



Erweitere das Programm so, dass die LED einmal von rechts nach links und wieder zurück läuft!

Programm 3.b

```
Config portd = Output
```

```
Do
```

```
...
```

```
LOOP
```

```
END
```



Fallen dir noch andere Muster ein, die du programmieren

kannst



7-Segment-Anzeige

Eine Segmentanzeige kann durch eine elektronische Ansteuerung aus einzelnen oder mehreren Segmenten („Strichen“) Buchstaben, Zahlen oder Zeichen darstellen.

Die bekannteste ist die 7-Segment-Anzeige, welche z.B. in Digitaluhren zum Einsatz kommt und alle Ziffern von 0-9 aus bis zu sieben einzelnen Segmenten zusammensetzt.

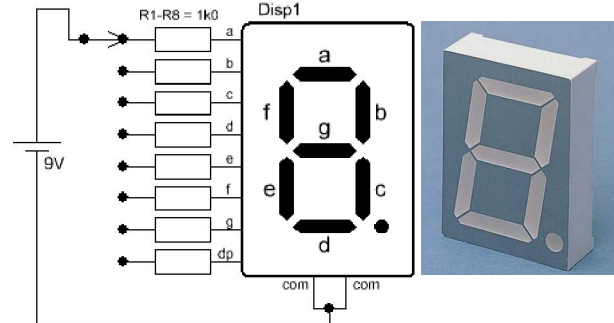


Abb. 4.13: 7Segment-Anzeige

Aufgabe



Erstelle eine Tabelle, welche Segmente für jede Zahl von 0 bis 9 angesteuert werden müssen!

Zahl	Benötigte Segmente	Programmierung
0	a, b, c, d, e, f	Portd = &B00111111
1	b, c	Portd = &B00000110
2	a, b, d, e, g	Portd = &B01011011
3	a, b, c, d, g	Portd = &B01001111
4	b, c, f, g,	Portd = &B01100110
5	a, c, d, e, f	Portd = &B01101101
6	c, d, e, f, g,	Portd = &B01111100
7	a, b, c,	Portd = &B00000111
8	a, b, c, d, e, f, g	Portd = &B01111111
9	a, b, c, d, f, g	Portd = &B01101111



Kann man auch Buchstaben darstellen? Wenn ja, welche?



Aufgabe



Baue die 7-Segment-Schaltung nach Vorgabe des Fotos auf!

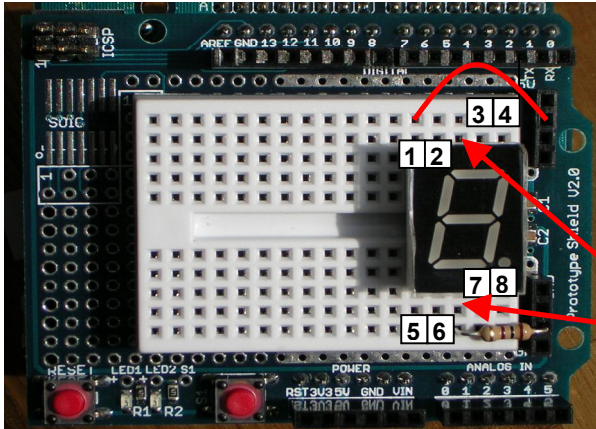


Abb. 4.14: Aufbau einer 7-Segment-Schaltung

Material:

- 1x 7-Segment-Anzeige
- 1x Widerstand 100 Ω
- 1x Kabel

Verwendete Pins:

Port D: D0, D1, D2, D3, D4, D5, D6, D7



Hier darf auf gar keinen Fall eine Leitung angeschlossen werden, da es sonst einen Kurzschluss gibt und das Board zerstört wird!

Leider steht uns für diese Anzeige kein Datenblatt zur Verfügung, so dass man erst einmal selber herausfinden muss, an welchen Anschlüssen die jeweiligen Segmente angesteuert werden.



Nehme das Kabel wie in der Abbildung oben und probiere nacheinander die nächsten Anschlüsse (2, 3, 4, ..., 8) aus und notiere die Belegung (a, b, ..., g) in der Abbildung unten!

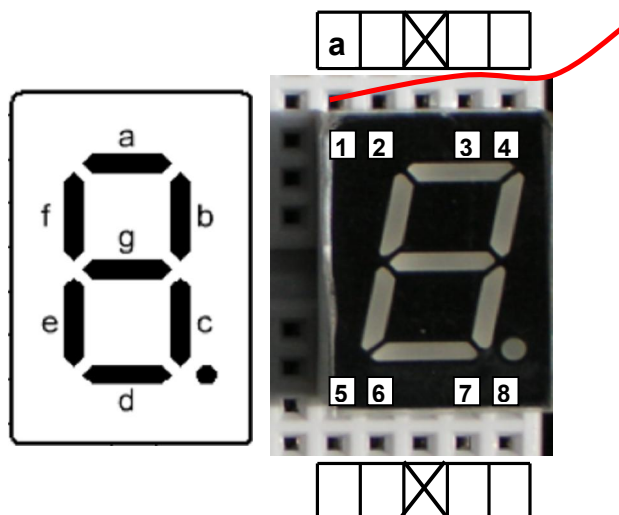


Abb. 4.15: Überprüfung der Anschlussbelegung der 7-Segment-Anzeige



Arduino Projekt

Name

→ Anschlüsse des Mikroprozessors (Ports und Pins)

Blatt
14/23



Verbinde nun die Anschlüsse der 7-Segmentanzeige mit den Ports des Mikrocontrollers nach der folgenden Zuordnung.

a → D0
b → D1
c → D2
... → ...



Schreibe ein Countdown Programm!

Programm 4.a

```
Config portd = Output  
  
Portd = Portd = &B01101111  
...  
...  
  
END
```



Arduino Projekt

→ Anschlüsse des Mikroprozessors (Ports und Pins)

Name

Blatt
15/23

Taster als Eingabegerät

Bis jetzt haben wir uns ausschließlich mit der Ausgabe an Ports beschäftigt. Ebenso wichtig ist aber auch, den Mikrokontroller von außen zu steuern oder Informationen zur Verarbeitung bereitzustellen. In der einfachsten Form geschieht dieses mit Hilfe eines Tasters.



Abb. 4.16: Verschiedene Tastertypen

Der Mikrokontroller kann an seinen Ports zwischen zwei unterschiedliche Zuständen unterscheiden:

- **High** (Spannung 5V) und
- **Low** (GND=0V)

High wird auch oft als "1" und low als "0" bezeichnet.

In der Abb. 4.16 kann man sehen, was passiert wenn der Taster gedrückt wird.

Im Ausgangszustand ist der Taster **geöffnet** und am Port PD3 liegt über einen Widerstand **GND = 0V** (low) an. Wenn der Taster **geschlossen** ist, liegt am Port eine Spannung von **5V**, also ein High-Signal an.

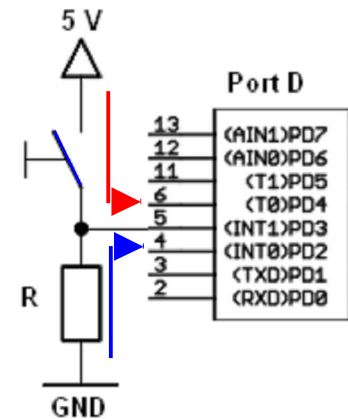


Abb. 4.17: Tasterschaltung (Schaltplan)



Was passiert, wenn man den Taster wieder loslässt? Richtig am Port liegt wieder das Low-Signal an.

Mit diesem Wissen ist es nun möglich, mit einem Taster ein Programm zu steuern. Wie das genau geht, werden wir anhand eines kleinen Programms sehen.



Baue die Tasterschaltung nach Vorgabe des Fotos auf!

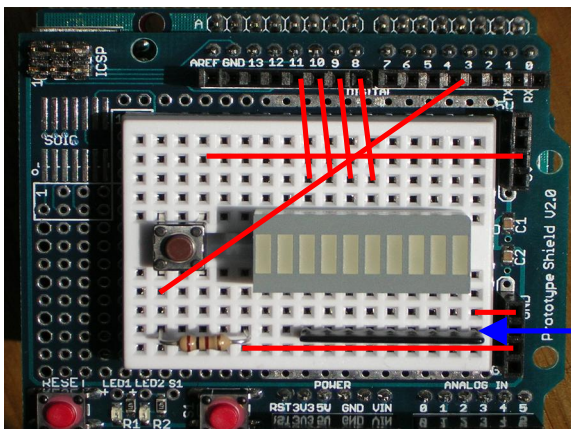


Abb. 4.18: Aufbau einer Taster-Schaltung

Material:

- 1x Bargraph-Anzeige
- 1x Widerstands-Array 100 Ω
- 7x Kabel

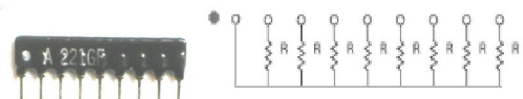
Verwendete Pins:

Port B: B0, B1, B2, B3

Port D: D3



Der weiße Punkt muss auf die rechte Seite!





Durch die Tasterschaltung wird jetzt ein High (1) oder ein Low (0) erzeugt, aber nun muss man dem Mikrokontroller beibringen, was er nun mit den Signalen machen soll. Auf jeden Fall kennt er ganz bestimmte Befehle (statements) für die Ports PD2 und PD3, die man **Interrupts** nennt, um mit dem Taster ein Programm zu steuern.



INTERRUPT aktivieren

Action

Ein INTERRUPT unterbricht sofort das Hauptprogramm und führt ein Unterprogramm aus. Dabei wird der Interrupt durch ein Signal entweder am Port PD2 (INT0) oder PD3 (INT1) z.B. durch einen Taster ausgelöst.

Um einen INTERRUPT festzulegen (konfigurieren) und zu aktivieren, sind mehrere Befehle notwendig.

Syntax

Enable Interrupts

→ hier wird die Interruptfunktion aktiviert

Config IntX = Status

→ hier wird der Trigger (Auslöser) festgelegt

X = 0 für **INT0** z.B. Taster an Port PD2 oder

X = 1 für **INT1** wenn Taster an Port PD3

Status = **Falling** für Signalwechsel von 1 → 0

Status = **Rising** für Signalwechsel von 0 → 1

Status = **Change** für Signalwechsel von 0 → 1 oder 1 → 0

Enable IntX

→ hier wird der jeweilige Interrupt **INT0** oder **INT1** aktiviert

X = 0 für **INT0**

X = 1 für **INT1**

Mit **Disable** kann man den jeweiligen Interrupt wieder abschalten

On Int0 label Nosave

→ gibt an, welches Unterprogramm (Name hier Label) bei einem Interrupt aufgerufen werden soll.

Beispiel

```
' CONFIG Interrupt -----  
Enable Interrupts  
Config Int1 = rising  
Enable Int1  
On Int1 Susi Nosave  
  
' Hauptprogramm -----  
Do  
  Pinb.1 = 0  
Loop  
End  
  
' Unterprogramm -----  
Susi:  
  Pinb.1 = 1  
Return  
' durch Return wird wieder ins Hauptprogramm gesprungen
```



Arduino Projekt

Name

→ Anschlüsse des Mikroprozessors (Ports und Pins)

Blatt
17/23



Schreibe das folgende Programm!

Programm 5.a

```
Config Portb = Output
'CONFIG Interrupts-----
Enable Interrupts      'Aktiviert die Interruptfunktion
Config Int1 = Falling  'Definiert den Trigger
Enable Int1            'Aktiviert den Interrupts Int0
On Int1 Susi Nosave    'Springt zum Unterprogramm
                       ' Susi, wenn Taste Gedrückt

'-----
'Hauptprogramm
Do
  Pinb.0 = 1
Loop
End                       'Ende Hauptprogramm Label

'-----
'Unterprogramm
Susi:                      'Beginn des Unterprogramms Susi
  Pinb.0 = 0
Return                    'Rückkehrung ins Hauptprogramm
```



Was beobachtest du?



Erkläre, was das Haupt und das Unterprogramm macht!



Was muss man ändern, dass die LED bei Tastendruck eingeschaltet wird



Was muss im Programm geändert werden, wenn der Taster an PD2 angeschlossen wird?

Dieses Programm ist total langweilig, denn so ein Aufwand für nur eine einzige Lampe anzuschalten!?!



Hast Du vielleicht eine Idee, wie man ein Programm schreiben kann, womit man seine Reaktionen testen kann? Ich mache mal den Anfang!

Programm 5.b

```
'CONFIG Interrupts-----
Enable Interrupts      'Aktiviert die Interruptfunktion
Config Int1 = Falling  'Definiert den Trigger
Enable Int1            'Aktiviert den Interrupts Int0
On Int1 Reaktion Nosav 'Springt zum Unterprogramm
                       ' Reaktion, wenn Taste gedrückt

'-----
'Hauptprogramm
Do
  Portd = &B00000001
  Waitms 100
  Portd = &B00000010
  ...
Loop
End                       'Ende Hauptprogramm Label

'-----
'Unterprogramm
Reaktion:                'Beginn des Unterprogramms
...
```



Arduino Projekt

→ Anschlüsse des Mikroprozessors (Ports und Pins)

Name

Blatt
18/23



Man kann auch mal dieses Programm ausprobieren:

Programm 5.c

```
'CONFIG Interrupts-----
Enable Interrupts      'Aktiviert die Interruptfunktion
Config Int1 = Falling  'Definiert den Trigger
Enable Int1            'Aktiviert den Interrupts Int0
On Int1 Susi Nosave    'Springt zum Unterprogramm
                       ' Susi, wenn Taste Gedrückt

'-----
'Hauptprogramm
Do
  Pinb.0 = 1
Loop
End                       'Ende Hauptprogramm Label

'-----
'Unterprogramm
Susi:                     'Beginn des Unterprogramms Susi
  Pinb.0 = 0
Return                   'Rücksprung ins Hauptprogramm
```



Was beobachtest du?

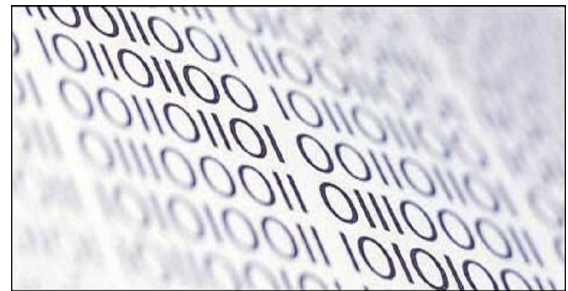


Was verursacht der Befehl shift



Wie rechnet eigentlich ein Mikrocomputer?

Bis jetzt haben wir gesehen, dass man mit dem Mikrocomputer über Ports LEDs, 7-Segmentanzeigen oder eine Bargraphanzeige ansteuern können. Das funktioniert immer über die Information Port oder Pin an (1) oder aus (0). Bei der Eingabe über Taster ist das genau so. Viele Leute sagen ein Computer wäre eigentlich dumm, denn er kann nur 1 oder 0.



Aber wie rechnet er dann? **Mit Einsen und Nullen ...?**

Genau so geht es! Die Zahlen, die der Computer kennt, sehen genau so aus, wie wir es schon einmal bei der Ansteuerung mit dem Portbefehl gesehen haben.

```
→ Portd = &B00000101
```

Für den Computer sehen unsere Dezimalzahlen wie in Tab.5.1 aus und heißen Binärzahlen.

Tab. 5.1

Dezimalzahl	Binärzahl						
	32	16	8	4	2	1	
1							1
2					1		0
3					1		1
?							
4							
5					1	0	1
?							
7							
8					1	0	0
					1	1	0
12							
?							
15							
17							
55							
103							



Eine Knobelaufgabe:
Wie werden die Binärzahlen gebildet?



Wie wird die Reihe nach 32 fortgesetzt?



Wie heißen die fehlenden Binärzahlen 4, 7, 15 und 103?

Der Computer rechnet mit Eins und Null. Das ist ziemlich einfach und geht super schnell.



Rechne doch einfach einmal, wie ein Computer und wandel zwei Zahlen in Binärzahlen um (z.B. 7 und 11)


- Addiere beide Binärzahlen schriftlich
- Wandle das Ergebnis wieder in eine Dezimalzahl um!
- Stimmt das Ergebnis?



Was mit der Addition geht, sollte auch bei der Subtraktion funktionieren.



... und was ist mit der Multiplikation?

 HOF	Arduino Projekt	Name
	→ Computer-Mathe	Blatt 20/23

Hier noch Mateaufgaben im Bascom Emulator



HOF

Arduino Projekt

→ Computer-Mathe

Name

Blatt
21/23



Befehlsliste



IF-Schleife

Action

Eine IF-Schleife prüft ob eine Bedingung erfüllt ist und führt dann die entsprechenden Befehle (statements) aus.

Syntax

```
IF Bedingung THEN  
  statement 1  
  statement 2  
  ...  
ENDIF
```