Mikrocontroller: Einführung

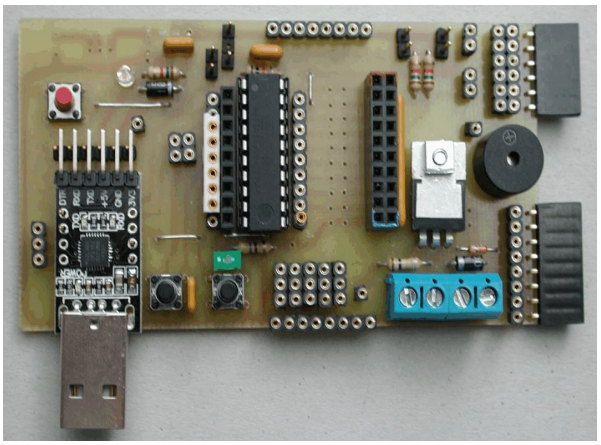

Abbildung 1

Mikrocontroller sind heute allgegenwärtig; ob Zahnbürste, Türschloss, Backofen, Fahrradcomputer, Stereoanlage, Multimeter oder Postkarte, überall sind sie zu finden. Im Prinzip handelt es sich bei Mikrocontrollern um winzige Computer. Diese Unterrichtsreihe soll zeigen, wie diese Mikrocontroller funktionieren, wozu sie eingesetzt werden können und natürlich: wie man sie programmiert.

Auf der Platine in Abb. 1 kannst Du unter Anderem sehen:

- den Mikrocontroller, das Herzstück der Platine: das ist der schwarze Baustein mit den vielen Kontakten. Er heißt Attiny2313.
- ein USB-Stecker zum Anschluss an den Computer
- einen Reset-Taster (oben links)
- zwei Taster zur Eingabe (unten links)
- einen Pieper (rechts in der Mitte)
- viele Buchsen

Mithilfe der Buchsen können zahlreiche elektrische Geräte an den Mikrocontroller angeschlossen werden. So können wir z. B. eine Leuchtdiode an den Mikrocontroller anschließen und diese blinken lassen. Über die 4-polige Anschlussleiste (unten rechts) können auch Geräte, die mehr Strom benötigen (z. B. Glühbirnen oder Motoren), gesteuert werden.

Die Anschlussstellen am Mikrocontroller werden **Ports** genannt. Diese Ports haben Namen, z. B. PortB.0, PortB.1, PortB.2,..., PortB.7, PortD.2, ...

Mikrocontroller: Einführung

Achtung: Nicht jedes elektrische Gerät kann an die Platine angeschlossen werden. Insbesondere darfst Du keine anderen elektrischen Quellen an die Platine anschließen. **Auf keinen Fall darfst Du die Platine mit dem Stromnetz verbinden! Lebensgefahr!**

Wir wollen eine Leuchtdiode (LED) blinken lassen

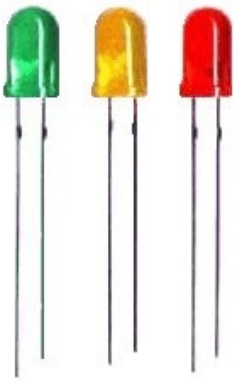

Abbildung 2

Zunächst: Was ist eine LED? Das ist ein elektronischer Baustein, der Licht aussendet, wenn Strom durch ihn fließt. Im Gegensatz zu einer Glühbirne besitzt eine LED keine Glühwendel; sie wird beim Leuchten auch nicht heiß.

In Abb. 2 erkennst Du, dass die LEDs zwei unterschiedlich lange Anschlussdrähte haben. Das ist wichtig: LEDs lassen den Strom nämlich nur in einer Richtung hindurchfließen. Ist die Polung falsch, dann kann kein Strom fließen und die LED leuchtet in diesem Fall auch nicht.

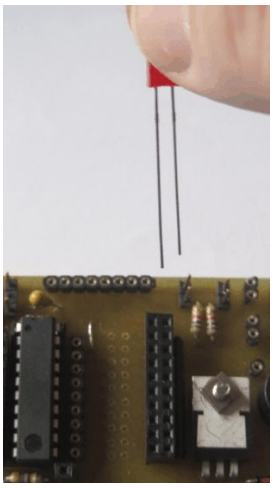

Abbildung 3

Zum Anschluss von LEDs an den Mikrocontroller ist die doppelreihige Buchsenleiste vorgesehen. Hier kannst Du bis zu 8 LEDs an die Ports PortB.0 ... PortB.7 anschließen. Damit die Polung korrekt ist, muss das lange Anschlussbein immer nach links zum Mikrocontroller hinweisen, vgl. Abb. 3.

Wir schließen eine rote LED nun an PortB.0 an. Die zugehörigen Anschlussstellen sind im Lageplan von Abb. 4 markiert.

Jetzt verbinden wir die Platine über das USB-Kabel mit dem PC; dadurch wird die Platine mit Strom versorgt: Die Kontroll-LED neben dem Reset-Taster sollte nun aufleuchten.

Aber die rote LED blinkt nicht. Das kann sie auch nicht, denn der Mikrocontroller

Mikrocontroller: Einführung

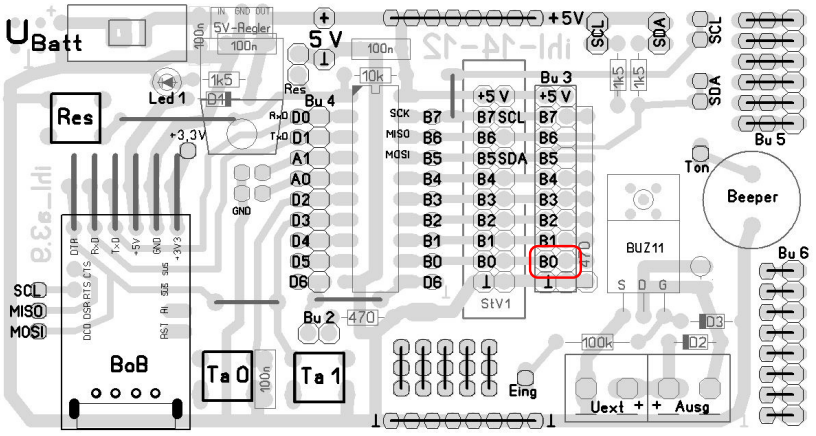

Abbildung 4

hat von uns noch keine entsprechenden Befehle erhalten. Wir müssen ihn noch programmieren. Deswegen schalten wir die Platine zunächst wieder aus!

Was ist ein Programm? Das ist nichts anderes als eine Folge von Anweisungen (Befehlen). In unserem Fall müssen diese Anweisungen dafür sorgen, dass der Mikrocontroller über den PortB.0 die LED mit Strom versorgt. Allerdings darf der Strom nicht immer fließen; vielmehr soll er fortwährend ein- und ausgeschaltet werden, damit die LED blinkt.

Diese Befehlsfolge erstellen wir am PC mithilfe des Programms BASCOM. Dazu öffnen wir die Datei "vorlage.bas". In dieser Datei stehen schon bestimmte Befehle, die der Mikrocontroller immer braucht. Damit wir sie nicht immer wieder neu schreiben müssen, greifen wir lieber auf die Vorlagedatei zurück. Bevor wir sie nun ergänzen, speichern wir sie unter dem Namen blinken.bas in einen neuen Ordner "blinken" ab.

Um Strom bei PortB.0 fließen zu lassen, benutzen wir den Befehl

PortB.0 = 1

Die 1 bedeutet hier also "LED eingeschaltet". Um die LED auszuschalten, benutzt man den Befehl

```
PortB.0 = 0
```

Soll der Mikrocontroller die LED immer wieder ein- und auschalten, könnte man die Befehle PortB.0 = 1 und PortB.0 = 0 immer wieder untereinander schreiben. Das wäre sehr lästig (und hätte auch andere Nachteile, die wir später noch behandeln).

Stattdessen schreibt man:

```
Do
        PortB.0 = 1
        wait 1
        PortB.0 = 0
        wait 1
Loop
```

Zusätzlich haben wir noch den Befehl "wait 1" verwendet; er sorgt dafür, dass nach dem Ein- und Ausschalten der LED der Mikrocontroller jeweils eine Pause von 1 Sekunde einlegt.

Der Mikrocontroller wird nun die Befehle zwischen "Do" und "Loop" endlos wiederholen. Man spricht hier von einer **Endlosschleife** (loop = Schleife).

Merke: Die beiden Befehle do und loop haben eine bestimmte Funktion: Sie sorgen dafür, dass alle Befehle, die zwischen ihnen stehen, fortwährend der Reihe nach wiederholt werden.

Wir speichern nun das Programm ab, indem wir die Schaltfläche  betätigen.

Es gibt übrigens viele unterschiedliche Programmiersprachen. Die Programmiersprache, die hier gerade benutzt wird, wird als **BASIC** bezeichnet.

Jetzt muss diese Programm in den Mikrocontroller geladen werden. Man nennt diesen Vorgang "**Upload**", zu deutsch "**Hochladen**". Leider versteht der Mikrocontroller unsere Programmiersprache BASIC nicht direkt; es muss deswegen zuvor noch in eine Sprache übersetzt werden, die der Mikrocontroller verstehen kann.

Mikrocontroller: Einführung

Diesen Übersetzungsvorgang nennt man **compilieren**. Das Compilieren erledigt BASCOM für uns mit einem Knopfdruck: Wir klicken dazu einfach die Schaltfläche  an.

Nachdem die Attiny-Platine mit dem USB-Kabel mit dem PC verbunden worden ist, klicken wir im BASCOM-Programm auf die Upload-Schaltfläche  . Nun erscheint ein neues Fenster mit dem Upload-Programm:

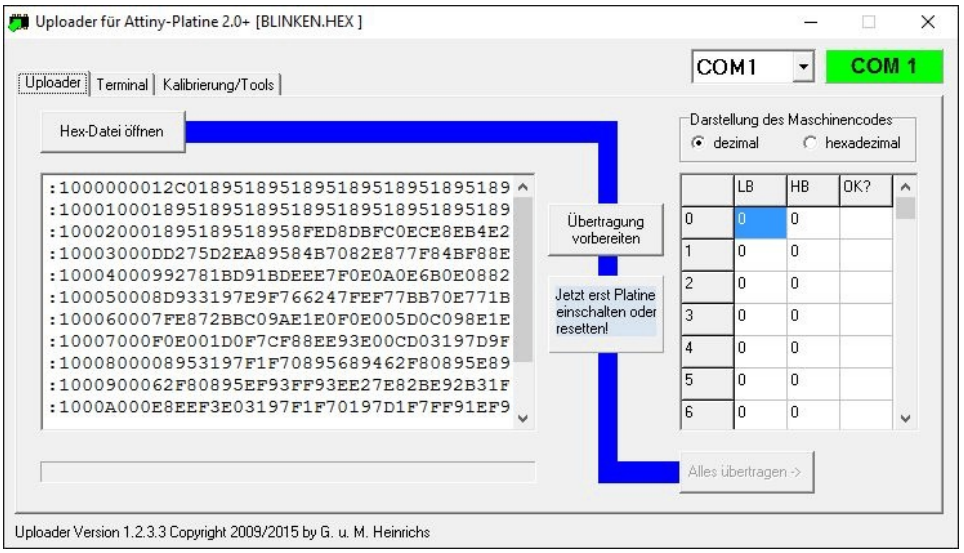

Abbildung 5

Im linken Teilbereich siehst Du übrigens das Ergebnis des Übersetzens: ein Salat aus Ziffern und Buchstaben. Diese Folge von Buchstaben und Ziffern bilden den so genannten Maschinencode. Der Maschinencode ist genau die Sprache, die der Mikrocontroller versteht; wir tun uns damit wohl etwas schwerer!

Nun kommen die entscheidenden Schritte:

1. Schaltfläche "Übertragen vorbereiten" anklicken
2. Reset-Taster (kurz) betätigen
3. Schaltfläche "Alles übertragen" anklicken

Man sieht nun, wie ein Fortschrittsbalken den Vorgang des Uploads anzeigt. Und wenn wir keinen Fehler gemacht haben, dann sollte die LED auf unserer Platine jetzt blinken!!!

Mikrocontroller: Einführung

Aufgaben:

1. Warum darfst Du die Attiny-Platine nicht an das Stromnetz anschließen?

2. Was beobachtet man, wenn bei unserem Blink-Programm keine wait-Befehle vorhanden sind?

3. Die rote LED soll nun jeweils für 2 s leuchten und für 5 s aus sein. Wie muss das Programm nun lauten?

4. Nun soll eine gelbe LED bei PortB.3 angeschlossen werden.

4.1 Wo muss sie in die Buchsenleiste gesteckt werden? Markiere die Stelle mit gelber Farbe in Abb. 4.

4.2 Nur die gelbe LED soll im Sekundenrhythmus an- und ausgehen. Schreibe das zugehörige Programm auf.

4.3 Jetzt sollen die rote und die gelbe LED abwechselnd blinken. Wie sieht das Programm nun aus?

5. Recherchiere: Wofür steht die Abkürzung LED? Was bedeutet das?

6. Was ist beim Anschließen von LEDs zu beachten?

7. Was versteht man unter einem Programm?

8. Wozu dient der Compiliervorgang?

9. Nenne eine weitere Programmiersprache. Forsche gegebenenfalls in einem Lexikon bzw. im Internet nach.

10. Wie nennt man die Anschlüsse beim Mikrocontroller, an die elektrische Geräte angeschlossen werden können?

11. Wie nennt man das Übertragen eines Programms vom PC auf den Mikrocontroller?

12. Wofür steht die Abkürzung BASCOM? Stelle eine Vermutung auf oder recherchiere.