Wir steuern Anlagen

Sicherlich möchtest du am liebsten selbst einen großen Roboter programmieren. Leider gibt es nur wenige Schulen, die derartige Geräte zur Verfügung haben. Du kannst dir aber für wenige Euro Bauteile besorgen, mit denen du selbst – in der Schule oder auch zuhause – zahlreiche interessante Versuche zum Steuern, Messen und zur Datenübertragung durchführen kannst, z.B. eine Fußgängerampel, einen Lügendetektor oder eine Anlage, welche Texte über Glasfasern überträgt.

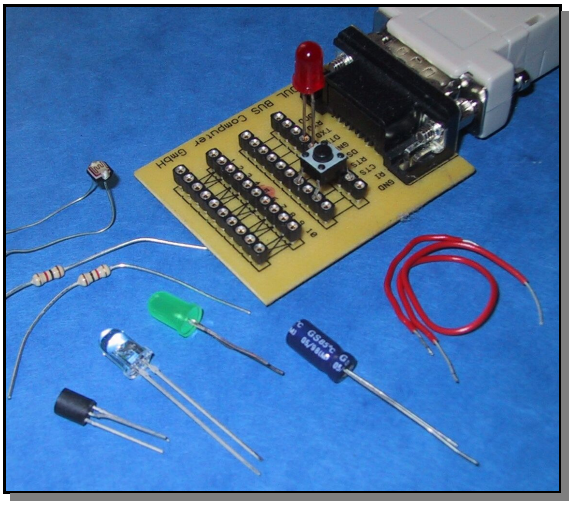

Abb. 1: Unser Experimentierset

Manchmal kann es gefährlich sein, wenn du selbst gebastelte Geräte an den PC anschließt: Die Bauteile oder sogar der PC können Schaden nehmen. Bei den hier vorgestellten Versuchen kann dies nicht geschehen, solange du dich an diese Anweisungen hältst:

> Benutze für unsere Versuche nur die serielle Schnittstelle. Benutze keine elektrischen Quellen wie Akkus, Batterien oder Netzgeräte, erst recht nicht die Steckdose!

In der Abb. 1 ist das Experimentierset dargestellt, welches in diesem und den beiden folgenden Kapiteln zum Einsatz kommen soll[1]. Einige der Bauteile kennst du vielleicht schon aus dem Physikunterricht, z.B. die Leuchtdiode oder den Schalter. Diese Bauteile werden je nach Bedarf in die Kontakte der Platine gesteckt; über das Kabel wird die Platine an die **serielle Schnittstelle** (in der Regel mit **COM1** oder **COM2** bezeichnet) angeschlossen.

Über diesen Anschluss kann der Computer z. B. die Leuchtdiode in Abb. 1 ein- und ausschalten; er liefert auch den für den Betrieb der Leuchtdiode erforderlichen Strom. Geräte, die von dem PC geschaltet werden, bezeichnet man als **Aktoren**. Weitere Aktoren sind Motoren, Glühbirnen oder Elektromagnete.

Der Computer kann aber auch in Erfahrung bringen, ob ein Schalter auf der Platine geschlossen ist oder gerade viel Licht auf einen lichtempfindlichen Widerstand trifft. Geräte, welche dem Computer Informationen über Helligkeiten, Temperaturen oder Schalterzustände liefern, bezeichnet man als **Sensoren**.

Das Bindeglied zwischen den Aktoren bzw. Sensoren und dem Computer bezeichnet man allgemein als **Interface**. Für viele Modellanlagen muss man ein solches Interface zusätzlich kaufen. Für unsere Versuche benutzen wir als Interface die serielle Schnittstelle, die in nahezu allen PCs zu finden ist. Wie diese nun funktioniert, behandeln wir im nächsten Abschnitt.

---

[1] Bezugsquellen für solche Experimentiersets findest du im Anhang.

Die serielle Schnittstelle und COMX

Die serielle Schnittstelle hat eine Reihe von Anschlüssen (Abb. 2); man unterteilt sie in die so genannten **Ausgänge**, die **Eingänge** und den **Masseanschluss** (häufig kurz Masse genannt). Die Aktoren werden mit jeweils einem Anschluss an einen der Ausgänge und mit dem anderen an die Masse (⊥ oder GND) angeschlossen. Die Bezeichnung der Ein- und Ausgänge ist zurückzuführen auf ihre Bedeutung bei der Datenübertragung. Wir benutzen sie hier einfach nur zur Unterscheidung der verschiedenen Anschlüsse.

Pin	Ein-/Ausgang	Bezeichnung
3	Aus	TxD (Transmit Data)
2	Ein	RxD (Receive Data)
7	Aus	RTS (Request to Send)
8	Ein	CTS (Clear to Send)
5		GND (Ground)
1	Ein	DCD (Data Carrier Detect)
4	Aus	DTR (Data Terminal Ready)
9	Ein	RI (Ring Indicator)
6	Ein	DSR (Data Set Ready)

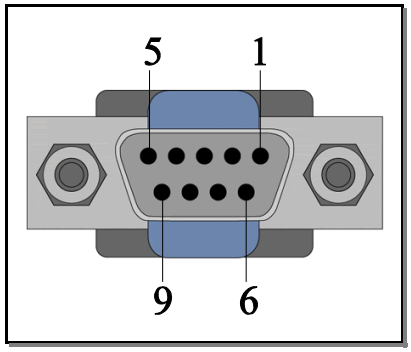

Abb. 2: Die serielle Schnittstelle (Buchse)

In diesem Kapitel kümmern wir uns zunächst nur um die Ausgänge. Mit ihnen wollen wir unsere Aktoren steuern. Mit geeigneten Steuerbefehlen können die Ausgänge auf +12 V (**High**) bzw. -12 V (**Low**) eingestellt werden. Manchmal ist dabei die Vorstellung in Abb. 3 hilfreich: Ein Schalter S wird im Innern der seriellen Schnittstelle durch entsprechende Anweisungen umgeschaltet. Steht er oben, hat der Ausgang den Zustand High, andernfalls den Zustand Low. In diesem Modell wird ein angeschlossener Aktor je nach Zustand des Ausgangs einmal mit der oberen und einmal mit der unteren Quelle verbunden. Ein Wechsel des Zustands von High nach Low bedeutet eine Umpolung und damit eine Änderung der Stromrichtung.

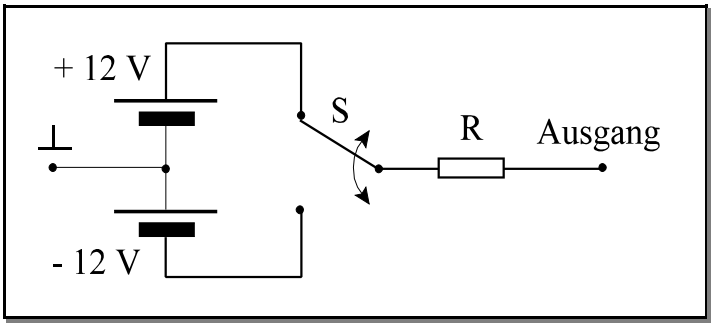

Abb. 3: Schaltermodell für einen Ausgang

Der Widerstand R hat eine wichtige Bedeutung; er begrenzt den Strom auf etwa 20 mA. Dies hat zwei Vorteile:

1. Empfindliche Bauteile wie unsere Leuchtdioden können problemlos an diesen Ausgang angeschlossen werden.

2. Auch wenn man versehentlich den Ausgang direkt mit dem Masse-Anschluss verbindet, gibt es keine Überlastung; die serielle Schnittstelle wird nicht zerstört.

JavaScript kennt von Hause aus keine Steuerbefehle für die serielle Schnittstelle. Deswegen erweitern wir den Befehlsvorrat von JavaScript durch eine **ActiveX-Komponente** mit dem Namen COMX. Dazu fügen wir in jedes HTML-Dokument, mit dem Steuerbefehle an die serielle Schnittstelle gegeben werden sollen, diese COMX-Komponente ein. Mit dem Programm Frontpage Express geht das sehr einfach: Zuerst wählt man im Einfügen-Menü die Auswahl `Andere Komponenten – ActiveX-Steuerelement` (Abb. 4).

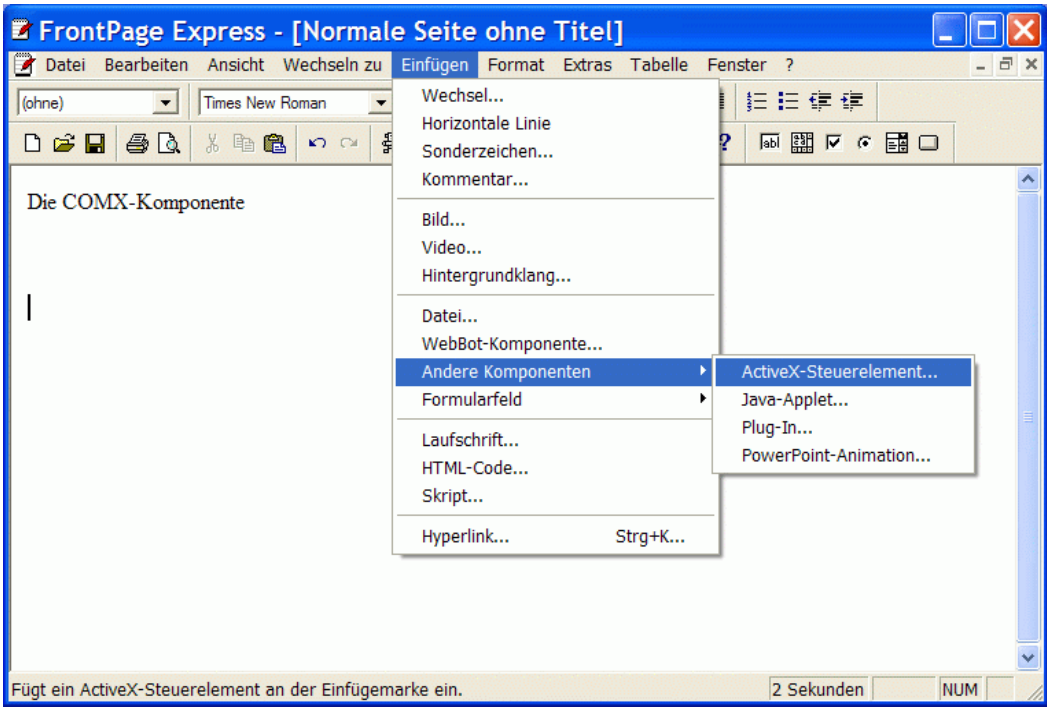

Abb. 4: So fügt man mit FPExpress unsere COMX-Komponente in ein HTML-Dokument ein.

Es erscheint das Fenster aus Abb. 5. Hier wählen wir als Steuerelement das COMX-Element[2]. Bevor wir die OK-Taste betätigen, geben wir dem Element noch einen Namen. Er kann beliebig gewählt werden. Wir benutzen hier und im Folgenden immer den Namen COMX.

[2]Solltest du die COMX-Komponente nicht in dieser Auswahl finden, muss sie erst noch auf dem Rechner installiert werden. Wie das geht, findest du im Anhang.

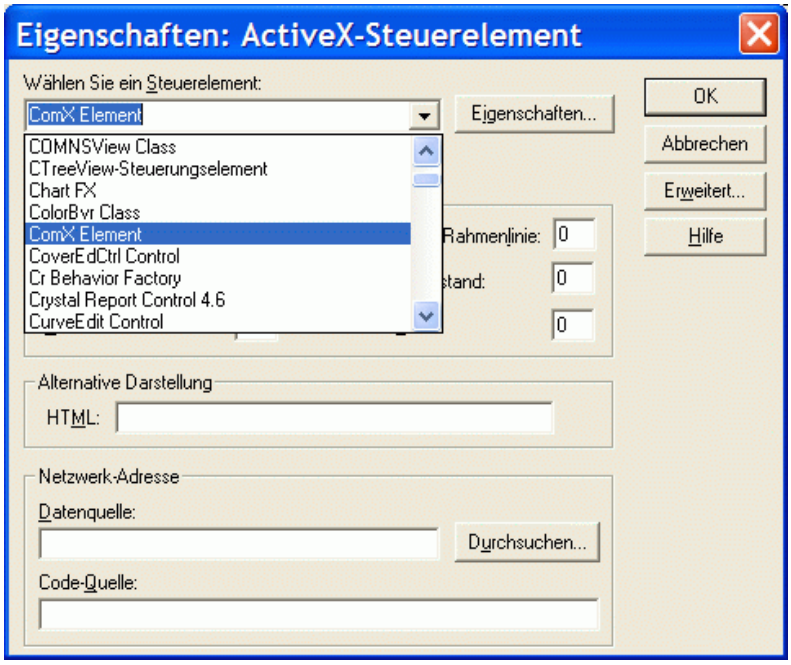

Abb. 5: Aus dem großen Angebot wählen wir `ComX Element`.

Danach sehen wir im Dokument die eingefügte COMX-Komonente. Sie kann wie ein Bild in der Größe verändert oder auch an anderen Stellen positioniert werden.

Abb. 6: Unsere COMX-Komponente im HTML-Dokument

Die COMX-Komponente wird von JavaScript als ein neues Objekt angesehen. Dieses Objekt besitzt zum Steuern folgende Methoden:

Meth. /Eigens.	Bedeutung	Beispiel
`open(Nr)`	öffnet die COM-Schnittstelle Nr	Durch `COMX.open(1)` wird COM1 geöffnet
`close()`	schließt alle COM-Schnittstellen	
`TXD`	Zustand des Ausgangs TxD	Durch `COMX.TXD = 1` wird der Ausgang TxD auf High gesetzt.
`RTS`	Zustand des Ausgangs RTS	Durch `COMX.RTS = 0` wird der Ausgang RTS auf Low gesetzt.
`DTR`	Zustand des Ausgangs DTR	Durch `COMX.DTR = 0` wird der Ausgang DTR auf Low gesetzt.
`delay(t)`	lässt das Programm t Millisekunden warten	`COMX.delay(500)` lässt das Programm eine halbe Sekunden warten
`doEvents()`	s. u.	

Bevor die serielle Schnittstelle benutzt werden kann, muss sie geöffnet werden. Dies geschieht mit der `open`-Methode. Ist die Platine an COM1 angeschlossen, muss als Parameter die Zahl 1 angegeben werden; ist sie hingegen an COM2 angeschlossen, wird der Parameter 2 benutzt. Günstig ist es, wenn die `open`-Methode direkt beim Laden des Dokuments ausgeführt wird – etwa durch ein onLoad-Ereignis. Am Ende sollte die Schnittstelle wieder geschlossen werden. Dazu dient die `close`-Methode. Sie hat keinen Parameter.

Wir schalten eine Leuchtdiode

Zunächst schließen wir die Leuchtdiode (kurz LED) wie in Abb. 7 an. Dabei sollte der längere Anschluss der LED mit DTR verbunden sein, der andere mit dem Masse-Anschluss (GND). Wir benutzen nun ein HTML-Dokument mit eingefügter COMX-Komponente. Zunächst fügen wir vier Schaltflächen hinzu: die Öffnen-Schaltfläche soll die serielle Schnittstelle COM1 öffnen, die An-Schaltfläche soll die LED zum Leuchten bringen, die Aus-Schaltfläche soll sie wieder ausschalten und die Schließen-Schaltfläche soll die serielle Schnittstelle wieder schließen (Abb. 8).

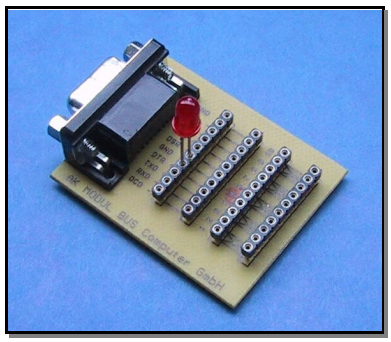

Abb. 7: Experiment mit LED

Die zugehörigen Funktionen lauten:

```
function oeffnen()
{
      COMX.open(1);
}
```

```
function an()
{
      COMX.DTR = 1;
}

function aus()
{
      COMX.DTR = 0;
}

function schluss()
{
      COMX.close();
}
```

Jetzt kommt der spannende Augenblick: Wir schließen unsere Platine mit dem Kabel an die Schnittstelle COM1 an und starten unser HTML-Dokument. Zuerst sieht die COMX-Komponente weiß aus wie in Abb. 8. Das Fragezeichen soll signalisieren, dass die Schnittstelle noch nicht geöffnet wurde. Wenn man jetzt die Schaltflächen „An“ bzw. „Aus“ betätigen würde, würde die LED keinerlei Reaktion zeigen.

Abb. 8: Vor dem Öffnen

Nun wird die Öffnen-Schaltfläche angeklickt. Im selben Augenblick wird die COMX-Komponente grün und das Fragezeichen wird durch die Nummer der Schnittstelle – in unserem Fall 1 – ersetzt (Abb. 9). Jetzt betätigen wir die An- und Aus-Schaltflächen: Das LED geht gehorsam an und aus.

Abb. 9: Nach dem Öffnen

An dieser Stelle fragst du dich vielleicht, warum die LED im LOW-Zustand nicht leuchtet, ist sie nach der Abb. 3 doch auch mit einer elektrischen Quelle verbunden, nur eben umgekehrt gepolt. Nun, die Lösung dieses Rätsels ist ganz einfach, wenn man weiß, dass Leuchtdioden wie ein Ventil wirken: Sie lassen den elektrischen Strom nur in eine Richtung hindurch, und zwar, wenn die Anode (A) an den Plus-Pol der elektrischen Quelle und die Kathode (K) an den Minuspol angeschlossen wird (Abb. 10). Bei der umgekehrten Polung sperren sie und leuchten dann auch nicht.

Zeigt der Pfeil im Schaltsymbol in Richtung auf den Minuspol der elektrischen Quelle, dann ist die Leuchtdiode in Durchlassrichtung geschaltet, andernfalls in Sperrrichtung.

Wenn am Schluss die Schnittstelle mit der Schließen-Schaltfläche geschlossen wird, dann wird die COMX-Komponente rot eingefärbt; nun lässt sich die Leuchtdiode nicht mehr ein- und ausschalten.

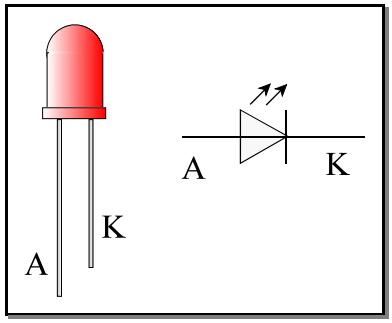

Abb. 10: Die LED und ihr Symbol

Aufgaben

1. Schließe die Leuchtdiode auf der Platine genau umgekehrt an wie im Text zu Abb. 7 beschrieben. Betätige nun die Schaltflächen „an“ und „aus“. Was fällt auf? Erkläre.

2. Durch Betätigen der Schaltfläche „an“ soll die Leuchtdiode eingeschaltet werden und anschließend nach 3 Sekunden selbstständig wieder ausgehen.

Wir lassen eine LED blinken

Zunächst legen wir ein neues HTML-Dokument mit unserer COMX-Komponente an. Darunter setzen wir eine Schaltflächen mit der Bezeichnung „blinken“ ein. Wenn diese betätigt wird, soll die LED im Sekundentakt 10 mal ein- und ausgeschaltet werden. Die zugehörige Funktion sieht so aus:

```
function blinken()
{
     for (var i = 0; i < 10; i++)
     {
          COMX.DTR = 1;
          COMX.delay(1000);
          COMX.DTR = 0;
          COMX.delay(1000);
     }
}
```

Bevor die init-Funktion aufgerufen werden kann, muss die Schnittstelle geöffnet werden. Dazu benutzen wir die Funktion `init()`:

```
function init()
{
     COMX.open(1);
}
```

Die blinken-Funktion lässt man praktischerweise durch das onLoad-Ereignis auslösen; dazu schreibt man `onLoad = "init()"` in den BODY-Tag des Dokuments:

```
<BODY ... onLoad = "init()">
```

Wenn du jetzt das HTML-Dokument startest, wird die Schnittstelle augenblicklich geöffnet; du erkennst das daran, dass die COMX-Componente von Beginn an grün ist. Sobald du nun die Blinken-Schaltfläche betätigst, geht das Lämpchen im Sekundenrhythmus zehn mal an und aus.

Gleichzeitig wirst du vielleicht auch feststellen, dass in dieser Zeit, der Browser auf keine weitere Eingabe reagiert. Das liegt daran, dass Windows sämtliche Maus- oder Tastatur-Eingaben als Ereignisse ansieht, die es mithilfe von Botschaften an die entsprechenden Programme weiterreicht. Der Browser ist aber nach dem Aufruf einer Funktion voll und ganz damit be-

schäftigt, die einzelnen Anweisungen des Funktionsrumpfes auszuführen; er widmet sich normalerweise den Botschaften erst wieder, wenn die letzte Anweisung des Funktionsrumpfes ausgeführt wurde.

In vielen Programmiersprachen gibt es nun spezielle Anweisungen, welche ein Programm zwingen, anliegende Botschaften sofort zu bearbeiten. JavaScript kennt einen derartigen Befehl nicht. Deswegen erhielt unsere Komponente COMX die Methode `doEvents`:

> Die Methode `doEvents` von COMX zwingt den Browser dazu, die Ausführung einer Funktion zu unterbrechen, um anliegende Botschaften zu bearbeiten.

Damit kann die Blinkfunktion nun bei Bedarf abgebrochen werden. Dazu fügen wir neben der Schaltfläche „blinken“ noch eine weitere Schaltfläche „stopp“ ein. Wenn du diese Schaltfläche betätigst, wird eine Funktion `ende()` aufgerufen. In dieser Funktion wird der Schleifenindex `i` auf 10 gesetzt. Natürlich kann die Funktion `ende()` auf diese Variable nur zugreifen, wenn diese global definiert ist. Hat `i` den Wert 10 erhalten, wird die Schleife abgebrochen.

```
var i; // i global

function blinken()
{
     for (i = 0; i < 10; i++)
     {
          COMX.DTR = 1;
          COMX.delay(1000);
          COMX.DTR = 0;
          COMX.delay(1000);
          COMX.doEvents(); Ereignisse behandeln
     }
}


function ende()
{
     i = 10;
}
```

Wenn nun die stopp-Schaltfläche z. B. beim dritten Schleifendurchlauf betätigt wird, sorgt der `doEvent`-Befehl dafür, dass die Ende-Funktion bereits am Ende dieses Schleifendurchlaufs ausgeführt wird. Ohne den `doEvent`-Befehl würde sie erst im Anschluss an die Funktion `blinken`, also erst nach der gesamten Schleife ausgeführt; dann wäre sie natürlich nutzlos.

Aufgaben

1. Programmiere ein Blinklicht mit variabler Frequenz[3]. Die Frequenz und die Gesamtdauer sollen in Textfeldern einzugeben sein.

2. Programmiere eine Fußgänger-Ampel, welche 10 Ampelphasen mit wählbarem Zeitintervall durchführt. Benutze eine rote und eine grüne Leuchtdiode. Das Programm soll auch vorzeitig zu stoppen sein.

3. Programmiere eine Straßenampel ähnlich wie in Aufgabe 3. Mache dich vorher kundig, in welcher Reihenfolge welche Signale (rot, gelb, grün oder Kombinationen davon) gesetzt werden.

4. Eine Ampel soll programmgesteuert geschaltet werden. Das Programm kann in einem Textfeld eingegeben werden und besteht aus einer Zeichenkette mit den Ziffern 1, 2, 3 und 4. Dabei steht 1 für grün, 2 für gelb, 3 für rot und 4 für rot-gelb. Mehrere gleiche Ziffern hintereinander sollen die entsprechende Ampelphase vervielfachen.

5. Programmiere ein Disko-Licht: Die LED soll in zufällig bestimmten Zeitabständen Lichtblitze aussenden.

6. Baue in Aufgabe 2 statt der LED einen Kopfhörer ein. Teste verschiedene Frequenzen aus.

7. Programmiere eine Funktion `sound(f, t)`. Diese soll für *t* Sekunden einen Ton der Frequenz *f* Hz erzeugen.

8. Facharbeit: Schreibe ein Programm, welches einen Text in (akustische) Morsezeichen umsetzt. Verfasse auch einen Aufsatz über Morsezeichen. Lies dazu auch die Kapitel „Entscheidungsbäume und Informationsgehalt“ und „Codes überall“.

[3] Die Frequenz gibt an, wie viele Ein- und Ausschaltvorgänge in einer Sekunde erfolgen. Wenn in einer Sekunde 5 mal ein- und ausgeschaltet wird, sagt man: Die Frequenz ist 5 Hz (Hertz).

Wir regulieren die Helligkeit einer LED

Bislang haben wir unsere LED nur ein- und ausgeschaltet. Jetzt wollen wir eine LED mit unterschiedlicher Helligkeit leuchten lassen. Leider lässt sich die Spannung an den Ausgängen der seriellen Schnittstelle nicht verändern. Deswegen greifen wir auf ein Verfahren zurück, welches vor allem bei der Ansteuerung von Elektromotoren (z.B. bei Modelleisenbahnen) Anwendung findet: die so genannte **Pulsweitenmodulation**.

So kompliziert das Wort klingt, so einfach ist das Verfahren selbst. Die LED erhält nur kurze Stromimpulse. Wenn sie zeitlich sehr rasch erfolgen, dann kann das Auge die einzelnen Lichtblitze nicht mehr auflösen. Es nimmt vielmehr eine gemittelte Helligkeit war; diese ist um so höher, je weiter die Pulse im Zeitdiagramm (Abb. 11) sind.

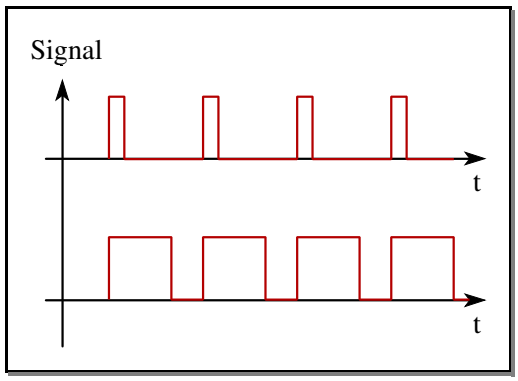

Abb. 11: Das untere Signal führt zu einer größeren Helligkeit.

In unserem Programm benutzen wir eine Frequenz von 100 Lichtblitzen pro Sekunde. Wenn dann ein Puls z. B. 3 ms lang dauert, ist die LED 30% der Zeit angeschaltet; in diesem Fall leuchtet sie recht schwach. Wenn der Puls hingegen 10 ms lang ist, dann leuchtet die LED permanent und entsprechend hell.

Aufgaben

1. Schreibe ein Programm, durch welches die Helligkeit einer LED durch eine Zahl zwischen 0 und 10 gesteuert wird. Die LED soll nach 5 Sekunden wieder ausgehen.

2. Eine LED soll langsam im heller und anschließend wieder dunkler werden. Schreibe ein entsprechendes Programm.

3. Bei neueren Ampelanlagen werden die Lampen sanft ein- und ausgeschaltet. Welchen Vorteil hat das? Schreibe ein Programm für eine Fußgängerampel, bei der die LEDs sanft ein- und ausgeschaltet werden.