Aufzugsteuerung mit Arduino

Von Günther Zivny

Das bekannte Mikrocontroller-Lernsystem ARDUINO wird auch in der Schule verwendet, wie verschiedene Beiträge in der letzten Zeit zeigen. Dieses System bietet auf der einen Seite einen problemlosen Einstieg in die Mikrocontroller-Programmierung, hat auf der anderen Seite aber keine Begrenzungen wie die meisten anderen „einfachen“ Systeme. Hier soll nun als Anwendungsbeispiel eine Aufzugsteuerung beschrieben werden. Aufzugsteuerungen gehören nicht gerade zu den einfachsten Übungen, das hier beschriebene Konzept ist aber überschaubar.

Das Funktionsmodell

Eine Aufzugsteuerung braucht natürlich ein Modell, das gesteuert wird. Man kann hier auf ein Fertigprodukt, das speziell für den Einsatz in der Schule entwickelt wurde, zurückgreifen (Bild 1). Auch der Selbstbau eines Modells ist möglich, wobei sich der Rückgriff auf einen Konstruktionsbaukasten anbietet. Der Autor verwendete ein flaches Minimalmodell, welches aber auch seinen Zweck erfüllt (Bild 2 und 3). Hier kann sich also Kreativität entfalten, wobei man den Zeitaufwand nicht unterschätzen und allzu ambitionierte Pläne der jungen Konstrukteure eher dämpfen sollte. Es ist auch sinnvoll, den Bau des Modells ganz von der Programmierung der Steuerung zu trennen und das Modell vielleicht erst einmal manuell zu steuern. Im Bildungsplan „Technik“ von Baden-Württemberg wird eine Torsteuerung explizit erwähnt. Ein Aufzug, der nur zwischen zwei Stockwerken pendelt, ist von der Steuerung her im Prinzip nichts anderes als ein senkrecht gestelltes Schiebetor. Hier könnte man also anknüpfen.

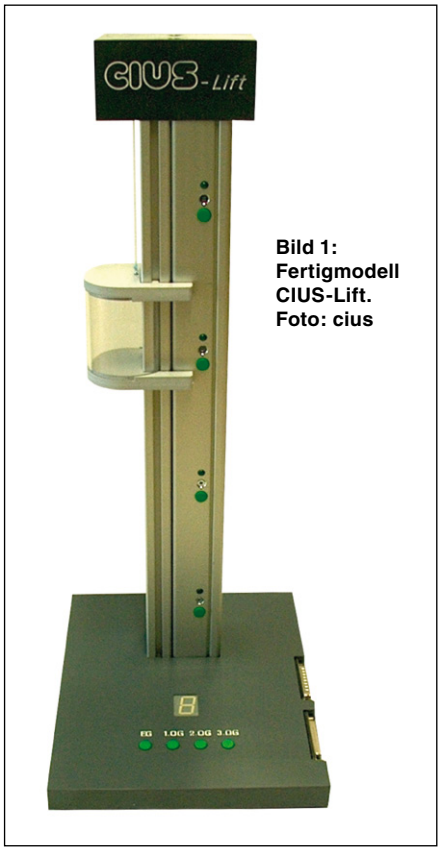

Bild 1: Fertigmodell CIUS-Lift. Foto: cius

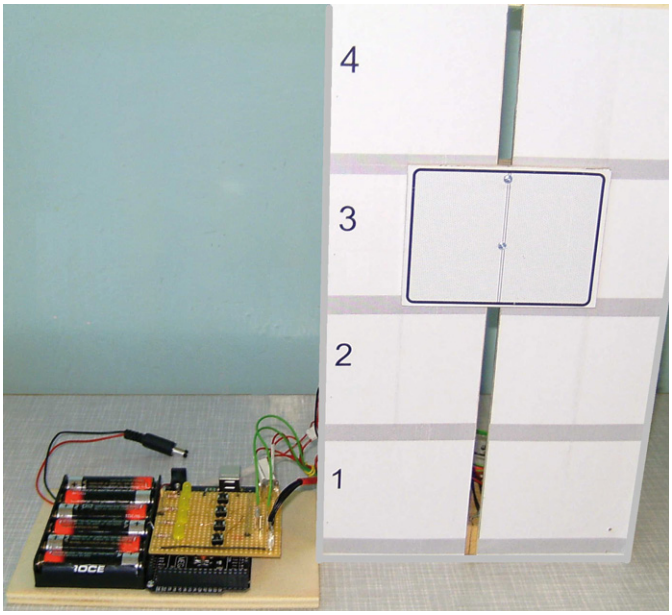

Bild 2: Darstellung des selbst gebauten Modells von vorne. Die Ruftasten und die Anforderungs-LEDs wurden hier auf einer Leiterplatte montiert. Diese Zusatzplatine, in der ARDUINO-Bezeichnungsweise „shield“ genannt, kann direkt auf die ARDUINO-Platine gesteckt werden. Für die Verbindung gibt es spezielle Stiftleisten.

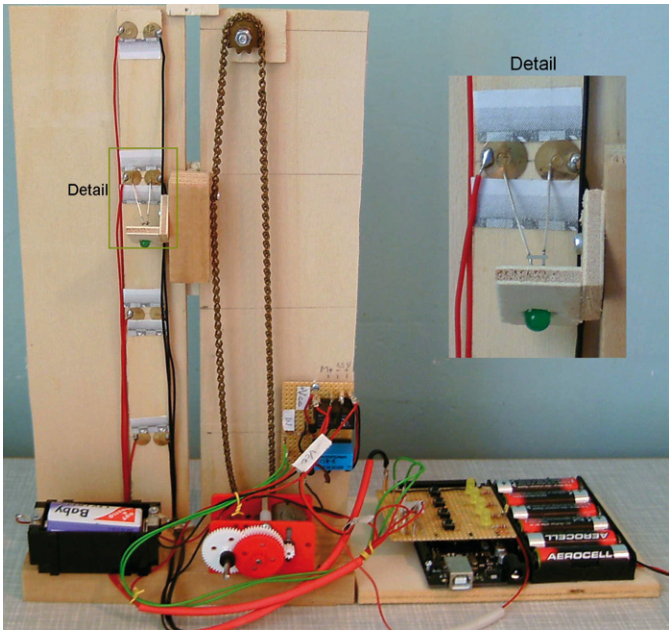

Bild 3: Ansicht von hinten. Man sieht den Getriebemotor und die Antriebskette. Über dem Getriebemotor ist die Relaisplatine zu sehen. Die Stockwerkskontakte wurden hier mit blanken Reißnägeln realisiert (siehe Detail). Als Kontaktdrähte dienen die kurzgeschlossenen Anschlussdrähte einer LED.

Die Steuerung

Ein Aufzug hat in der Kabine Zielwahltasten und in den Stockwerken Ruftasten. Bei dem Modell wurde auf die Zielwahltasten in der Kabine verzichtet, das wäre zu aufwändig. Zielwahltasten und Ruftasten haben ohnehin dieselbe Funktion: Die Kabine fährt das entsprechende Stockwerk an. In den Stockwerken kann man wählen, ob man nach oben oder nach unten fahren möchte. Diese Wahlmöglichkeit gibt es hier nicht. Es gibt also nur vier Tasten, die man als Ruftasten oder als Zielwahltasten sehen kann. Nach dem Drücken einer Taste leuchtet die zugehörige Anforderungs-LED auf und signalisiert, dass der Auftrag angenommen wurde.

Schaltplan

Der Leistungsteil, also der Getriebemotor, wird ganz klassisch mit Relais gesteuert. Dadurch ergibt sich eine galvanische Trennung zwischen dem ARDUINO (9 V) und dem Motor (1,5 V). Das ist zwar nicht unbedingt notwendig, aber sinnvoll. Es gibt drei Zustände: Aufwärts – Stopp – Abwärts. Die Ansteuerung kann manuell über Tasten oder eben durch den ARDUINO erfolgen.

Tragen Sie bitte Sorge, dass die Ausgänge des ARDUINO nicht kurzgeschlossen werden, da dies zur Zerstörung des Mikrocontrollers führen kann.

Der ARDUINO UNO oder die größere Version MEGA 2560 können verwendet werden.

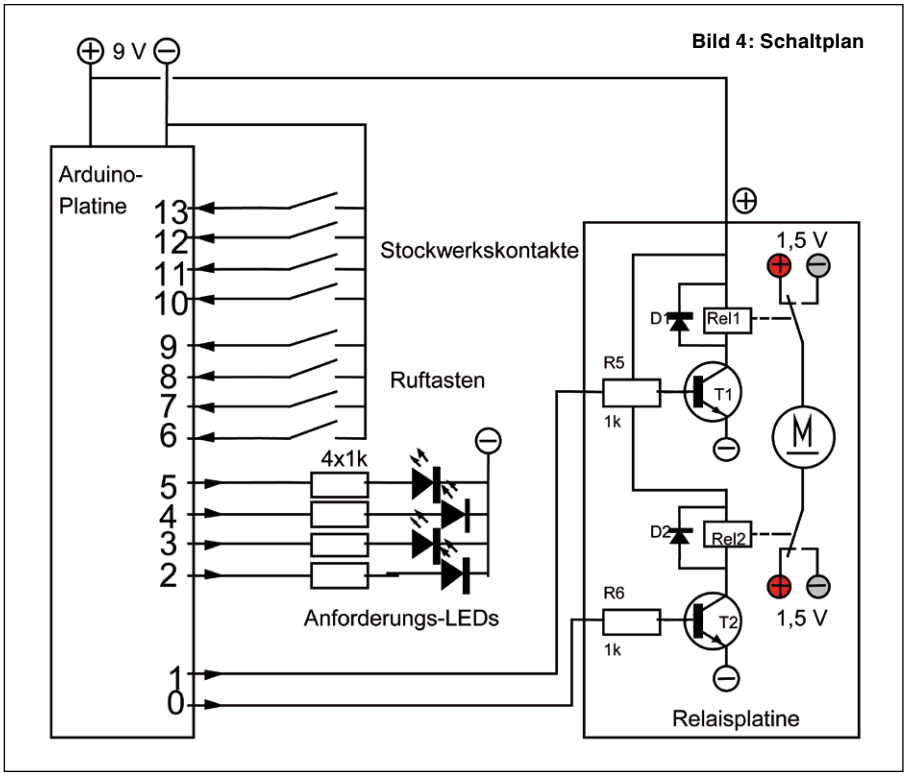

Bild 4: Schaltplan

Das Programm

Die Mikrocontroller-Programmierung ist an sich sehr komplex. Das Programmpaket, das man von der ARDUINO-Webseite herunterlädt (IDE = integrated development environment), enthält aber alle „tools“, die man benötigt, und es sind auch bereits gewisse Voreinstellungen getroffen, sodass der Start wirklich sehr problemlos ist.

Die Programmierung des ARDUINO erfolgt standardmäßig mit der Programmiersprache C++. Diese ist auch in dem Programmpaket enthalten. Das Paket enthält auch spezielle Funktionen, die eine leichte Programmierung ermöglichen, ohne in die Details des Mikrocontrollers einsteigen zu müssen. Um ein Auto zu starten, genügt es ja auch, den Zündschlüssel zu drehen. Um das, was dann im Einzelnen abläuft, muss man sich nicht kümmern. Am Ende läuft der Motor eben.

Das hier vorgestellte Programm hat eine einfache, lineare Struktur und es werden nur wenige Basisbefehle verwendet.

In Bild 5 ist das Programmkonzept dargestellt.

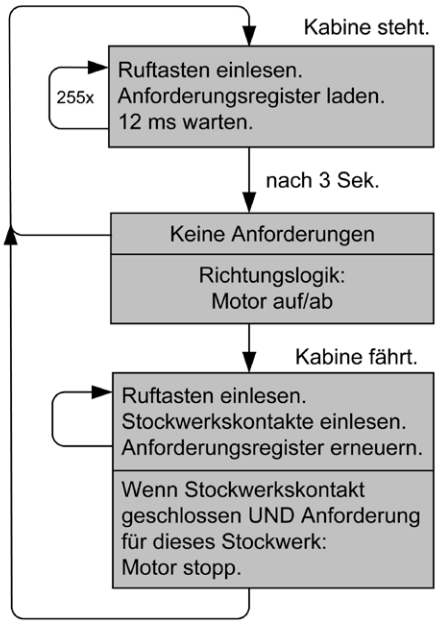

Bild 5: Konzept des Programms.

Variablendefinition

Das Programm beginnt mit der Definition der Variablen. Es werden nur 8-Bit-Variablen vom Typ unsigned char verwendet.

Setup-Routine

Hier wird festgelegt, welche der digitalen Anschlüsse Ausgänge sein sollen. Ferner werden die internen pull-up-Widerstände eingeschaltet. Diese verbinden die Eingänge mit der internen 5-V-Spannung. Die Eingänge haben also im unbeschalteten Zustand HIGH-Potential. Sie können von außen auf LOW gezogen werden. Schließlich wird noch der Anforderungsspeicher für den 4. Stock gesetzt (a4 = 1). Nach dem Anlegen der Betriebsspannung fährt die Kabine aus einer beliebigen Anfangsposition in den 4. Stock. Danach ist die Anlage betriebsbereit.

Endlosschleife *loop*

In diese Schleife wird das eigentliche Programm geschrieben.

Einlesen der Tastenzustände und Steuerung der LEDs: Auf der ARDU-

INO-Platine sind die digitalen Anschlüsse durchnummeriert (Bild 4). Die Anschlüsse werden immer einzeln mit diesen Nummern angesprochen.

Die Funktion *digitalWrite(3, HIGH)* setzt beispielsweise den Anschluss Nr. 3 auf High (+5V). Die Anschlussnummer kann auch eine Variable sein und das Wort HIGH (bzw. LOW) kann auch durch „1“ (bzw. „0“) ersetzt werden.

So kann man auch schreiben:
X = 3
Y = 1
digitalWrite(X,Y).

Um die Eingaben richtig zu verarbeiten, ist etwas Aussagenlogik notwendig. In C++ wird der Wert „wahr“ durch eine „1“ dargestellt und der Wert „falsch“ durch eine „0“. Es handelt sich um normale Zahlen, mit denen man auch rechnen kann. Technisch wird „wahr“ durch einen hohen Spannungspegel (HIGH) und „falsch“ durch einen niederen Spannungspegel (LOW) dargestellt.

Das Programm beginnt mit einer Warteschleife. In dieser Phase steht die Kabine in einem Stockwerk, sodass die Fahrgäste ein- und aussteigen können. Hier werden nur die Ruftasten eingelesen. Beim Drücken einer Ruftaste geht der zugehörige Anschluss auf LOW, also null. Da wir aber lieber positiv denken, kann man den Wert durch Vorsetzen eines Ausrufezeichens „!“ ins Gegenteil verkehren.
Es ist: r = 0 → !r = 1

Beim Drücken einer Ruftaste wird der Anforderungsspeicher geladen und die zugehörige Anforderungs-LED leuchtet auf. Nach dem Loslassen der Taste darf sich nichts ändern. Eine solche Funktion wird in der fest verdrahteten Schaltungstechnik mit einer Selbsthalteschaltung (Flip-Flop) realisiert. Softwaremäßig lässt sich das mit einer ODER-Verknüpfung erreichen. Diese wird in C++ durch zwei senkrechte Striche symbolisiert. Man kann sich unter dem Symbol eine Parallelschaltung von zwei Schaltern vorstellen.

Die Formulierung
a = !r || a

hat genau die gewünschte Wirkung. Ist der Anforderungsspeicher a = 0 und wird r = 0 bzw. !r = 1, dann wird a = 1 und bleibt so, auch wenn nach dem Loslassen der Taste r = 1 wird.

Der Zustand des Anforderungsspeichers wird durch die Anforderungs-LEDs angezeigt.

Nach etwa 3 Sekunden ist die Warteschleife zu Ende.

Danach wird geprüft, ob überhaupt noch Anforderungen vorhanden sind. Wenn nicht, wird in die Warteschleife zurückgesprungen. Hierzu werden die Anforderungsbits a1 bis a4 1-2-4-8 gewichtet und addiert und man erhält die Zahl a. Gleiches geschieht weiter unten mit den Stockwerksbits s1 bis s4 und man erhält die Zahl s. Für die Richtungslogik werden a und s verglichen. Ist der Anforderungsspeicher a höher als die Ist-Position s, fährt die Kabine nach oben, sonst nach unten.

Während der Kabinenfahrt läuft eine Endlosschleife. Hier werden sowohl die Ruftasten als auch die Stockwerkskontakte eingelesen. Wenn ein Stockwerkskontakt in einem Stockwerk geschlossen wird, für das eine Anforderung besteht, hält die Kabine an und es wird in die Warteschleife zurückverzweigt. Die Bedingung „Stockwerkskontakt geschlossen“ UND „Anforderung vorhanden“ wird durch die UND-Verknüpfung realisiert. Hierfür gibt es in C++ das Zeichen „&&“.

Die Formulierung
x = !s1 && a1

hat den gewünschten Effekt für den ersten Stock. Dies muss für alle Stockwerke geprüft werden. Man kann hierzu die ODER-Verknüpfung aneinander reihen. In der Formulierung
w = (x) || (y) || (z)

wird w wahr, wenn mindestens einer der in Klammern gesetzten Ausdrücke wahr ist.

Verbessertes Programm

Das beschriebene Programm funktioniert so weit ganz gut. Kritische Beobachter werden aber anmerken, dass die Kabine immer zuerst nach oben fährt. Nur wenn oberhalb der aktuellen Position keine Anforderung mehr besteht, fährt die Kabine nach unten. Für das Modell spielt das keine Rolle, in der Realität darf das aber nicht sein. Nehmen wir an, zwei Fahrgäste besteigen im 4. Stock die Kabine. Der eine möchte in den 2. Stock, der andere in den 1. Stock. Während die Kabine nun fährt, betätigt jemand die Ruftaste für den 4. Stock. Die Kabine hält dann im 2. Stock, wo der Fahrgast aussteigt, fährt danach aber mit dem zweiten Fahrgast zurück in den 4. Stock.

Um den Mangel zu beheben, muss die Kabine ihre Richtung so lange beibehalten, bis in dieser Richtung keine Anforderungen mehr vorhanden sind. Die Lösung des Problems ist nicht ganz einfach. Da die vorhandene Methode für die Aufwärtsfahrt funktioniert, ist es naheliegend, für den Fall der Abwärtsfahrt den Spieß umzudrehen und die Anforderungsbits a1 bis a4 anders zu gewichten, sodass das Bit für den 1. Stock jetzt das höchstwertige ist. Gleiches muss mit den Stockwerksbits s1 bis s4 geschehen. Damit fährt die Kabine so lange abwärts, bis es unterhalb der aktuellen Position keine Anforderungen mehr gibt. Man muss hier also zweigleisig fahren. Um die aktuelle Richtung zu kennzeichnen, gibt es die Richtungsvariable ri. Es sind hier zwei if-Abfragen ineinandergeschachtelt. Das ist möglich, aber etwas unübersichtlich.

Inbetriebnahme

Vor dem Start sollte die Kabine eine mittlere Position haben. Zuerst wird die 1,5-V-Batterie eingelegt. Dann wird der ARDUINO angeschlossen. Die Anforderungs-LED für den 4. Stock leuchtet auf und die Kabine fährt in den 4. Stock. Sollte sie nach unten fahren, die Stromversorgung unterbrechen und die Anschlüsse der Relaisplatine vertauschen. Danach ist die Anlage betriebsbereit.

Damit wollen wir enden. Die Basis für einen weiteren Ausbau des Modells und des Programms ist gelegt.

Programm-Liste

Die rot gedruckten Teile können durch die erweiterte Richtungslogik ersetzt werden.

```
//Programm Aufzug_tu
//Variablendefinitionen
(globale Variablen)
unsigned char a, aa, a1, a2, a3, a4;
unsigned char r, r1, r2, r3, r4;
unsigned char s, ss, s1, s2, s3, s4;
unsigned char i, n, ri;
```