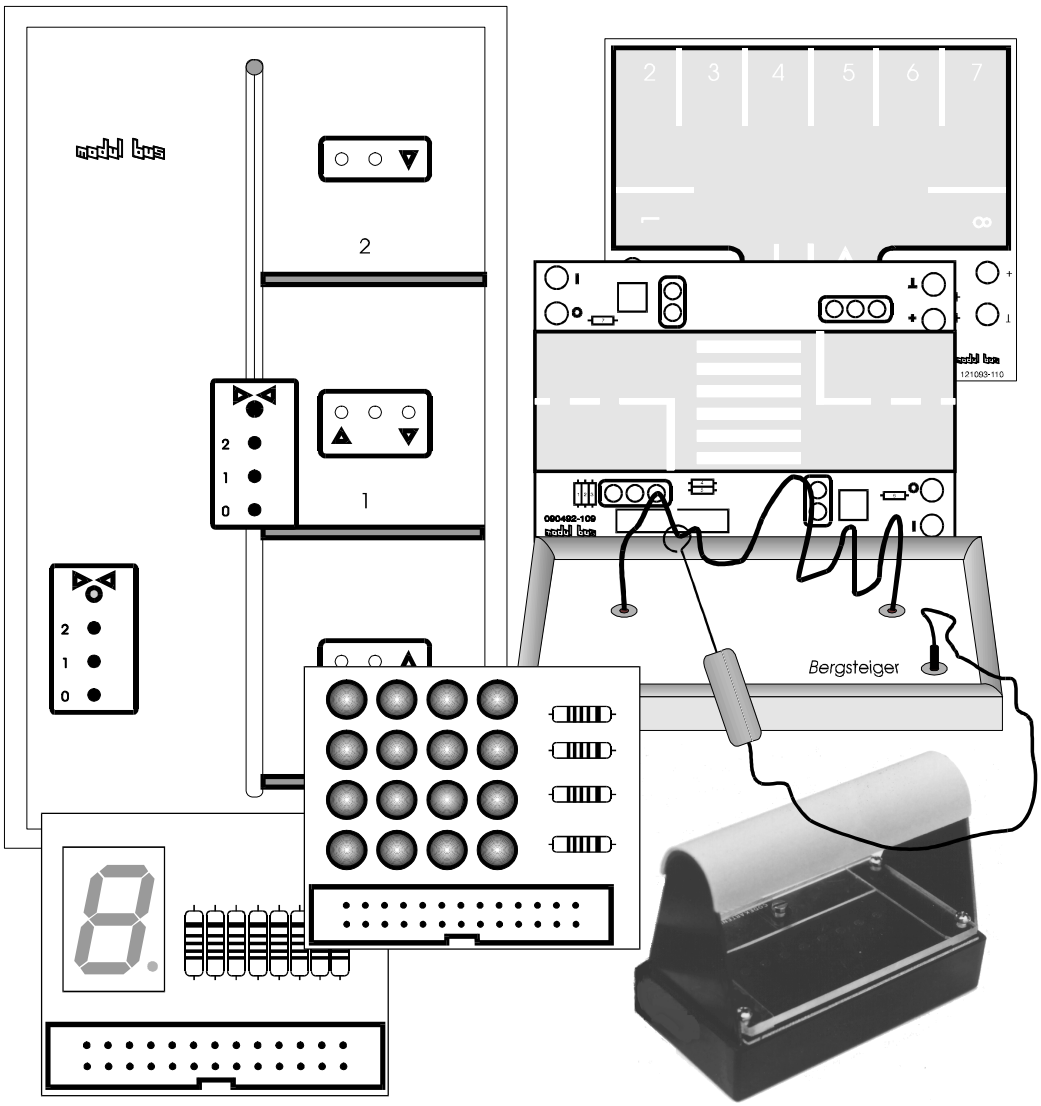
Josef Hüvelmeyer

Modellversuche

Teil I

Experimente mit Kleinmodellen und der Programmierumgebung *Do-it* oder einer Programmiersprache



Inhalt

Seite

Vorwort 3

1. Auftrag von Richtlinien und Lehrplänen 4
2. Grundlagen 5
3. Steuerung von Ampelanlagen 7
4. Im Parkhaus 12
5. Der heiße Draht 14
6. Steuerung von Schrittmotoren *) 16
7. Aufzugsteuerung *) 19
8. Ansteuerung einer 7-Segment-Anzeige 23
9. Matrix 27
10. Codekartenleser 32

Anhang 37

Arbeitsblätter 44

*) Bei den so gekennzeichneten Kapiteln wird eine Programmiersprache zur Lösung eingesetzt. Bei allen Versuchen kann sie jedoch auch verwandt werden.

Impressum
2. Auflage, Nov. 03

Autor: J. Hüvelmeyer, IFS - Uni Dortmund

Kontaktadresse:
Arbeitskreis Schulsoftware, Postfach 3026, 48472 Hörstel

Die Beschreibung basiert auf den im April 96 verfügbaren Produkten. Änderungen können jederzeit auch ohne Vorankündigung durchgeführt werden.

161195.pjk

Vorwort

„Prozeßdatenverarbeitung“ oder genauer gesagt „Messen, Steuern, Regeln mit dem Computer“ ist ein Teilbereich im Informatikunterricht, den viele Lehrerinnen und Lehrer mit gemischten Gefühlen betrachten. Auf der einen Seite erkennen sie, daß der Einsatz des Computers in dem Bereich der Prozeßsteuerung und Automation für unsere Wirtschaft wichtig ist und eine Reihe von nicht gerade wünschenswerten Nebeneffekte mit sich bringt. Grundlegende Kenntnisse in diesem Bereich werden daher von vielen Lehrerinnen und Lehrern als wichtige Lernziele anerkannt. Auch die Grundbildung, die sich insbesondere mit den Auswirkungen der Neuen Technologien beschäftigt, weist diesem Bereich eine bedeutende Rolle zu.

Hinzu kommt, daß die Schülerinnen und Schüler diesem Bereich ein großes Interesse entgegen bringen. Ampelanlagen zu schalten, Alarmanlagen zu programmieren oder Aufzüge zu steuern ist für viele motivierender, als ein Programm für die Ausleihe in der Schülerbücherei zu entwickeln oder eine Datenbank für das Betriebspraktikum aufzubauen.

Auf der anderen Seite setzt dieses Gebiet einen höheren apparativen Aufwand voraus, da zusätzlich Interface und Funktionsmodelle benötigt werden. Denn ein Fahrstuhl soll sich ja tatsächlich bewegen und nicht nur als Animation auf dem Bildschirm fahren. Für viele Lehrerinnen und Lehrer erhöhen sich mit den zusätzlichen Komponenten auch die Unwägbarkeiten im Unterricht. Konnte man sich inzwischen halbwegs darauf verlassen, daß die Computer mit der installierten Software auch im Unterricht verläßlich laufen, so befürchten viele, daß Interfaces und Funktionsmodelle nicht einwandfrei funktionieren und Situationen entstehen, die man im Unterricht lieber vermeiden möchte.

Dieses Buch will Hilfestellung bei der Umsetzung dieses interessanten Themenbereichs geben und zeigen, daß bei vertretbarem Aufwand dieses Gebiet mit Schülerinnen und Schülern behandelt werden kann.

Und noch ein Versprechen zum Schluß: Motivationsschwierigkeiten wird es dabei nicht geben!

1. Auftrag von Richtlinien und Lehrplänen

Bereits im Rahmenkonzept der BLK werden als wesentliche Elemente einer „Informationstechnischen Bildung in Schule und Ausbildung“ [1] die „informationstechnische Grundbildung“, die verpflichtend für alle Schülerinnen und Schüler ist, und darauf aufbauend eine „vertiefende informationstechnischen Bildung in Form der Informatik“ im Wahlpflichtbereich gefordert [1].

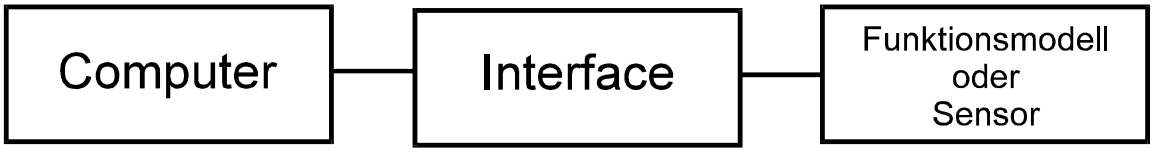
In beiden Bereichen spielt die Prozeßdatenverarbeitung eine bedeutende Rolle. Sie wird z.B. in den im Jahr 1992/93 in Kraft getretenen „Vorläufigen Richtlinien zur Informations- und Kommunikationstechnologischen Grundbildung in der Sekundarstufe I“ in NRW [2] als eine von drei Lernbereichen an erster Stelle genannt. Dabei sollen die Schülerinnen und Schüler Prinzipien der Datenerfassung und Steuerung technischer Vorgängen kennen lernen. „Zu diesem Zweck sollen sie mit den an den Schulen vorhandenen Computersystemen und mit Hilfe einiger Zusatzgeräte und entsprechender Programme z.B. Messungen durchführen, Codes lesen und technische Modelle von Handhabungsgeräten, Fertigungsmaschinen, Transportsystemen etc. steuern und regeln“ ([2] - S. 11) Das Kennenlernen von technischen Möglichkeiten und deren Auswirkungen für Individuum und Gesellschaft sind wesentliche Ziele der Grundbildung.

In dem auf der Grundbildung aufbauenden Informatikunterricht wird dieser Bereich erweitert. Die Schülerinnen und Schüler sollen nicht nur die Rolle von Bedienenden, Nutzenden oder Betroffenen einnehmen, sondern auch gestaltend in derartige Prozesse eingreifen. Die Erstellung eines Programms für die Steuerung eines Maschinenmodells [3] ist eine typischen Aufgabe. Nach Auffassung der Bund-Länder-Kommission für Bildungsplanung und Forschungsförderung gehört in den Informatikunterricht der Sekundarstufe I auch die „Erörterung von Prozeßsteuerung durch Mikroprozessoren“ [4].

Damit ein Unterricht entsprechend diesen Vorgaben durchgeführt werden kann, ist ein Zusatzausstattung erforderlich. Das Landesinstitut für Schule und Weiterbildung in Soest schreibt dazu in seinen Orientierungshilfen zur Ausstattung von allgemeinbildenden Schulen mit Hard- und Software: „Für das Lernfeld Prozeßdatenverarbeitung werden für die Informations- und kommunikationstechnologische Grundbildung sowie für den Wahlpflicht- /Differenzierungsbereich zusätzliche Funktionsmodelle ..., die mit dem Computer gesteuert und geregelt werden, benötigt. Die Kommunikation zwischen Rechner und Funktionsmodell erfolgt über eine geeignete Anschlußeinheit (Interface)“ [5]. Relativ genaue Angaben werden für ein typisches Schulinterface gemacht: „Ein universelles Interface sollte neben digitalen Ein- und Ausgänge (je mindestens 8) auch über (zwei) analoge Eingänge verfügen“ [5].

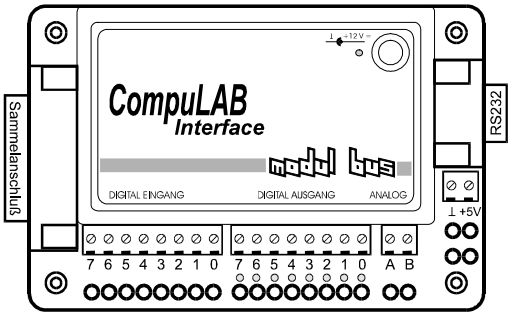
2. Grundlagen

Gemäß dem Auftrag der Richtlinien sollen Messungen durchgeführt und technische Prozesse mit dem Computer gesteuert werden. Die Signale, die ein Computer direkt liefern kann, sind erstens häufig codiert (z.B. werden mehrere Signale häufig als Block hintereinander gesandt) und zweitens zu schwach, um Lichtanlagen oder gar Maschinen zu steuern. Daher ist ein Interface erforderlich, mit dem die Signale des Computers dekodiert und verstärkt werden.



Die Kommunikation zwischen Computer und Interface erfolgt über vereinbarte Befehle. Programme, mit denen mit einem Computer technische Prozesse gesteuert werden, müssen sich dieser elementaren Befehle bedienen, die von Hersteller zu Hersteller unterschiedlich sind. Das erklärt, warum nicht mit jedem Programm jedes Interface angesteuert werden kann.

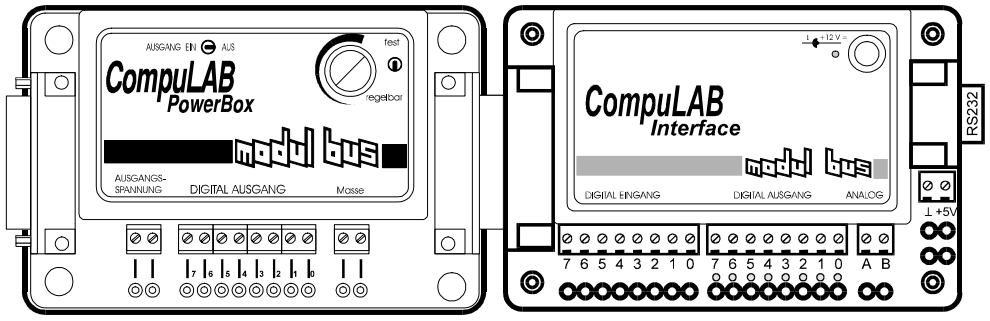
Verwendete Hardware



Die bisherigen Ausführung begründen, warum sich die Beschreibung auf definierte Produkte beziehen. In diesem Buch wird mit dem *CompuLAB*-System gearbeitet, daß an die serielle Schnittstelle des Computers angeschlossen wird. Nähere technische Informationen befinden sich im Anhang.

Alle Ein- und Ausgänge des Interfaces sind an 2mm Steckern, Schraubanschlüssen und an einen Sammelstecker herausgeführt. Während an den ersten beiden Anschlüssen vorwiegend eigene Experimentalaufbauten angeschlossen werden, erfolgt die Verbindung zu den Funktionsmodellen meist über den Sammelanschluß, die richtige Verbindung aller Ein- und Ausgänge ist damit auf einfache Art- und Weise gesichert.

Für viele Versuche reicht diese Konfiguration aus. In einigen Fällen wird jedoch eine höhere Ausgangsleistung benötigt (z.B. bei der Steuerung von Schrittmotoren). Zu diesem Zweck wurde die *PowerBox* entwickelt, die direkt an das *CompuLAB* Interface angedockt werden kann. Das bisher verwendete Steckernetzteil muß dabei durch eine leistungsfähigere Stromversorgung ersetzt werden.



Alternativ zu diesem Interface-System kann auch die *miniRS-Box* oder das *SIOS*-Interface eingesetzt werden. Nähere Angaben dazu befinden sch im Anhang.

Die nachfolgenden Kapitel beschäftigen sich jeweils mit einem Funktionsmodell. Sie alle sind als Fertigmodell erhältlich (siehe Anhang), können aber auch vielfach in Projektgruppen oder im Technikunterricht hergestellt werden.

Verwendete Software

Diesem Buch liegt eine Diskette bei, auf der sich die Beispielprogramme für die Experimentiersoftware *Do-it* befinden. *Do-it* liegt jedem *CompuLAB* Experimentier-Set bei, kann aber auch als Klassenraumlizenz bezogen werden. Die Installation dieses WINDOWS-Programms wird im Anhang beschrieben. Eine DOS-Version ist ebenfalls erhältlich. Nähere Angaben dazu befinden sich im Anhang.

Do-it enthält eine Reihe von Meßwerkzeugen, mit dem zeitliche Verläufe und Abhängigkeiten der gemessenen Spannungen dargestellt werden können. Daneben ist eine Programmierumgebung integriert, mit der nahezu ohne Vorkenntnisse Steuerungsprogramme entwickelt werden können. Selbst Fallunterscheidungen und Wiederholungen lassen sich damit codieren. Alle im Buch angeführten Beispielprogramme befinden sich auf der Diskette.

In einigen Kapiteln wird als Programmierwerkzeug eine Programmiersprache eingesetzt. Um sich nicht auf ein Programmiersystem zu beschränken, erfolgt die Beschreibung in einer Metasprache, die von denjenigen, die eine bestimmte Programmiersprache kennen, leicht übersetzt werden kann. Der Quellcode einiger Beispielprogramme für die Programmiersprachen Pascal und Comal befindet sich ebenfalls auf der beigefügten Diskette. Zur Ausführung ist eine Erweiterung für die jeweilige Programmiersprache (Unit oder Modul) erforderlich.

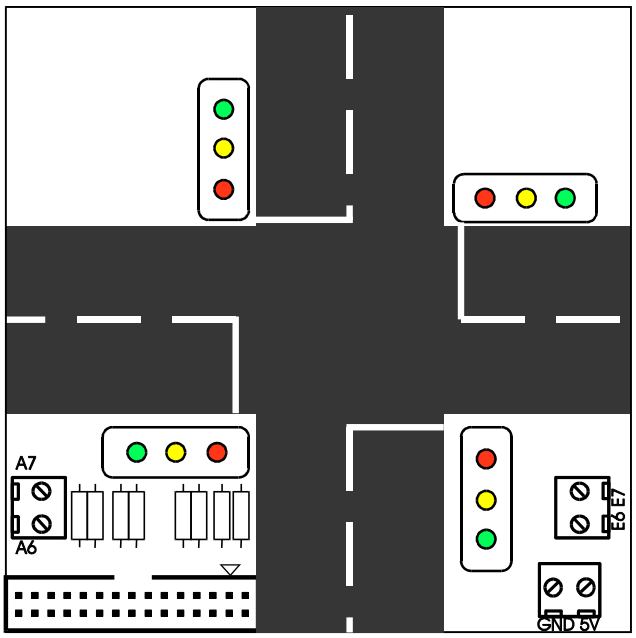
3. Steuerung von Ampelanlagen

Seitdem der Computer in der Schule im Bereich der Prozeßdatenverarbeitung eingesetzt wird, gehört die Steuerung von Ampelanlagen zu den beliebtesten Themen. Das liegt einerseits an den geringen hardwaremäßigen Voraussetzungen (gepufferte Datenleitungen genügen) und andererseits an der einfachen, überschaubaren Ansteuerung. Daher ist dieses Thema besonders für den Anfangsunterricht geeignet.

Die Funktionsmodelle

In dieser Beschreibung wird mit zwei Ampelanlagen gearbeitet: einer Kreuzungs- und einer Fußgängerampel. Bei den Modellen handelt es sich um bestückte Platinen, auf die Straßenbegrenzungen, Zebrastreifen etc. gedruckt wurden. Zunächst wird die Ansteuerung der einzelner Modelle beschrieben, danach werden sie über E/A-Leitungen miteinander verbunden, so daß Signale ausgetauscht werden können.

3.1 Die Ampelkreuzung



Damit die 12 Leuchtdioden der Ampelanlage mit den acht digitalen Ausgangsleitungen angesteuert werden können, wurden die jeweils gegenüberliegenden Ampeln zusammengeschaltet. Die Schülerinnen und Schüler benutzen zur experimentellen Untersuchung der Anschlüsse von *Do-it* den Menüpunkt „Eingänge - Ausgänge“. Die Ergebnisse der Untersuchung werden in das Arbeitsblatt 1 (siehe Anhang) eingetragen. Wie die Lampen der Ampeln den einzelnen Ausgangsleitungen zugeordnet sind, zeigt die nebenstehende Tabelle :

Bit	5	4	3	2	1	0
	Ampel B			Ampel A		
	grün	gelb	rot	grün	gelb	rot

Danach muß das Muster der einzelnen Ampelphasen beschrieben werden. Die Schülerinnen und Schüler benutzen dazu das Arbeitsblatt 2 (siehe Anhang). Die nachfolgende Abbildung zeigt das Ergebnis. Danach erfolgt die Programmierung mit *Do-it.*

Stufe	Ampel B			Ampel A			Bitmuster							
	grün	gelb	rot	grün	gelb	rot	7	6	5	4	3	2	1	0
1			X			X	O	O	O	O	I	O	O	I
2			X		X	X	O	O	O	O	I	O	I	I
3			X	X			O	O	O	O	I	I	O	O
4			X		X		O	O	O	O	I	O	I	O
5			X			X	O	O	O	O	I	O	O	I
6		X	X			X	O	O	O	I	I	O	O	I
7	X					X	O	O	I	O	O	O	O	I
8		X				X	O	O	O	I	O	O	O	I

Das Programm

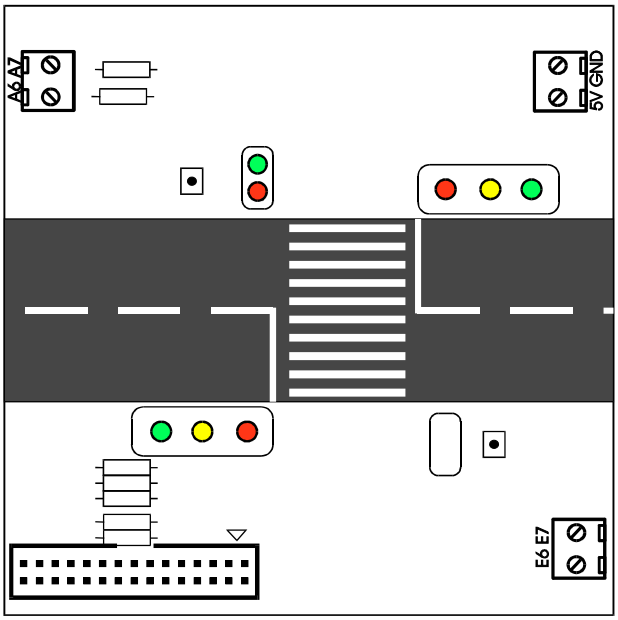
Durch ein Programm werden die einzelnen Ampelphasen fortlaufend aufgerufen. Die eingeschobenen Warte-Befehle lassen sich zeitlich verändern. Eine Anpassung an das Verkehrsaufkommen in der Realität ist darüber möglich:

```
Ampel:
PROGRAMM
 Wiederhole
    Ausgänge = OOIOOI
    Warte 2 Sekunden
    Ausgänge = OOIOII
    Warte 2 Sekunden
    Ausgänge = OOIIOO
    Warte 8 Sekunden
    Ausgänge = OOIOIO
    Warte 2 Sekunden
    Ausgänge = OOIOOI
    Warte 2 Sekunden
    Ausgänge = OIIOOI
    Warte 2 Sekunden
    Ausgänge = IOOOOI
    Warte 4 Sekunden
    Ausgänge = OIOOOI
    Warte 2 Sekunden
  Bis Tastendruck
ENDE.
```

Dieses Beispiel bietet eine Reihe von Erweiterungsmöglichkeiten insbesondere beim Einsatz einer Programmiersprache:

- Die Länge der Grünphasen kann durch Übergabeparameter verändert werden.
- Zusätzliche Ampelschaltungen, wie z.B. eine "Nachtschaltung" (gelbes Blinklicht) oder "alle Ampeln auf Rot" (für die freie Durchfahrt von Rettungsfahrzeugen) können auf Knopfdruck abgerufen werden.
- Mit den freien Schaltleitungen (Bit 6 und 7) können zusätzliche Fußgängerampeln oder Vorampeln geschaltet werden.
- Die Eingabeleitungen können mitbenutzt werden, um Knopfdruckampeln zu installieren.

3.2 Die Fußgängerampel



Bei diesem Ampelmodell befinden sich auf jeder Straßenseite 5 Lampen, die gesteuert werden müssen. Ähnlich wie bei der Kreuzungsampel müssen zunächst die Anschlüsse ermittelt werden. Die Schülerinnen und Schüler tragen das Ergebnis ihrer Untersuchung in das Arbeitsblatt 3 (siehe Anhang) ein. Die nachfolgend aufgeführte Tabelle zeigt das Ergebnis.

		digitale Ausgänge							
		7	6	5	4	3	2	1	0
Straßenampel	rot								X
	gelb							X	
	grün						X		
Fußgängerampel	rot					X			
	gelb				X				
		digitale Eingänge							
		7	6	5	4	3	2	1	0
Taster 1									X
Taster 2								X	

Die Situation, in der sich die Ampelanlage normalerweise befindet lautet: Straßenampel auf Grün und Fußgängerampel auf Rot. Sobald einer der beiden Taster gedrückt wird, schaltet die Straßenampel über Gelb auf Rot und die Fußgängerampel auf Grün. Nach einer bestimmten Wartezeit springt die Fußgängerampel wieder auf Rot und die Straßenampel wird über Gelb auf Grün geschaltet. Diese Phasen werden von den Schülerinnen und Schülern in Arbeitsblatt 4 (siehe Anhang) festgehalten. Die nachfolgende Tabelle zeigt das Ergebnis.

Stufe	F-Ampel		S-Ampel			Bitmuster								
	grün	rot	grün	gelb	rot	7	6	5	4	3	2	1	0	
1		X	X			O	O	O	O	I	I	O	O	Tasterdruck
2		X		X		O	O	O	O	I	O	I	O	
3		X			X	O	O	O	O	I	O	O	I	
4	X				X	O	O	O	I	O	O	O	I	
5		X			X	O	O	O	O	I	O	O	I	
6		X		X	X	O	O	O	O	I	O	I	I	

Das Programm

Nachdem die Ampeln in ihre Ausgangsstellung gebracht wurden, werden fortlaufend die digitalen Eingänge abgefragt. Solange die digitalen Eingänge auf "0" liegen, wurde keiner der beiden Taster gedrückt. Erst wenn das Ergebnis von "0" verschieden ist, möchte jemand die Straße überqueren, und die entsprechenden Lampensteuerungen beginnen.

Fußgängerampel:

```
PROGRAMM
 Wiederhole
    Ausgänge = OIIOO
    Wiederhole
    Bis Eingänge > 0
    Warte 1 Sekunden
    Ausgänge = OIOIO
    Warte 1 Sekunden
    Ausgänge = OIOOI
    Warte 1 Sekunden
    Ausgänge = IOOOI
    Warte 5 Sekunden
    Ausgänge = OIOOI
    Warte 1 Sekunden
    Ausgänge = OIOII
    Warte 1 Sekunden
    Ausgänge = OIIOO
 Bis Tastendruck
ENDE.
```

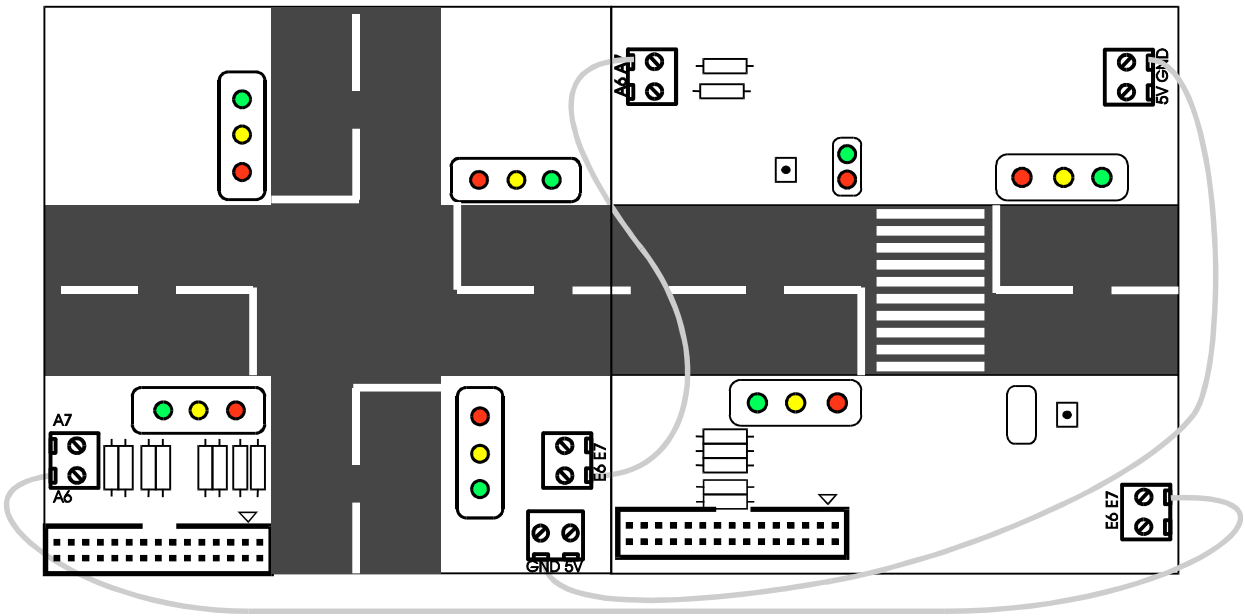
Auch hier sind einige Erweiterungen möglich:

- Um zu verhindern, daß die Taster der Fußgängerampel fortlaufend gedrückt werden, kann man nach einer Grünphase für die Fußgänger Wartezeiten einrichten. In diesen Zeiten ist es zwar möglich, die Taster zu drücken, die Fußgängerampel schaltet aber erst nach einer vorher zu bestimmenden Zeit auf Grün.

An realen Fußgängerampeln befinden sich oft Induktionsschleifen, durch die der Fahrzeugverkehr erfaßt wird. Bei diesem Modell können an die freien Eingangsleitungen Reed-Kontakte oder Schalter angeschlossen werden, mit denen die entsprechenden Impulse der Schaltung zugespielt werden. Damit sind folgende Änderungen des Steuerungsprogramms möglich:

- Die Fußgängerampel wird erst dann auf Grün geschaltet, wenn in der Fahrzeugschlange eine Lücke auftritt.
- Die Fahrzeugampel ist immer Rot. Erst wenn sich ein Auto nähert, schaltet sie auf Grün (vorausgesetzt, kein Fußgänger hat einen Taster gedrückt).

3.3. Zusammenschalten von Ampelanlagen



Beide Ampelmodelle können zu einer kompexeren Anlage zusammengeschaltet werden, wobei jedes Modell durch einen eigenen Computer gesteuert wird. Über die noch ungenutzen und an speziellen Buchsen anliegenden digitalen E/A-Leitungen können Informationen über den jeweiligen Schaltzustand ausgetauscht und in einem Programm verarbeitet werden.

Dazu werden die beiden Modelle entsprechend der Abbildung nebeneinander gelegt und mit Experimentierleitungen verbunden: Ein Ausgang der Ampelkreuzung wird mit einem Eingang der Fußgängerampel und ein Ausgang der Fußgängerampel mit einem Eingang der Ampelkreuzung verbunden. Darüber hinaus müssen die beiden Masseleitungen zusammengeschaltet werden.

Die Aufgabe

Die Programme bauen auf die bereits dargestellten Algorithmen auf. Für die Grünschaltung der Fußgängerampel muß nach dem Tasterdruck eine zusätzliche Bedingung erfüllt sein: Die entsprechende Ampelkreuzung muß auf Rot geschaltet sein, damit der Verkehrsfluß nicht doppelt unterbrochen wird.

Das Programm

Zunächst sollen die Änderungen an dem Programm der Fußgängerampel betrachtet werden. Sobald der Taster gedrückt wurde, sendet die Fußgängerampel ein Signal zur Ampelkreuzung und teilt dieser mit, daß ein Fußgänger die Straße überqueren möchte. Anschließend wird abgewartet, bis die Ampelkreuzung signalisiert, daß die entsprechende Ampel auf Rot geschaltet wurde.

Fußgängerampel:

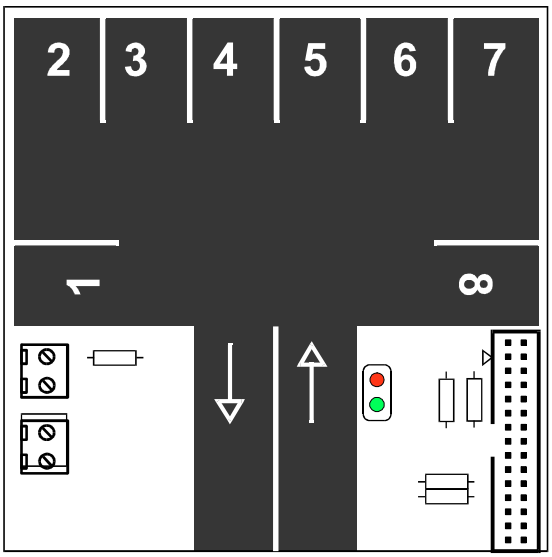
```
PROGRAMM
 Wiederhole
   Ausgänge = OIIOO
   Wiederhole
   Bis Eingänge > 0
   Ausgang 7 = I
   Wiederhole
   Bis Eingang 7 = I
   Ausgang 7 = O
   Warte 1 Sekunden
   Ausgänge = OIOIO
   Warte 1 Sekunden
   Ausgänge = OIOOI
   Warte 1 Sekunden
   Ausgänge = IOOOI
   Warte 5 Sekunden
   Ausgänge = OIOOI
   Warte 1 Sekunden
   Ausgänge = OIOII
   Warte 1 Sekunden
   Ausgänge = OIIOO
 Bis Tastendruck
ENDE.
```

Auch bei der Ampelkreuzung läuft das normale Ampelprogramm mit einer kleinen Änderung. Bevor die Ampeln für die Straße, die in der Abbildung senkrecht verläuft, auf Grün geschaltet werden, wird die Eingangsleitung 6 abgefragt. Liegt dort ein Signal an - wurde also ein Taster der Fußgängerampel gedrückt - wird ein Signal zur Fußgängerampel geschickt. Der Übergang kann für die Fußgänger freigegeben werden. Wenn am digitalen Eingang 6 "0" liegt, wird die entsprechende Ausgangsleitung zurückgesetzt und das normale Ampelprogramm fortgesetzt.

Straßenampel:

```
PROGRAMM
 Wiederhole
   Ausgänge = OOIOOI
   Warte 2 Sekunden
   Wenn Eingang 6 = I Dann
     Ausgang 6 = I
     Wiederhole
     Bis Eingang 6 = O
     Ausgang 6 = O
   EndeWenn
   Ausgänge = OOIOII
   Warte 2 Sekunden
   Ausgänge = OOIIOO
   Warte 8 Sekunden
   Ausgänge = OOIOIO
   Warte 2 Sekunden
   Ausgänge = OOIOOI
   Warte 2 Sekunden
   Ausgänge = OIIOOI
   Warte 2 Sekunden
   Ausgänge = IOOOOI
   Warte 4 Sekunden
   Ausgänge = OIOOOI
   Warte 2 Sekunden
 Bis Tastendruck
ENDE.
```

4. Im Parkhaus



Die Abbildung zeigt das Modell eines Parkhauses. Es wird über einen Vielfachstecker mit dem Interface verbunden.

Das Modellparkhaus verfügt über 8 Einstellplätze, die mit kleinen Modellautos belegt werden können. Zur Darstellung können auch kleine aus Styropor ausgeschnitte Modelle dienen. Auf getrennten Ein- und Ausfahrten gelangen die Autos zu den Parkplätzen. Unterhalb der Fahrbahn sind Reed-Kontakte montiert, die mit Magneten betätigt werden. Wird einer der kleinen mitgelieferten Magneten an der Unterseite der Modellautos montiert, wird bei der Einfahrt wie auch bei der Ausfahrt ein Schalter betätigt. Diese Schalter sind mit den digitalen Eingängen des Interfaces verbunden.

An der Einfahrt ist eine Ampel zu erkennen. Solange noch Parkplätze frei sind, steht sie auf Grün. Erst wenn alle Parkplätze belegt sind, schaltet sie auf Rot. Durch ein Computerprogramm soll angezeigt werden, wieviel Plätze belegt sind. Außerdem muß die Ampel richtig gesteuert werden.

Ein zusätzlicher Ein- und Ausgang wird jeweils über eine Steckbuchse herausgeführt. Bei einer zusätzlichen Erweiterung des Programms können sie verwandt werden.

4.1 Die Verbindung der Ein- und Ausgänge

Um eine Programmierung durchzuführen muß man genau wissen, welche Ein- bzw. Ausgänge vom Modell benutzt werden. Diese Untersuchungen können mit dem Menüpunkt *Funktionen/Eingänge-Ausgänge* in *Do-it* durchgeführt und in das Arbeitsblatt 5 (siehe Anhang) eingetragen werden. Die nachfolgende Tabelle zeigt das richtige Ergebnis.

	Ausgänge							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Ampel rot								**X**
Ampel grün							**X**	
	Eingänge							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Einfahrt								**X**
Ausfahrt							**X**	

4.2 Vorüberlegungen

Eine gute Vorbereitung für die Programmentwicklung ist die Beantwortung von 4 Fragen. Sie sind im Arbeitsblatt 6 (siehe Anhang) aufgeführt und sollten mit den Schülerinnen und Schülern zuvor erarbeitet werden. Nachfolgend ist ein ausgefüllter Fragebogen dargestellt.

1) Beschreibe den Zustand nach der Öffnung des Parkhauses!	• alle Parkplätze sind leer • die Ampel zeigt grün
2) Wann muß der Computer aktiv werden?	• immer wenn ein Auto die Ein- oder Ausfahrt benutzt
3) Was muß passieren, wenn ein Auto ins Parkhaus fährt?	• die Zahl der belegten Parkplätze muß um 1 erhöht werden • wenn es voll ist, muß die Ampel auf Rot geschaltet werden
4) Was muß passieren, wenn ein Auto das Parkhaus verläßt?	• die Zahl der belegten Plätze muß um 1 veringert werden. • wenn noch Plätze frei sind, muß die Ampel Grün anzeigen

4.3 Programmentwicklung

Entsprechend den Vorüberlegungen erfolgt die Programmentwicklung in Stufen:

In der Variablen *Zahl* soll die Anzahl der belegten Parkplätze festgehalten werden. Ihr Wert ist zu Anfang 0. Zwei Befehle definieren den Anfangszustand:

```
Zahl = 0
Ausgänge = IO
```

In einer Schleife werden fortlaufend die digitalen Eingänge abgefragt. Solange kein Fahrzeug die Ein- bzw. Ausfahrt benutzt ist der gelesen Wert 0. Erst ein von 0 verschiedener Wert führt zum Verlassen der Schleife:

```
Wiederhole
Bis Eingänge > 0
```

Benutzt ein Fahrzeug die Einfahrt, wird der Wert 1 gelesen. Entsprechend den Vorüberlegungen muß die Variable *Zahl* um 1 erhöht werden. Sind alle Parkplätze belegt, ist *Zahl* also größer als 7, muß die Ampel auf Rot geschaltet werden:

```
Zahl + 1
Wenn Zahl > 7 Dann
  Ausgänge = OI
EndeWenn
```

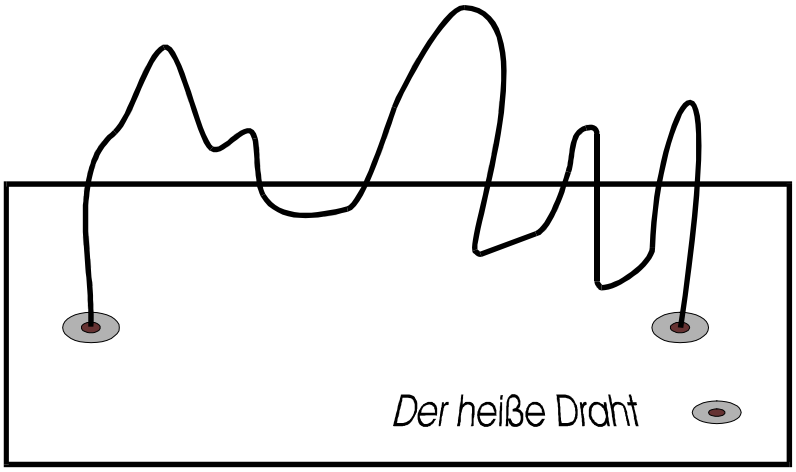
Wenn der Wert der digitalen Eingänge nicht 1 war, kann nur die Ausfahrt benutzt worden sein. In diesem Fall muß die Anzahl der belegten Plätze um 1 vermindert werden. Sind weniger als 8 Parkplätze belegt, zeigt die Ampel Grün:

```
Zahl - 1
Wenn Zahl < 8 Dann
  Ausgänge = IO
EndeWenn
```

Die einzelnen Programmteile müssen nun zu einem Gesamtprogramm zusammengesetzt werden. Das unten aufgeführte Listing enthält darüber hinaus noch eine Anweisung, die die aktuelle Belegung des Parkhauses anzeigt:

```
PROGRAMM
 Zahl = 0
 Ausgänge = IO
 Wiederhole
   Wiederhole
   Bis Eingänge > 0
   Wenn Eingänge = 1 Dann
      Zahl + 1
      Wenn Zahl > 7 Dann
         Ausgänge = OI
      EndeWenn
   Sonst
      Zahl - 1
      Wenn Zahl < 8 Dann
         Ausgänge = IO
      EndeWenn
   EndeWenn
   Schreibe Zahl" von 8 Parklätzen sind belegt.",D
 Bis Tastendruck
ENDE.
```

5. Der heiße Draht



Bei diesem Fertiggerät handelt es sich um ein Geschicklichkeitsspiel, das man häufig auf Schulfesten und ähnlichen Veranstaltungen finden kann. Die Aufgabe besteht darin, einen Ring von der Startposition zum Ziel um einen gebogenen Draht herumzuführen, ohne ihn zu berühren. Mit einem Computerprogramm kann die Zeit automatisch gemessen, jeder Fehler festgestellt und mit einer bestimmten Strafzeit geahndet werden.

Der Programmieraufwand ist so gering, daß sich dieses Beispiel sehr gut für den Einstieg ins algorithmische Problemlösen eignet.

5.1 Das Modell

Das Modell *Der heiße Draht* wird über den Sammelanschluß mit dem Interface vebunden. Damit sind sowohl die Ringe an der Start- bzw. Zielposition als auch der gebogene Draht mit den digitalen Eingängen verbunden. Für die Programmierung des Spiels benötigt man allerdings genaue Angaben über die Belegung der Eingänge. Mit dem Menüpunkt „Funktionen/Ausgänge-Eingänge“ werden sie untersucht und in das Arbeitsblatt 7 (siehe Anhang) eingetragen. Die untere Abbildung zeigt das richtige Ergebnis.

	Digitale Eingänge							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Start						**X**		
Ziel							**X**	
Draht								**X**

5.2 Die Programmentwicklung

Bei dem Spiel müssen vier Stufen durchlaufen werden:

- Die Fehlerzahl wird zu Anfang des Spiels auf 0 gesetzt und eine Stoppuhr durch die Berührung des Startrings gestartet.
- Bei jeder Berührung des Drahts wird die Fehlerzahl um 1 erhöht.
- Die Uhr wird durch die Berührung des Zielrings gestoppt.
- Schließlich erfolgt die Anzeige der gestoppten Zeit und der Berührungen.

Entsprechend diesen Stufen erfolgt die Programmentwicklung:

Die Variabel *Zahl* wird auf 0 gesetzt und der Eingang 0 wird solange abgefragt, bis er gesetzt ist. Danach wird die Uhr eingeschaltet:

```
Zahl = 0
Wiederhole
Bis Eingang 0 = I
Uhr Start
```

In einer Schleife wird fortlaufend der Eingang 1 abgefragt. Ist er gesetzt, wird die Schleife verlassen und die Uhr gestoppt:

```
Wiederhole
   ...
Bis Eingang 1 = I
Uhr Stop
```

Wenn innerhalb der Schleife der Eingang 2 gesetzt wird, wird die Variable *Zahl*, mit der die Fehleranzahl gezählt wird, um 1 erhöht.

```
Wenn Eingang 2 = I Dann
  Zahl + 1
EndeWenn
```

Am Programmende werden die Ergebnisse angezeigt:

```
Schreibe Zeit" Sekunden"
Schreibe Zahl" Fehler",D
```

Nachfolgend ist das komplette Listing zu sehen:

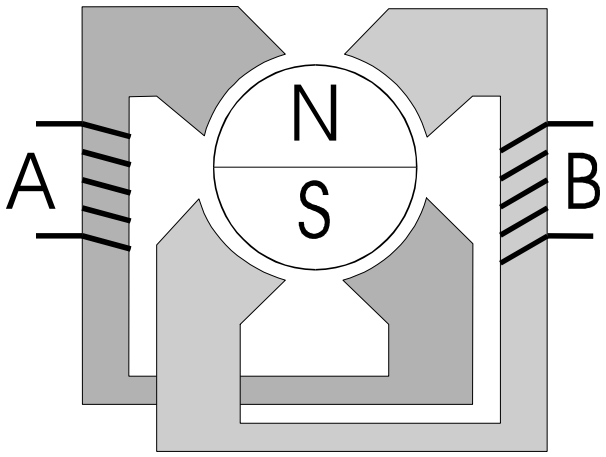
```
PROGRAMM
 Zahl = 0
 Wiederhole
 Bis Eingang 0 = I
 Uhr Start
 Wiederhole
    Wenn Eingang 2 = I Dann
       Zahl + 1
    Sonst
    EndeWenn
 Bis Eingang 1 = I
 Uhr Stop
 Schreibe Zeit" Sekunden"
 Schreibe Zahl" Fehler",D
ENDE.
```

6. Steuerung von Schrittmotoren

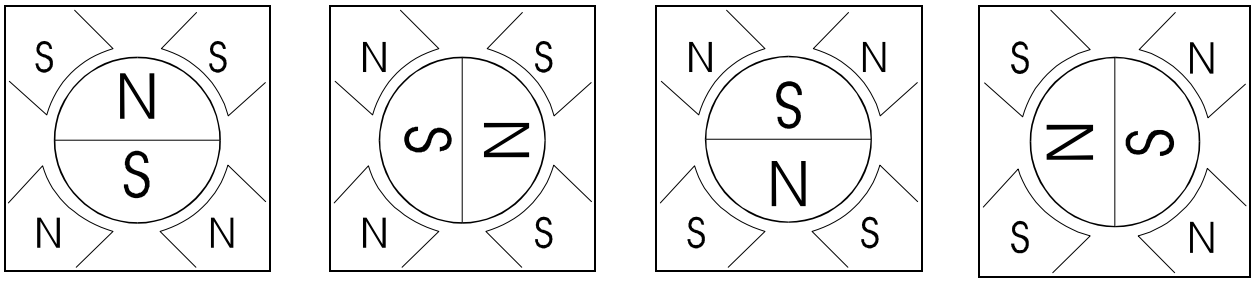
Genau genommen haben Schrittmotore viele Nachteile. Neben einem schlechten Wirkungsgrad, einem eingeschränkten Drehzahlbereich und einer geringen mechanischen Leistung ist es vor allem die komplizierte Ansteuerung, die den Einsatz häufig erschwert: Im Gegensatz zu den üblichen Gleichstrommotoren ist es nicht mit dem einmaligen Anlegen einer Spannung getan, sondern die Versorgungsspannung muß nach einem bestimmten Muster geschaltet werden.

Trotzdem haben diese besonderen Motore eine hervorragende Eigenschaft: man kann exakte Bewegungen durchführen und muß nicht durch eine komplizierte Rückmeldung die tatsächliche Position ermitteln. Damit sind auch schon die Anwendungsfälle beschrieben: Roboter, Plotter, Drucker, Nachführungssysteme für Antennen etc. Kurz: Das Einsatzgebiet der Schrittmotoren ist überall da, wo genau positioniert werden muß und nur geringe Drehmomente erforderlich sind.
Zum besseren Verständnis der komplizierten Ansteuerung vorab einige Informationen zum Aufbau von Schrittmotoren:

6.1 Aufbau von Schrittmotoren



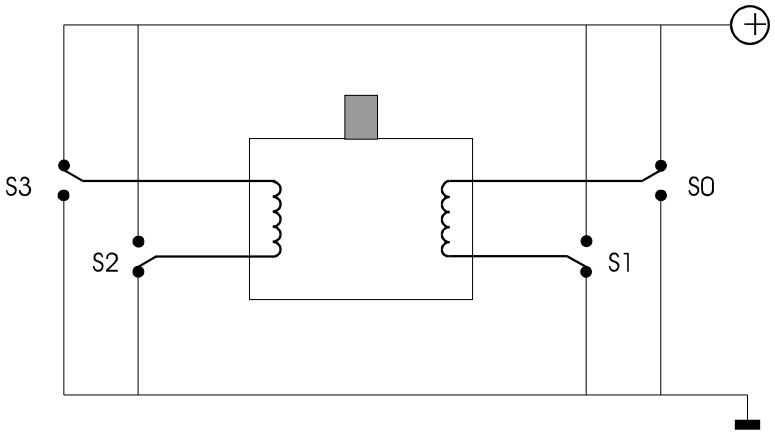
Die Zeichnung zeigt das Grundmuster eines Schrittmotors. Der Aufbau ist wesentlich einfacher als bei einem klassischen Gleichstrommotor. In einem magnetischen Feld, das durch die beiden Spulen A und B erzeugt wird (Stator), befindet sich eine Dauermagnet, der Rotor. Wird der Strom durch eine Spule umgepolt, so ändert sich auch das magnetische Feld. Bei zwei Spulen sind bei abwechselnder Umpolung vier Zustände möglich, die jeweils eine Drehung des Magneten um 90 Grad bewirken. Die unten stehende Abbildung zeigt die vier Schritte, die (in dem Beispiel) für eine Drehung um 360 Grad nötig sind.



1. Schritt 2. Schritt 3. Schritt 4. Schritt

In der Realität hat der Stator oft mehr als 4 magnetische Pole. Eine Drehung um kleinere Winkel wird dadurch möglich.

6.2 Prinzipielle Ansteuerung



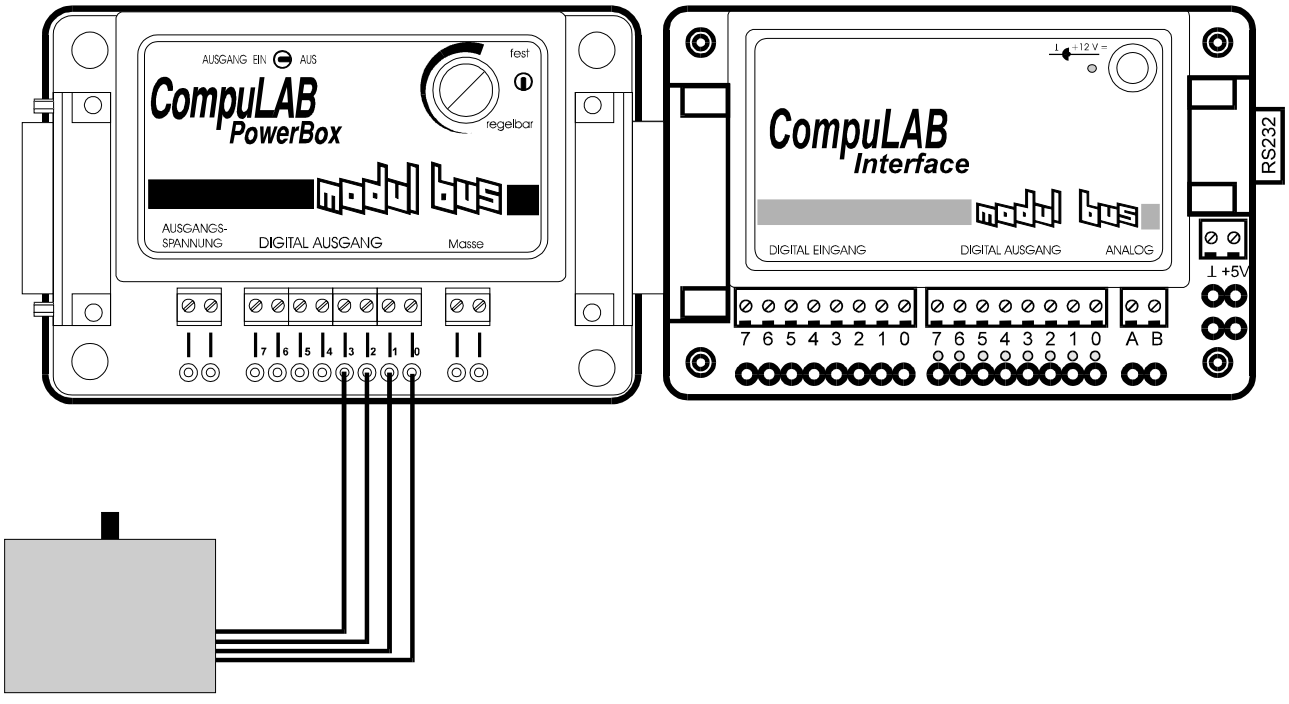
Die Abbildung skizziert die beiden Spulen innerhalb des Motors. Mit Hilfe von jeweils zwei Schaltern kann der Strom und damit das Magnetfeld umgepolt werden.

Entsprechend den weiter oben beschriebenen Schritten muß folgendes Schaltmuster eingehalten werden:

Schritt	S3	S2	S1	S0
1	+	-	+	-
2	-	+	+	-
3	-	+	-	+
4	+	-	-	+

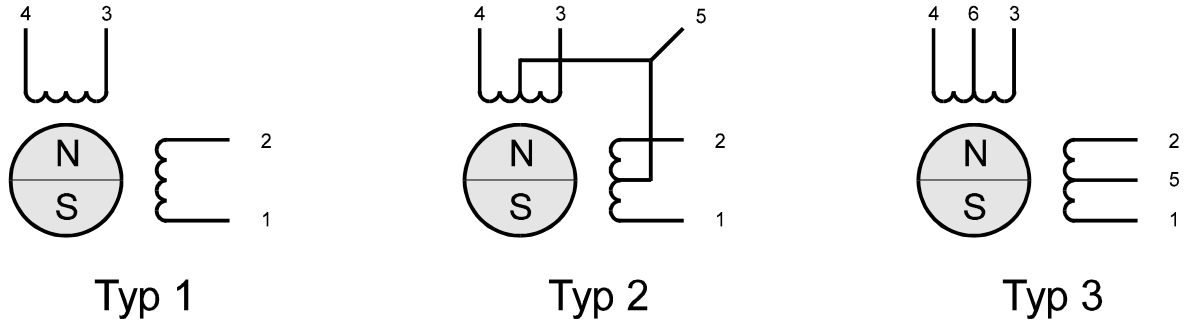
Werden die Schritte 1 bis 4 durchlaufen, dreht der Motor rechtsherum, eine umgekehrte Schrittfolge bewirkt eine Linksdrehung.

6.3 Anschluß



Zur Ansteuerung von Schrittmotoren ist das *CompuLAB* mit *PowerBox* oder das *SIOS*-Interface erforderlich.

Die richtige Verbindung der Motoranschlüsse mit den digitalen Ausgängen der *PowerBox* muß experimentell ermittelt werden. Die nachfolgende Skizze erleichtert vielleicht die Sache.



Im Inneren des Schrittmotors befinden sich zwei Spulen, deren Anschlüsse unterschiedlich herausgeführt sein können. Die in der Skizze dargestellten Typen sind am verbreitesten. Die jeweiligen Spulenenden (1 - 4) werden mit den digitalen Ausgängen verbunden.

6.4 Ein einfaches Ansteuer-Programm

Die Beschreibung der Ansteueralgorithmen erfolgt in einer Metasprache, die sowohl in eine Programmiersprache als auch in *Do-it* übertragen werden kann.

Durch Verwendung des Befehls BINAUS ist es möglich, durch die Angabe eines Musters ein bestimmtes Bit auf + ("I") oder - ("O") zu legen. Folgende Programme bewirken eine Drehung nach rechts bzw. links:

Rechts:
```
 binaus(XXXXIOIO)
 warte(0.1)
 binaus(XXXXOIIO)
 warte(0.1)
 binaus(XXXXOIOI)
 warte(0.1)
 binaus(XXXXIOOI
 warte(0.1)
```

Links
```
 binaus(XXXXIOOI)
 warte(0.1)
 binaus(XXXXOIOI)
 warte(0.1)
 binaus(XXXXOIIO)
 warte(0.1)
 binaus(XXXXIOIO)
 warte(0.1)
```

Je nach Rechnergeschwindigkeit müssen unterschiedliche Wartezeiten angegeben werden.

Da in der Regel in Schrittmotoren Mehrfachstatoren verwendet werden, wird durch dieses Programm nur eine Teildrehung erreicht. Für eine Volldrehung werden die oben aufgeführten Programme in einer Schleife mehrfach aufgerufen. Der genau Wert muß entweder experimentell ermittelt oder den genauen Daten des Motors entnommen werden.

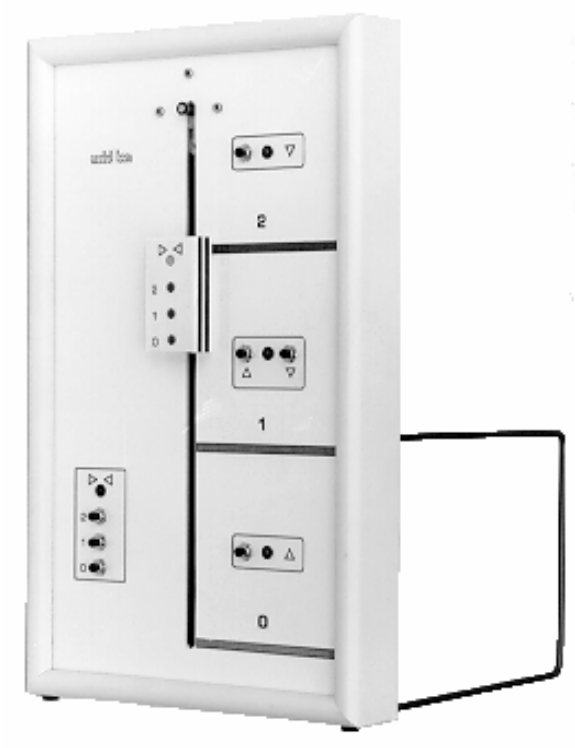
Rechtsdrehung:
```
 Wiederhole 25 mal
 rechts
```

Linksdrehung:
```
 Wiederhole 25 mal
  links
```

Darauf kann eine Prozedur RUNDE aufbauen, die als Übergabeparameter die gewünschte Anzahl der Volldrehungen hat. Ist die angegebene Anzahl positiv, sollen Rechtsdrehungen, sonst Linksdrehungen durchgeführt werden.

Runden (Anzahl):
```
 wenn Anzahl > 0 dann
  wiederhole Anzahl mal
   Rechtsdrehung
 sonst
  wiederhole Anzahl mal
  Linksdrehung
```

7. Aufzugsteuerung



In diesem Beispiel soll das Modell eines Aufzugs vom Computer gesteuert werden. Neben der Ansteuerung eines Schrittmotors, an dem der Fahrkorb hängt, müssen auch Anzeigen geschaltet und Taster abgefragt werden. Da bei der Programmentwicklung besonders Wert auf das prozedurale Prinzip gelegt werden soll, wird als Programmwerkzeug eine Programmiersprache vorgeschlagen. Die Beschreibung der Algorithmen erfolgt allerdings auf einem programmiersprachenunabhängigen Niveau.

7.1 Das Modell

Dieses Modell erfordert die Verwendung des *CompuLAB* mit *PowerBox* der des *SIOS*-Interfaces. Als Betriebsspannung muß 12 V eingestellt werden.

Bei diesem Modell hängt eine Fahrstuhlkabine an einem Seil, das um die Achse eines Schrittmotors gewickelt ist. Die Drehung des Motors bewirkt, daß sich die Kabine hinauf oder hinunter bewegt.

Mittels eines Tasters auf jeder Etage kann der Fahrstuhl angefordert werden. In einem Block auf der linken Modellhälfte sind die Taster angebracht, die die Wahlschalter in der Kabine darstellen sollen. Ein Mikroschalter am unteren Ende des Fahrschachtes dient als Referenzpunkt für die Steuerung.

Die Verbindung der einzelnen Komponenten zu den Ein- und Ausgängen des Interfaces wird untersucht und in Arbeitsblatt 8 (siehe Anhang) eingetragen. Die nachfolgende Tabelle zeigt ein mögliches Ergebnis.

Bit	Ausgänge	Eingänge
0	Schrittmotor	Mikroschalter am Boden
1	Schrittmotor	Taster im Fahrkorb Etage 0
2	Schrittmotor	Taster im Fahrkorb Etage 1
3	Schrittmotor	Taster im Fahrkorb Etage 2
4	Anzeige auf Etage 0	Taster auf Etage 0
5	Anzeige auf Etage 1	Taster auf Etage 1
6	Anzeige auf Etage 2	Taster auf Etage 2
7	Anzeige im Fahrkorb	- frei -

7.2 Drehen des Motors

Damit sich ein Schrittmotors dreht, muß an die vier Anschlüsse die Spannung in einem bestimmten Muster angelegt werden. Die Folge dieser Muster bewirkt, daß sich der Motor links- oder rechtsherum dreht:

rechts:	**links:**
IOOI	OIIO
OIOI	OIOI
OIIO	IOOI
IOIO	IOIO

Dieses Muster gilt für die Ausgänge 0 bis 3. Ein "I" bedeutet, daß das entsprechende Bit auf HIGH, ein "O" auf LOW gesetzt ist. Nach dem vierten Muster muß wieder mit dem ersten begonnen werden.

Da durch eine Rechtsdrehung der Fahrkorb nach unten und durch eine Linksdrehung nach oben bewegt wird, lautet der Name der folgenden Prozeduren *höher* und *tiefer*. Die Leitungen 4 bis 7 werden bei der Ansteuerung der Schrittmotoren nicht berücksichtigt.

Bei dem Befehl zur Ausgabe von Bitmuster BINAUS muß daher an den entsprechenden Stellen "X" eingegeben werden.

Die Ausführungsgeschwindigkeit ist in der Regel für die Ansteuerung von Schrittmotoren zu hoch. Daher muß nach jeder Ausgabe ein Wartebefehl dazwischen geschaltet werden. Die Länge der Pause kann nur experimentell ermittelt werden.

```
tiefer:
 BINAUS("XXXXIOOI")
 warte
 BINAUS("XXXXOIOI")
 warte
 BINAUS("XXXXOIIO")
 warte
 BINAUS("XXXXIOIO")
warte
```

```
höher:
 BINAUS("XXXXOIIO")
 warte
 BINAUS("XXXXOIOI")
 warte
 BINAUS("XXXXIOOI")
 warte
 BINAUS("XXXXIOIO")
warte
```

Diese Prozeduren bewirken eine Drehung der Motorachse um ca. 10 Grad und eine dementsprechend geringe Verschiebung des Fahrkorbs. Längere Fahrstrecken erfordern einen häufigen Aufruf dieser Prozeduren.

7.3 Anfahren des Nullpunktes

Am unteren Ende des Fahrschachtes befindet sich ein Mikroschalter, der mit dem Digitaleingang 7 verbunden ist und vom Boden des Fahrkorbes betätigt wird. Damit der Computer ein bestimmtes Stockwerk genau anfahren kann, wird beim Programmstart die Prozedur *Nullpunkt* aufgerufen, die diesen Referenzpunkt ansteuert. Eine absolute Positionierung ist von da an möglich.

```
Nullpunkt:
 Wiederhole
  tiefer
 bis bitein$(7)="I"
 Wiederhole 5 mal
  höher
 Standort:=0
```

Nachdem der Fahrkorb gegen den Mikroschalter gefahren ist, fährt er ein kleines Stück in entgegengesetzter Richtung zum Stockwerk 0.

7.4 Anfahren von Stockwerken

In diesem Absatz soll eine Prozedur geschrieben werden, die das direkte Ansteuern eines Stockwerkes, das als Übergabeparameter angegeben wird, ermöglicht. In Vorversuchen kann ermittelt werden, wie oft die Prozedur höher oder tiefer aufgerufen werden muß, um von einem Stockwerk zu einem nächsten zu gelangen. In diesem Beispiel wird angenommen, daß die Prozedur 137 mal aufgerufen werden muß. Damit feinere Anpassungen später leichter möglich sind, wird sie in der Variablen *Etage* gespeichert.
Die Prozedur muß sechs unterschiedlich Fälle bearbeiten:

- Wenn das Stockwerk "0" angesteuert werden soll, muß der Korb eine Etage tiefer fahren, wenn der aktuelle Standort "1" ist. Zwei Etagen müssen überwunden werden, wenn der Korb im Stockwerk "2" stand.
- Entsprechendes gilt mit entgegengesetzten Fahrtrichtung für das Stockwerk "2". Je nach aktuellem Standort muß der Fahrkorb eine oder zwei Stockwerke höher fahren.
- Etwas anders ist die Ansteuerung, wenn der Fahrkorb in das Stockwerk "1" fahren soll. Stand er vorher in "0", muß er eine Etage höher fahren, beim Standort "2" muß er eine Etage tiefer fahren.

Nach dem Verfahren des Fahrkorbs muß noch der aktuelle Standort gesichert werden. Eine Anzeige (Ausgang 7) soll leuchten, wenn der Fahrstuhl fährt. Am Anfang der Prozedur wird das Bit auf "I" und am Ende auf "O" gesetzt.

```
Stockwerk(Nummer):
 BITAUS(7,"I")
 wenn Nummer=O dann
  wenn Standort=1 dann eine Etage tiefer
  wenn Standort=2 dann zwei Etage tiefer
 wenn Nummer=1 dann
  wenn Standort=0 dann eine Etage höher
  wenn Standort=2 dann eine Etage tiefer
 wenn Nummer=2 dann
  wenn Standort=0 dann zwei Etage höher
  wenn Standort=2 dann eine Etage höher
 Standort:=Nummer
 BITAUS(7,"O")
```

7.5 Abfrage der Taster

Mit der Prozedur Stockwerk kann per Aufruf eine Etage direkt angesteuert werden. Dieser Aufruf soll aber nicht durch die Tastatur der Computers sondern durch die Taster am oder im Fahrkorb erfolgen. Sie müssen also fortlaufend abgefragt werden.

Solange kein Taster gedrückt wurde, steht der Fahrkorb. Die Taster im Fahrkorb sind mit den Eingängen 0, 1 und 2 verbunden. Werden sie gedrückt, soll das entsprechende Stockwerk direkt angesteuert werden.

Die Taster auf den einzelnen Stockwerken dienen zum Ruf des Fahrkorbes. Damit der Bediener erkennen kann, daß sein Ruf regristriert wurde, muß auf der jeweiligen Etage die Anzeige eingeschaltet werden. Im Stockwerk 1 befinden sich zwei Taster, mit denen man angeben kann, ob man nach unten oder oben fahren möchte. In unserem Beispiel sollen sie aber per Programm parallel geschaltet werden.

```
Taster:
 Wiederhole
  Solange kein Taster gedrückt wurde tue nichts
  Wenn Taster 0 gedrückt dann Stockwerk(0)
  Wenn Taster 1 gedrückt dann Stockwerk(1)
  Wenn Taster 2 gedrückt dann Stockwerk(2)
  Wenn Taster 3 gedrückt dann
   BITAUS(4,"I")
   Stockwerk(0)
   BITAUS(4,"O")
  Wenn Taster 4 gedrückt oder Taster 5 gedrückt dann
   BITAUS(5,"I")
   Stockwerk(1)
   BITAUS(5,"O")
  Wenn Taster 6 gedrückt dann
   BITAUS(6,"I")
   Stockwerk(2)
   BITAUS(6,"O")
 bis Tastendruck (Rechnertastatur)
```

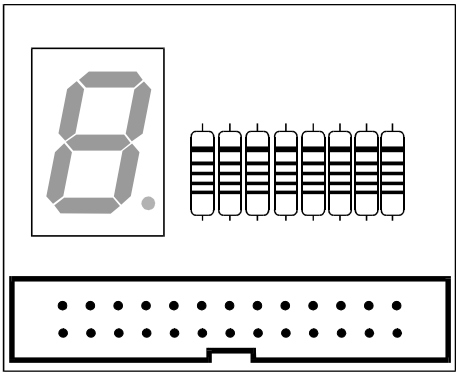
7.6 Erweiterungen

Während der Fahrt wird das Drücken eines Tasters nicht registriert. Auf den einzelnen Etagen kann man den Fahrkorb erst anfordern, wenn der Aufzug steht. Um dies zu ändern, muß in den Basisroutinen *höher* und *tiefer* eine Tastenabfrage eingebaut werden. In einer Variablen wird gespeichert, welche Taste gedrückt wurde und der entsprechende Fahrauftrag anschließend ausgeführt.

Da das zwischenzeitliche Lesen der Eingänge Zeit erfordert, müssen wahrscheinlich nach diesem "Einbau" die Warte-Zeiten verändert werden. Bei langsamen Rechnern kann sogar ohne Pausenzeiten die Ausführungsgeschwindigkeit zu gering sein. In diesen Fällen muß der langsame Befehl BINAUS durch den schnellen Befehl DEZAUS ersetzt werden. Es ist dabei allerdings zu berücksichtigen, daß bei der Ansteuerung der Motoren auch Anzeigen geschaltet werden müssen. Die entsprechenden Dezimalwerte sind bei den Ausgaben zu addieren.

Aufwendig wird das Programm, wenn die gedrückten Taster während der Fahrt ausgewertet werden sollen: Wenn der Fahrkorb in Stockwerk "0" steht und der Fahrauftrag Stockwerk 2 angegeben wird, müßte er in Stockwerk 1 halten, wenn rechtzeitig der Anforderungstaster auf Stockwerk 2 gedrückt wurde. In diesem Fall können auch die beiden Taster auf der zweiten Etage aufgewertet werden. Der Fahrkorb hält nur an, wenn er einen Fahrgast mitnehmen kann. Will dieser jedoch in die Richtung aus der der Fahrkorb gerade kommt, wird zunächst der alte Auftrag ausgeführt und dann der Fahrgast auf Stockwerk 2 abgeholt.

8. Ansteuerung einer 7-Segment-Anzeige



Den Schülerinnen und Schülern sind 7-Segment-Anzeigen aus vielen Anwendungen bekannt. Die Ansteuerung einer solchen Anzeige ist auch für den Anfänger eine überschaubare Aufgabe. Bei einer Einbindung in komplexere Anwendungsfälle wie z.B. Glücksspielautomat, läßt sich der Schwierigkeitsgrad jedoch erheblich steigern.

In diesem Kapitel werden einige interessante Anwendungen beschrieben. Die Programmentwicklung erfolgt mit *Do-it*. Bei der Verwendung einer Prozeßsprache ergeben sich weitere Möglichkeiten, die weiter unten angedeutet werden.

8.1 Unterrichtliche Ziele

Bei der unterrichtlichen Behandlung dieses einfachen Themas können unterschiedliche Schwerpunkte gesetzt werden. Nicht nur die Sicherung und Anwendung von Gelerntem sondern auch die Erarbeitung neuer Inhalte kann erfolgen.

Thema: Codierung

Die Ansteuerung der digitalen Ausgänge kann entweder durch ein Bitmuster oder einen Dezimalwert erfolgen, wobei jede Ausgabe ein bestimmtes Muster ergibt. Eine derartige eindeutige Zuordnung nennt man Codierung.

Thema: Algorithmische Strukturen

Neben der Folge als einfachste Struktur können sowohl Wiederholungen (fortlaufende Anzeige von Zufallszahlen) als auch Fallunterscheidungen (wurde eine angezeigte Zahl richtig erkannt) eingeführt oder angewendet werden. Vor allem die Einbindung in Anwendungsfälle, die über die reine Darstellung von Ziffern hinausgehen, bietet zahlreiche Möglichkeiten.

Thema: Prozedurenkonzept (nur bei Prozeßsprachen)

Die 7-Segment-Anzeige besteht aus sieben (mit Dezimalpunkt acht) Elementen, die einzeln angesteuert werden können. Werden mehrere dieser Elemente zum Leuchten gebracht, können Ziffern dargestellt werden. Die Anzeige einer bestimmten Zahl kann Teilaufgabe in einem größeren Anwendungszusammenhang sein.

8.2 Das Modell

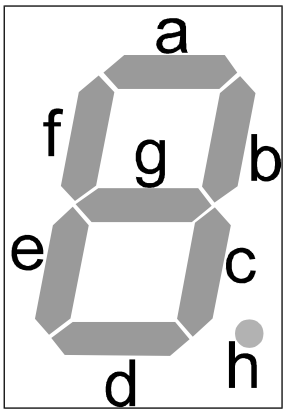
Die Anzeige besteht aus einer 4x3 cm großen Platine mit der 7-Segment-Anzeige und acht Widerständen. Der Anschluß erfolgt über das Modellanschlußkabel an den Sammelstecker des Interfaces.

8.3 Anzeige aller Muster

```
PROGRAMM
 Zahl = 0
 Wiederhole
    Ausgänge = Zahl
    Zahl + 1
 Bis  Durchläufe = 256
ENDE.
```

Alle acht Ausgangsleitungen sind einem Anzeigeelement zugeordnet. 256 unterschiedliche Muster sind demnach möglich. Sie sollen in einem ersten Programm fortlaufend angezeigt werden.

8.4 Untersuchung der Anschlüsse



Bei dem ersten Programm wurden zwar alle Muster angezeigt, aber um ein bestimmtes Muster wie z.B. eine Ziffer anzuzeigen, muß man die genaue Zuordnung von Ausgängen und den Leuchtsegmenten kennen. Mit dem Menüpunkt „Funktionen/Eingänge-Ausgänge“ in dem Programm *Do-it* kann die Zuordnung herausgefunden werden. Die Schülerinnen und Schüler tragen die Ergebnisse in das Arbeitsblatt 9 (siehe Anhang) ein und ordnen jedem Segment einem bestimmten Zahlenwert zu. Die nachfolgende Abbildung zeigt das richtig ausgefüllte Arbeitsblatt.

Segment	Ausgänge								Dez.-Wert
	7	6	5	4	3	2	1	0	
a								X	1
b							X		2
c						X			4
d					X				8
e				X					16
f			X						32
g		X							64
h	X								128

8.5 Lauflichter

Mit diesem Erkenntnissen können interessante Lauflichter oder Muster erzeugt werden. Mit etwas Phantasie kann man eine große Anzahl davon programmieren. In dem hier angeführten Beispiel läuft eine Leuchtspur "Runden". Bei einigen Geldspielautomaten findet man sie als "Pausenprogramm".

```
PROGRAMM
 Wiederhole
    Ausgänge = 3
    Ausgänge = 6
    Ausgänge = 12
    Ausgänge = 24
    Ausgänge = 48
    Ausgänge = 33
  Bis Tastendruck
ENDE.
```

8.6 Darstellung von Ziffern

Es soll nun ein Programm entwickelt werden, das alle Ziffern von 0 bis 9 aufleuchten läßt. Die Segmente, die dabei jeweils eingeschaltet werden müssen, werden zunächst aufgelistet und der dazugehörige Dezimalwert ermittelt. Dieser kann bei der Programmierung mit dem Befehl „Ausgänge“ direkt eingegeben werden (bei Prozeßsprachen wird der Befehl „dezaus“ benutzt). Die Schülerinnen und Schüler tragen das Ergebnis in das Arbeitsblatt 10 ein. Nachfolgend sind das ausgefüllte Arbeitsblatt und das entsprechende Programm aufgeführt.

	Ausgänge								
	128	64	32	16	8	4	2	1	
Ziffer	7	6	5	4	3	2	1	0	Dez.-Wert
0			X	X	X	X	X	X	63
1						X	X		6
2		X		X	X		X	X	91
3		X			X	X	X	X	79
4		X	X			X	X		102
5		X	X		X	X		X	109
6		X	X	X	X	X		X	125
7						X	X	X	7
8		X	X	X	X	X	X	X	127
9		X	X		X	X	X	X	111

```
PROGRAMM
 Wiederhole
    Ausgänge = 63
    Ausgänge = 6
    Ausgänge = 91
    Ausgänge = 79
    Ausgänge = 102
    Ausgänge = 110
    Ausgänge = 125
    Ausgänge = 7
    Ausgänge = 127
    Ausgänge = 115
 Bis Tastendruck
ENDE
```

8.7 Glücksspiel

Bei diesem Programm werden fortlaufend Zufallsmuster durch die Anzeige dargestellt. Der Anwender stoppt das Programm durch einen Tastendruck, wenn eine Ziffer korrekt angezeigt wird. Die Zeit wird dabei gestoppt. Wer bei einer korrekten Anzeige die kürzestet Zeit benötigt, hat gewonnen. Durch Veränderung der Befehlszeit (Menüpunkt Datei-Info) kann der Schwierigkeitsgrad gesteigert werden. Bei Verwendung einer Programmiersprache erfolgt eine langsamere Programmausführung durch den Einbau des „warte“-Befehls.

```
PROGRAMM
 Uhr Start
 Wiederhole
    Ausgänge = Zufallswert
 Bis Tastendruck
 Uhr Stop
 Schreibe Zeit
ENDE.
```

8.8 Weitere Möglichkeiten

Wenn als Programmierwerkzeug eine Programmiersprache eingesetzt wird, ergeben sich weitere Möglichkeiten. Hier ein paar Anregungen:

Würfel

Fast alle Programmiersprachen bieten die Möglichkeit, Zufallszahlen zu generieren. In der Variablen WERT soll eine natürliche Zufallszahl zwischen 1 und 6 gespeichert und durch die 7-Segmentanzeige dargestellt werden:

```
Würfel:
```

```
  Wert:= Zufallszahl zwischen 1 und 6
  Anzeige der Ziffer (Wert)
```

Bei jedem Aufruf dieser Prozedur erscheint eine andere Zufallszahl.

Zahlenraten

Eine Zufallszahl zwischen 0 und 999 wird durch drei schnell aufeinanderfolgenden Ziffern dargestellt. Der Anwender muß die Zahl eingeben, die er meint erkannt zu haben. Durch den Computer erfolgt die Mitteilung, ob er die richtige Zahl erkannt hat:

```
Raten:
  Einer:= Zufallszahl zwischen 0 und 9
  Zehner:= Zufallszahl zwischen 0 und 9
  Hunderter:= Zufallszahl zwischen 0 und 9

  Anzeige der Ziffer (Einer)
  warte(0,2)
  Anzeige der Ziffer (Zehner)
  warte(0,2)
  Anzeige der Ziffer (Hunderter)
  warte(0,2)
  Eingabe: Zahl
  Wenn Zahl = Einer + Zehner*10 + Hunderter*100
    dann "Du hast die richtige Zahl geraten"
    sonst "Du hast leider falsch geraten"
```

Reaktionstest

Nachdem eine bestimmte Ziffer aufleuchtet, muß diese Zahl möglichst schnell eingegeben werden. Der Computer soll bei diesem Reaktionstest die Zeit zwischen der Anzeige und der Eingabe der richtigen Zahl messen.

```
Test:
  Pause:= Zufallszahl zwischen 2 und 10
  Wert:= Zufallszahl zwischen 0 und 9
  warte(Pause)
  Anzeige der Ziffer (Wert)
  Starte Stoppuhr
  Wiederhole
    Warte auf Tastendruck
  bis gedrückte Taste = Wert
  Zeit:= gelesener Stoppuhrwert
  Anzeige: "Du hast nach " Zeit "Sekunden richtig reagiert!"
```

Für Programmierfreaks

Fast alle Spiele kann man z.B. für Schulfeste etc. in umfangreichen Programmen einbetten, die zusätzliche Informationen auf den Bildschirm schreiben. Es können z.B. Ranglisten geführt wird oder etwaige Gewinnquoten ermittelt werden. Die grafische Aufbereitung ist dabei ein lohnendes und für Schülerinnen und Schüler sicher interessantes Thema.

Die Anzeige kann darüberhinaus benutzt werden, um an Modellen bestimmte Informationen zu geben. An dem Aufzug kann z.B. das Stockwerk angezeigt werden oder im Modell "Parkhaus" kann die Anzahl der freien Stellplätze ausgewiesen werden.

Zusätzliche Möglichkeiten ergeben sich, wenn sich die Anzeige auf andere Anschlüsse des Interfaces bezieht. So kann z.B. die Anzahl der angesteuerten digitalen Eingänge zur Anzeige gebracht werden. Sicher sind viele mögliche Anwendungen noch nicht dargestellt. Für Ideen und Anregungen sind wir immer dankbar und würden uns freuen, wenn wir sie bei einer Überarbeitung des Heftes mit einfließen lassen könnten.

9. MATRIX

Unter einer Matrix versteht man die rechteckige Anordnung von Elementen in Zeilen (waagerecht) und Spalten (senkrecht). In dem hier dargestellten Fall handelt es sich um eine Leuchtdiodenmatrix, die unterschiedliche Muster anzeigen soll. Diese Form von Lichtspielen findet man z.B. bei Fernseh-Shows, Pop-Konzerten und bei fast allen Fahrgeschäften auf Jahrmärkten.

In dieser Darstellung soll die Ansteuerung der Leuchtdioden und die Ausgabe interessanter Lichtmuster beschrieben werden.

Neben den bereits erwähnten Zusammenhängen kommen matrixförmige Anordnungen in unterschiedlichsten Bereichen vor. Das Schachbrett ist eine Matrix, bei der jedes Feld durch die Angabe eines Buchstabens und einer Zahl bestimmt werden kann. Komplette Partien können somit in codierter Form wiedergegeben werden.

Im Mathematikunterricht tritt diese Anordnung häufig auf. Schon das karierte Papier kann als Matrix aufgefaßt werden, und die ersten grafischen Darstellungen nutzen die Rechenkästchen als Orientierungshilfen. Abbildungen in der Ebene folgen, wobei jedem Punkt zwei Zahlen (X- und Y-Koordinate) zugeordnet werden. Funktionen und andere Zusammenhänge können so grafisch dargestellt werden. Selbst in der höheren Mathematik sind die rechteckigen Anordungen bei der Matrizenrechnung wiederzufinden.

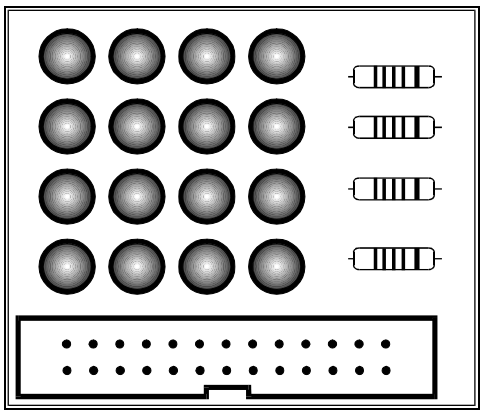
Auch bei der Beschäftigung mit dem Computer tauchen diese Strukturelemente auf. Die Darstellungen im Text- und im Grafikbildschirm beruhen auf einem Gitternetz von Punkten, die angesprochen werden können. Die genaue Beobachtung eines Zeichens auf dem Bildschirm ergibt, daß es aus einzelnen Punkten dargestellt wird, die gitterförmig angeordnet sind. Bei der Speicherung von Daten werden diese häufig matrixförmig strukturiert. Dies gilt insbesondere dann, wenn jedem Datum zwei Kriterien zugeordnet werden. Im Verlauf dieser Darstellung werden Beispiele dafür gegeben.

9.1 Ziele

Wie die Ausführungen zeigen, kommen matrixförmige Anordnungen in unterschiedlichen Zusammenhängen vor. Ziel der hier beschriebenen Unterrichtsreihe ist der spielerische Ungang mit diesem elementaren Strukturelement.

Das Leuchtdiodenfeld kann sowohl mit *Do-it* als auch auf Programmiersprachenebene angesprochen werden. Dabei kann als Ziel die Erarbeitung von algorithmischen Grundstrukturen und Programmiertechniken verfolgt werden. Durch die Vorgabe anwenderfreundlicher Routinen zur Ansteuerung einzelner LED´s, kann die Anforderung an die Lerngruppe herabgesetzt werden. Die Programmierung komplexer „Lightshows“ mit komfortablen Eingaberoutinen und die Möglichkeit der Archivierung ist wohl eher etwas für fortgeschrittene Programmierer.

9.2 Die Hardware



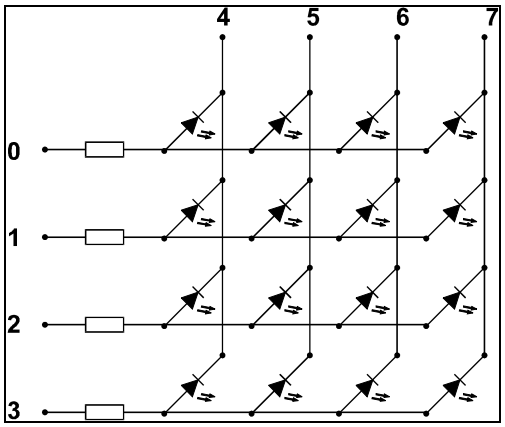
Das Modell besteht aus 16 Leuchtdioden, die in einer 4X4 Matrix angeordnet sind. Der Anschluß erfolgt über das Modellanschlußkabel an den Sammelstecker des *CompuLAB*.

9.3 Untersuchung der Anschlüsse

Mit Hilfe des Menüpunktes “Eingänge-Ausgänge“ werden mit Ausgang 0 beginnend alle Ausgänge gesetzt. Die Beobachtungen werden in das Arbeitsblatt 11 (siehe Anhang) eingetragen. Das Ergebnis zeigt die nachfolgende Tabelle.

Ausgang	Beobachtung
0	Die erste Zeile leuchtet
1	Die zweite Zeile leuchtet
2	Die dritte Zeile leuchtet
3	Die vierte Zeile leuchtet
4	Die erste Spalte (von rechts) wird ausgeschaltet
5	Die zweite Zeile leuchtet
6	Die dritte Zeile leuchtet
7	Die vierte Zeile leuchtet

Zusammenfassend kann man feststellen, daß die Ausgänge 0-3 den Zeilen und die Ausgänge 4-7 den Spalten zugeordnet sind. Dabei gilt für jede Leuchtdiode, daß sie nur dann leuchten kann, wenn der Ausgang für die Zeile in der sie sich befindet **ein**geschaltet und der Ausgang für die entsprechende Spalte **aus**geschaltet ist.



Die nebenstehende Abbildung zeigt die physikalische Erklärung: Die Zeilen sind mit den Anoden der LED´s und die Spalten mit den Kathoden verbunden. Eine LED leuchtet nur, wenn die jeweilige Zeile HIGH und die Spalte LOW sind.

Zeilenmuster

Damit eine Zeile mit einem beliebigen Muster aufleuchtet, muß zunächst die gewünschte Zeile durch Einschalten eines Ausgangs (0-3) ausgewählt werden. Ergebnis: Alle Leuchtdioden der ausgewählten Zeile leuchten. Durch das Einschalten eines der Ausgänge 4-7 kann dann jeweils eine Leuchtdiode ausgeschaltet werden.

Spaltenmuster

Zunächst sollten alle Ausgänge von 4-7 gesetzt werden. Dann wird die Spalte, die das Muster dargestellen soll, ausgeschaltet. Noch leuchtet keine Leuchdiode auf. Erst wenn einer der Ausgänge von 0-3 gesetzt wird, leuchtet die entsprechende Leuchdiode auf.

Zur Übung versuchen die Schülerinnen und Schüler die Muster, die in Arbeitsblatt 12 (siehe Anhang) aufgeführt sind, an der Matrix darzustellen. Ein beliebiges Muster der gesamten Matrix kann aus naheliegenden Gründen nicht durch eine einzelne Ausgabe erfolgen. Alle eingeschalteten Leuchtdioden müssen sich entweder in einer Spalte oder einer Zeile befinden. Daher läßt sich die Aufgabe d so einfach nicht lösen. Es kann nur dadurch realisiert werden, daß die einzelnen Spalten oder Reihen nacheinander in schneller Folge zur Anzeige gebracht werden. Diese Methode ist allgemein üblich und wird auch als Multiplex-Verfahren bezeichnet.

Die Ausführungsgeschwindigkeit von *Do-it* erlaubt leider keine stehenden Bilder, auch wenn die Befehlszeit auf 0 gesetzt wird. Verwendet man hingegen eine Programmiersprache so erscheinen dem Betrachter beliebige Muster als stehende Bilder.

9.4 Erstellung von Filmen

Ein Film setzt sich aus einzelnen Bildern zusammen, die in rascher Reihenfolge einander abwechseln. In unserem Fall bestehen die Bilder aus einzelnen Mustern, die in einem Programm fortlaufend aufgerufen werden.

```
PROGRAMM
 Wiederhole
    Ausgänge = OOOOOOOI
    Ausgänge = OOOOOOIO
    Ausgänge = OOOOOIOO
    Ausgänge = OOOOIOOO
 Bis  Tastendruck
ENDE.
```

In diesem Beispiel leuchten die Zeilen der Reihe nach auf. Damit die Abfolge der einzelnen Bilder rasch erfolgt, muß die Befehlszeit (Menüpunkt „Programm Info“) auf 0 gesetzt werden.

Nach diesem Muster können Filme „produziert“ werden. Bei der Erstellung längerer Filme bietet sich ein arbeitsteiliges Vorgehen an. In kleinen Gruppen werden einzelne „Szenen“ erstellt. Indem man Szene 1 lädt, die Muster in die Zwischenablage kopiert, Szene 2 lädt und die Muster aus Szene 1 aus der Zwischenablage dort einfügt usw. können längere Filme zusammengesetzt werden.

9.5 Programmentwicklung auf Programmiersprachenebene

Nicht nur die höhere Ausführungsgeschwindigkeit kann ein Grund für den Einsatz von Prozeßsprachen auf der Basis einer Programmiersprache sein, sondern dieses Beispiel bietet sich besonders für eine klar strukturierte prozedurale Prorammentwicklung an. Nachfolgend werden in der bereits bekannten Metasprache Prozeduren entwickelt, die aufeinander aufbauen und letzlich zu komfortablen Routinen führen.

Löschen der Anzeige

Noch bevor mit der Darstellung einzelner Muster begonnen wird, soll eine Prozedur beschrieben werden, die das „Löschen“ der Anzeige bewirkt. Dies ist deshalb sinnvoll, weil die letzte Ausgabe gespeichert bleibt und durch den Aufruf einer solchen Prozedur ein definierte Anfangszustand wieder hergestellt wird.

Alle Leuchtdioden leuchten, wenn die Zeilenanschlüsse auf LOW und die Spaltenanschlüsse auf HIGH liegen. Die jeweils umgekehrten Polaritäten bewirken das „Löschen“ der Anzeige. Außerdem ist bei diesem Zustand ein störungsfreis Umschalten unterschiedlicher Anzeigen möglich.

aus:
Ausgänge 240

Die Zeilen sind mit den Ausgängen 0-3 und die Spalten mit den Ausgängen 4-7 verbunden. Die Ausgabe des Wertes 240 bewirkt das „Löschen“ der Anzeige.

Setzen eines Punktes

Bei dieser Routine soll ein Punkt, der durch eine Zeile und Spaltenangabe definiert wird, zum Leuchten gebracht werden. Eine Prozedur *Punkt* muß beide Angaben als Übergabeparameter haben.

Zu jeder Zeile gehört ein bestimmter Zeilenwert, der zunächst bestimmt werden muß. Die erste Zeile ist mit dem Bit 0 verbunden, der auszugebende Wert ist 1. Bei der zweiten Zeile muß der Wert 2 ausgegeben werden, bei der dritten der Wert 4, der vierten der Wert 16. Der mathematische Zusammenhang lautet: Zeilenwert = $2^{(Zeile-1)}$

Bei der Berechnung des Spaltenwerts muß davon ausgegangen werden, daß alle Anschlüsse auf HIGH liegen. Soll ein bestimmter Punkt leuchten, muß der entsprechende Wert von 240 abgezogen werden. Bei der

ersten Spalte muß also 16, bei der zweiten 32, der dritten 64 und der vierten Spalte 128 in Abzug gebracht werden. Der mathematische Zusammenhang lautet: Spaltenwert = 240 - $2^{(Spalte+3)}$

```
Punkt (Zeile, Spalte):
   Zeilenwert =  2^(Zeile-1)
   Spaltenwert = 240 – 2^(Spalte+3)
   Ausgänge (Zeilenwert+Spaltenwert)
```

In dieser Prozedur werden zunächst aus der Zeilen- bzw. Spaltenangabe die entsprechenden Werte berechnet, addiert und an die digitalen Ausgänge ausgegeben.

Nachfolgend sollen einige Beispiele für den Einsatz dieser Prozedur gezeigt werden. Sie sind als Anregung für eigene Programmentwicklungen zu verstehen.

```
Laufzeile(Zeile):
   Zähler von 1 bis 4
   führe aus
   Punkt(Zeile, Zähler)
   warte 0,1 Sekunden
```

Durch die Prozedur *Laufzeile* soll ein Leuchtpunkt in einer angegebenen Zeile von der ersten bis zur letzten Position „laufen“.

```
Laufzeilen
   Zähler von 1 bis 4
   führe aus
   Laufzeile(zähler)
```

Durch die Prozedur *Laufzeilen* sollen alle 4 Zeilen von dem Leuchtpunkt durchlaufen werden.

Ähnliche Prozeduren lassen sich für die spaltenweise Anzeige schreiben.

Anzeige eines Bildes

Wie bereits erläutert wurde, ist es nicht möglich, ein komplettes Bild der gesamten Matrix auf einmal anzuzeigen. Es muß aus einzelnen Spalten aufgebaut werden.

```
Spaltenmuster (Spalte,Wert):
   Spaltenwert = 240 – 2^(Spalte+3)
   Ausgänge (Spaltenwert+Wert)
```

Die Prozedur Spaltenmuster bringt eine komplette Spalte zur Anzeige. Als Übergabeparameter muß die gewünschte Spalte und ein dem Muster entsprechender Wert angegeben werden. Wie aus der Spalte der entsprechende Spaltenwert berechnet wird, wurde bereits erläutert.

```
Bild:
   Bilddaten(1) = 5
   Bilddaten(2) = 4
   Bilddaten(3) = 7
   Bilddaten(4) = 8
   Wiederhole
      Zähler von 1 bis 4
         Führe aus
         SpaltenmusterZähler,
   Bilddaten(Zähler)
```

In den ersten Zeilen der Prozedur *Bild* werden die Bilddaten definiert. Sie befinden sich in inizierten Variablen, die in der nachfolgenden Schleife fortlaufend aufgerufen werden. Dadruch entsteht ein stehendes Bild mit einem beliebigen Muster auf der gesamten Matrix.

Programmierung eines Films

Ein Film besteht aus einer raschen Abfolge verschiedener Bilder. Ein Bild besteht aus 4 Spaltenmuster, die je in einem Speicher festgehalten werden. Bei einem Film müssen für jedes Bild diese Speicher vorhanden sein.

```
Film:
   Filmdaten(1,1-4):=1, 0, 0, 0
   Filmdaten(2,1-4):=2, 3, 0, 0
   Filmdaten(3,1-4):=4, 4, 7 0
   Filmdaten(4,1-4):=8, 8, 8. 15

   Wiederhole
    Zähler_1 von 1 bis 4
      Führe aus
      Zähler_2 von 1 bis 30
        Führe aus
        Zähler_3 von 1 bis 4
          Führe aus
          spaltenmuster (k,filmdaten(Zähler_1,Zähler_3))
   bis Tastendruck
```

In der Prozedur *Film* werden zunächst die Filmdaten definiert.

In dem hier dargestellten Beispiel besteht der Film aus 4 Bildern. Die dazugehörigen Daten werden in dem zweidimensionalen Datenfeld *Filmdaten* gespeichert. In der Prozedur *Film* befinden sich vier ineinander verschachtelte Schleifen. Die innere Schleife (Zähler _3) bringt die einzelnen Bilder (Zähler_1) zur Anzeige. Zähler_2 definiert, wie lange bzw. wie oft ein einzelnes Bild angezeigt werden soll. Durch Veränderungen an dieser Stelle kann ein Film schneller oder langsamer laufen.

Durch Zähler_1 wird von einem Bild auf das folgende umgeschaltet. Die Länge des Films kann damit verändert werden. Die entsprechenden Bildspeicher müssen allerdings zuvor definiert werden. Die äußere Schleife sorgt dafür, daß die Szene solange wiederholt wird, bis eine Taste gedrückt wird.

Ausblick

An dieser Stellen sollen einige Anregungen für Programmerweiterungen gegeben werden. Die Definition eines Bildes durch eine Eingabe von Werten ist sehr umständlich und bedarf einiger Überlegungen. Diese Aufgaben kann auch durch ein Programm erledigt werden. Die Eingabe eines Musters erfolgt entweder durch eine Zeichenkette oder - was aufwendiger in der Programmierung ist - es erscheint die Matrix auf dem Bildschirm. Mit Hilfe der Tastatur oder durch Mausklick können dann die einzelnen Leuchdioden eingeschaltet werden. Der Computer liest die so gesetzten Daten aus und speichert die entsprechenden Daten.

Es ist sehr ärgerlich, wenn gut geplante „Filme“ nach dem Verlassen des Programms nicht mehr vorhanden sind. Routinen zum Speichern und Laden von Filmen sind Aufgaben, die sich anschließen.

10. Codekartenleser

In vielen Portemonnaies befinden sich neben Geldscheinen und Münzen zunehmend kleine rechteckige Karten: Kreditkarte, Bahncard oder Telefonkarte eröffnen dem Besitzer unterschiedliche Möglichkeiten. Sie werden in automatische Lesegeräte geschoben und in der Regel von einem Computer ausgewertet. Manchmal weisen die Daten, die mit einer vorhandenen Datei verbunden werden, auf konkrete Personen hin, als Beispiel sei hier die Kreditkarten erwähnt. In einigen Fällen sind nur allgemeine Merkmale gespeichert. So wird durch eine Telefonkarte lediglich nachgewiesen, daß der Telekom ein Kredit gewährt wurden, der nach und nach „abtelefoniert“ wird.

Daß das Speichern von Merkmalen nicht immer ganz unproblematisch ist, soll in diesem Beispiel gezeigt werden. Dabei werden die technischen Voraussetzungen angesprochen und Algorithmen zur Auswertung entwickelt.

Ein konkreter unterrichtlicher Anlaß kann eine Untersuchung der Besucher der Schulbücherei sein. Zwar soll nicht herausgefunden werden, ob bestimmte Schülerinnen und Schüler die Bücherei besuchen und wenn ja wie oft, sondern es sollen lediglich Merkmale der Besucher festgehalten werden. Sie erlauben die Beantwortung z.B. folgender Fragen:

- Sind Mädchen eher an dem Lesen von Büchern interessiert als Jungen?
- Befinden sich unter den Besuchern mehr Brillenträger als im Durchschnitt? (Theorie: Lesen schadet den Augen!)
- Ist die Haarfarbe der Leser eher hell (blond) oder eher dunkel?
- Wie ist die Altersstruktur der Leser?
- Gibt es lokale Schwerpunkte? (Frage: Kommen die Leser häufiger aus bestimmten Ortsteilen?)
- Gibt es Klassen, die die Bücherei besonders oft besuchen?

Die Reihe der interessanten Fragen ließe sich sicher beliebig fortsetzen.

10.1 Das Modell

Als Kartenlesegerät wird der Codekartenleser eingesetzt. Er besteht aus 8 Fototransistoren und einer integrierten Beleuchtung. Mit Hilfe von Codekarten (Pappkarten mit bestimmten Lochmustern) können Daten in den Computer eingegeben werden.

Der Anschluß erfolgt über den Vielfachstecker an den Sammelanschluß des Interfaces.

10.2 Muster werden gelesen

Die acht Fototransistoren sind über den Vielfachstecker mit den digitalen Eingängen verbunden. Trifft Licht auf einen Fototransistor, ist der entsprechende Eingang gesetzt. Deckt man ihn z.B. durch einen Finger ab, liegt er also im Schatten, ist dieser Eingang nicht gesetzt. Diese Funktion kann mit dem Menüpunkt *Funktionen\Eingänge - Ausgänge* untersucht werden.

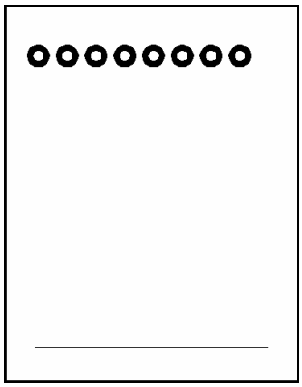
```
PROGRAMM
 Wiederhole
    Schreibe Eingänge
 Bis Tastendruck
ENDE.
```

Soll das gelesene Muster wie in einem Protokoll dargestellt werden, muß ein kleines Programm geschrieben werden, das die Eingänge liest und als Muster auf den Bildschirm schreibt.

```
PROGRAMM
 Wiederhole
    Schreibe "Eingangsmuster "Eingänge,B
    Schreibe "Eingangswert "Eingänge,D
    Schreibe " "
 Bis Tastendruck
ENDE.
```

Jedes Muster kann auch einem bestimmten Wert zugeordnet werden. Den Zusammenhang zwischen Muster und Wert können die Schülerinnen und Schüler anhand dieses Programms erkunden.

10.3 Die Codekarte



Für den Codekartenleser sind entsprechende Karten erhältlich, die in den Schlitz des Lesegerätes eingeschoben werden. Die darunterliegenden lichtempfindlichen Stellen sind durch Kreise markiert, die mit einem herkömmlichen Bürolocher ausgestanzt werden können. Damit man die Stanzung des Lochs gut plazieren kann, ist es günstig, den Bodendeckel des Lochers zu entfernen. Durch die unterschiedlichen Lochmuster kann die Karte programmiert werden.

Entsprechend der weiter oben dargestellten Aufgabenstellung sollen nun Merkmale des Kartenbesitzes durch entsprechende Lochungen markiert werden.

10.4 Merkmale codieren und Codekarten auswerten

Ein Merkmal ist das Geschlecht des Kartenbesitzers. Da diese Frage entweder mit *männlich* oder *weiblich* beantwortet werden kann, reicht eine Stelle für die Codierung aus. Die Zuordnung ist wahlfrei, muß aber innerhalb einer Klasse einheitlich sein. Die Vereinbarungen werden in das Arbeitsblatt 13 (siehe Anhang) eingetragen. Die nachfolgende Abbildung zeigt die korrekt ausgefüllte erste Zeile dieses Arbeitsblattes.

Position	Merkmale	Kennzeichnung
0	*Geschlecht*	● = *weiblich* ○ = *männlich*

```
Wenn Eingang 0 = I Dann
   Schreibe "Geschlecht = weiblich"
Sonst
   Schreibe "Geschlecht = männlich"
EndeWenn
```

Durch das nebenstehende Computerprogramm wird der Eingang 0 ausgewertet. Liegt die Karte in dem Lesegerät und wird das Programm gestartet, erscheint auf dem Bildschirm entweder die Meldung „Geschlecht = weiblich“ oder „Geschlecht = männlich“.

In ähnlicher Weise können die Fragen, ob der Kartenbesitzer eine Brille trägt oder welche Haarfarbe er hat, (eher hell oder dunkel) beantwortet werden.

Etwas aufwendiger sind Fragen, die mehr als zwei Antworten zulassen. Bei der Frage nach dem Alter bildet man vielleicht vier Gruppen und die Karte soll Auskunft darüber geben, zu welcher Gruppe der Kartenbesitzer gehört. Bei vier Antwortmöglichkeiten benötigt man 2 Stellen für die Codierung. Nachfolgend ein Vorschlag für die Einteilung in Gruppen und die Zuordnung zu dem Lochmuster:

Gruppe	Alter	Muster
1	über 15 Jahre	● ●
2	14 -15 Jahre	● ○
3	13 - 12 Jahre	○ ●
4	unter 12 Jahre	○ ○

```
Wenn Eingang 4 = I Dann
  Wenn Eingang 3 = I Dann
    Schreibe "Alter = über 15 Jahre"
  Sonst
    Schreibe "Alter = 14 oder 15 Jahre"
  EndeWenn
Sonst
  Wenn Eingang 3 = I Dann
    Schreibe "Alter = 12 oder 13 Jahre"
  Sonst
    Schreibe "Alter = unter 12 Jahre"
  EndeWenn
EndeWenn
```

Das Programm enthält eine geschachtelte Fallunterscheidung. Zunächst wird ein Position abgefragt. Die vier möglichen Anworten werden damit in zwei Gruppen eingeteilt. Ist in unserem Fall die Eingang 4 gesetzt, so muß der Kartenbesitzer entweder der Gruppe 1 oder 2 angehören. Über die eindeutige Zuordnung entscheidet nun der Eingang 3. Ist er zusätzlich gesetzt, so ist der Kartenbesitzer älter als 15 Jahre; ist sie nicht gesetzt, so ist das Alter zwischen 14 und 15.

Ist hingegen der Eingang 4 nicht gesetzt, so gehört der Kartenbesitzer den Gruppen 3 oder 4 an. Die genaue Ermittlung erfolgt ähnlich dem oben beschriebenen Verfahren.

Sind noch mehr Antwortmöglichkeiten erforderlich, so muß die Anzahl der Stellen vergrößert werden. Eine weitere Stelle ermöglicht bereits 8 Antworten. In unserem Fall soll die Klasse damit codiert werden:

Gruppe	Klasse	Muster
1	Primarstufe	○ ○ ○
2	5	○ ○ ●
3	6	○ ● ○
4	7	○ ● ●

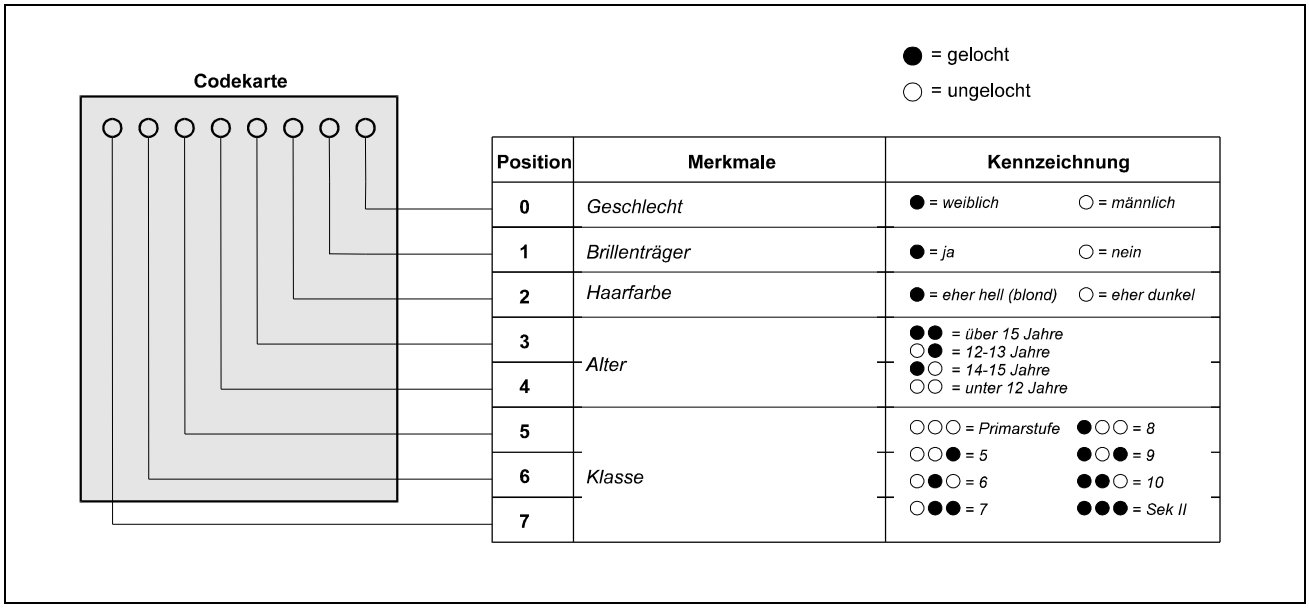
Gruppe	Klasse	Muster
5	8	● ○ ○
6	9	● ○ ●
7	10	● ● ○
8	Sek II	● ● ●

Auch bei diesem Programm könnten wieder geschachtelte Fallunterscheidungen verwandt werden. Hier soll jedoch eine andere Methode eingesetzt werden. Jeder Fall wird einzeln untersucht. Liegt das entsprechende Muster an, wird ein entsprechender Kommentar auf den Bildschirm geschrieben.

```
Wenn Eingänge = OOOXXXXX Dann
   Schreibe "Klasse = Primarstufe"
```

```
EndeWenn
Wenn Eingänge = OOIXXXXX Dann
  Schreibe "Klasse = 5"
EndeWenn
Wenn Eingänge = OIOXXXXX Dann
  Schreibe "Klasse = 6"
EndeWenn
Wenn Eingänge = OIIXXXXX Dann
   Schreibe "Klasse = 7"
EndeWenn
Wenn Eingänge = IOOXXXXX Dann
   Schreibe "Klasse = 8"
EndeWenn
Wenn Eingänge = IOIXXXXX Dann
   Schreibe "Klasse = 9"
EndeWenn
Wenn Eingänge = IIOXXXXX Dann
   Schreibe "Klasse = 10"
EndeWenn
Wenn Eingänge = IIIXXXXX Dann
   Schreibe "Klasse = Sekundarstufe II"
EndeWenn
```

Die nachfolgende Abbildung zeigt ein Beispiel einer kompletten Codierung. Das dazugehörige Programm befindet sich auf der beiliegenden Diskette. Wenn eine richtig codierte Karte in das Lesegerät geschoben und das Programm gestartet wird, erscheint ein kurzer Steckbrief des Kartenbesitzers.



10.5 Erweiterungen

Damit nicht jedes Mal das Programm erneut gestartet werden muß, um einen neuen Steckbrief anzuzeigen, soll dieser Vorgang automatisiert werden. Dazu muß man den Vorgang, daß eine Karte eingeschoben wird, genauer untersuchen: Zunächst sind alle Eingänge gesetzt, da keine Karte einliegt. Schiebt man jedoch eine Karte ein, so werden durch den vorderen Steg kurzzeitig alle lichtempfindlichen Schalter abgedunkelt, also alle Eingänge gelöscht. Erst danach liegt die richtige Codierung an. Das Herausziehen der Karte werden zunächst wieder alle Eingänge gelöscht um dann wieder gesetzt zu werden. Diese Vorgänge kann ein Computerprogramm so auswerten, daß beim Einschieben einer neuen Karte automatisch ein neuer Steckbrief erscheint.

Wenn als Programmierwerkzeug eine Programmiersprache eingesetzt wird, ergeben sich weitere Möglichkeiten. Für jedes Merkmal bzw. für jede Gruppe wird ein Speicher eingerichtet. Damit wird festgehalten, wie oft eine entsprechende Karte mit dem entsprechenden Merkmal eingesteckt wurde.

Bei dem Unterichtsprojekt erhält man nach einer Woche sicher aussagekräftige Daten. Die Interpretation bietet allerdings viel Diskussionsstoff in der Klasse.

10.6 Fragen zum Datenschutz

Die codierten Karten geben keine unmittelbare Auskunft auf den Besitzer. Dennoch enthalten sie eine Reihe von persönlichen Merkmalen, die nicht selten eindeutige Rückschlüsse auf den Besitzer erlauben. Soll die Problematik der Reanonymisierung deutlicher hervortreten, sollte man sich mehr auf äußerliche Merkmale konzentrieren (Größe, Gewicht, Augenfarbe ...).

Auch in diesem Beispiel sind noch weiter Informationen durch Kopplung von Merkmalen herauszuholen. Wenn man das Alter und die Klasse vergleicht, kann man schnell Schülerinnen und Schüler identifizieren, die „überaltert“ sind. Die Ursachen können vielfältig sein, Krankheit, Vorschule oder vielleicht mußten sie auch eine Klasse wiederholen. Ob diese besondere Schülergruppe die Bücherei wie „normale“ Mitschülerinnen und Mitschüler die Bücherei benutzen, ist sicher eine interessante Frage, die zeigt, wie schnell Probleme des Datenschutzes relevant sein können.

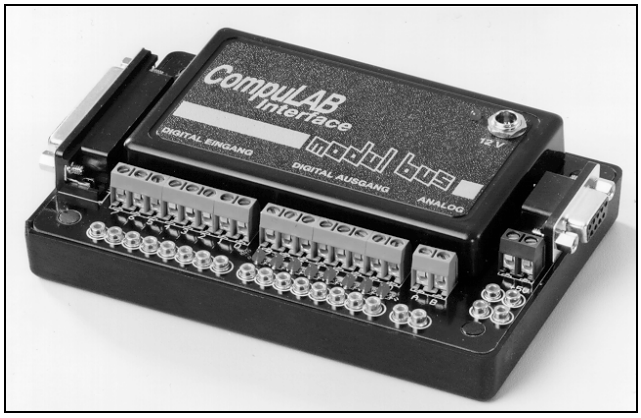
Anhang:
Zusammenstellung der verwendeten Hard- und Softwarekomponenten

1. Interface

CompuLAB - Interface

Das Interface besitzt folgende Ein- und Ausgänge:

- 8 digitale Eingänge (TTL-kompatibel), überspannungsfest und verpolungssicher
- 8 digitale Ausgänge (TTL-HC kompatibel), die Leuchtdioden und ähnliche Kleinverbraucher direkt antreiben können. Der Zustand der Ausgänge wird durch Leuchtdioden angezeigt
- 2 analoge Eingänge mit einem Meßbereich von 0-5 Volt



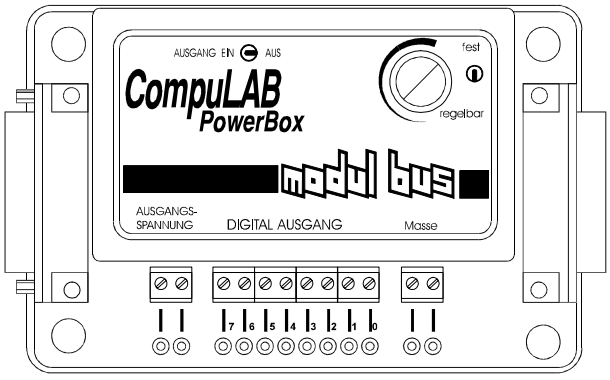
Die Ein- und Ausgänge werden an folgenden Anschlüssen herausgeführt:

- Sammelstecker mit allen Ein- und Ausgängen;
- Anschlußklemmen, in die mit Hilfe eines Schraubendrehers Leitungen oder elektronische Bauteile befestigt werden können;
- 2mm Steckbuchsen, diese hochwertigen Stecksysteme findet man immer mehr bei elektronischen Meßgeräten. Über entsprechende Verbindungsstücke sind auch 4mm Stecker anschließbar.

Diese universellen Anschlüsse ermöglichen sowohl den anwendungsorientierten Einsatz, wie zum Beispiel in der Grundbildung, als auch den experimentellen Einsatz, wie z.B. im Fachunterricht Technik.

CompuLAB-PowerBox

Die *PowerBox* verstärkt die digitalen Ausgänge des *CompuLAB*. Bei einer einstellbaren Ausgangsspannung von 5-12V oder der festen Ausgangsspannung des Netzteils NT101 wird ein Strom von >600 mA/Ausgang geliefert. Die Gesamtleistung wird durch das verwendete Netzteil begrenzt.



Die *PowerBox* wird über den 25 SubD-Stecker direkt an das *CompuLAB* angedockt. Der Anschluß über ein serielles Standardkabel ist allerdings auch möglich.

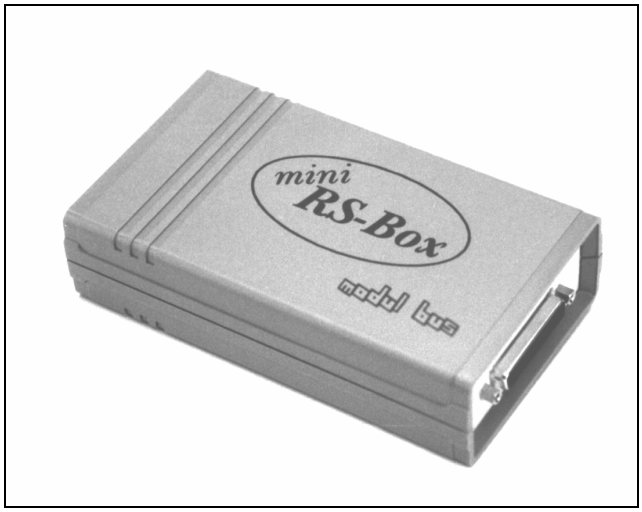
CompuLAB Set

Dieses Set enthält alles, was zu einem vollständigen Computerlabor notwendig ist:

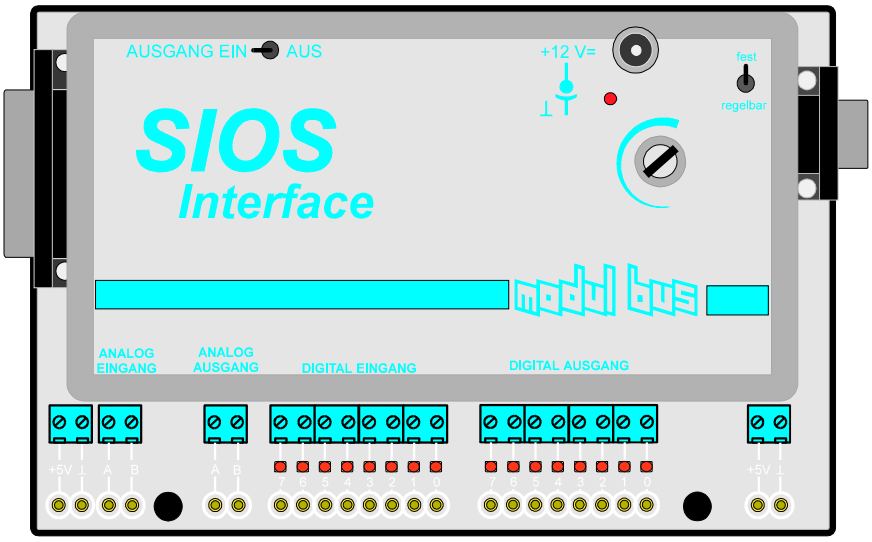
- ***CompuLAB* - Interface** mit Netzteil und Verbindungskabel zur seriellen Schnittstelle eines Computers. Natürlich ist dieses Interface vollkommen softwarekompatibel zur *miniRS-Box*!
- Experimentiermaterial, mit den zusätzlichen Komponenten können zahlreiche Versuche durchgeführt werden.
- *Do-it*, ein WINDOWS-Programm mit vielen integrierten Meßgeräten und einer universellen Programmierumgebung.
- Handbuch mit über 60 Seiten, auf denen zahlreiche Experimente und Anregungen für eigene Versuch stehen.
-

miniRS-Box

Als Alternative kann auch die *miniRS-Box* eingesetzt werden. Dieses universelle Schülerinterface besitzt ähnliche Leistungsmerkmale wie das *CompuLAB* Interface mit *PowerBox*. Alle Ein- und Ausgänge sind an einem 25 pol SubD-Stecker herausgeführt, über den viele Funktionsmodelle mit dem unviersellen Modellanschlußkabel angeschlossen werden können. Für die Ansteuerung eigener Versuchsaufbauten ist die Expermimentier-Box besonders geeignet. Sie stellt neben 2mm Buchsen dem Anwender auch Schraubanschlüsse für alle Ein- und Ausgänge zur Verfügung.



SIOS



Natürlich können auch alle Versuche mit dem *SIOS*-Interface durchgeführt werden. Dieses leistungsfähige, universell einsetzbare Interface schließt alle Leistungsmerkmale vom *CompuLAB* Interface mit *PowerBox* und der *miniRS-Box* ein und bietet darüber hinaus noch analoge Ausgänge und 4 analoge Eingänge mit einer höheren 10 Bit Auflösung.

Da alle Ein- und Ausgänge an 2 mm Steckbuchsen, an Schraubanschlüssen und an dem 25 pol SubD-Stecker herausgeführt sind, sind keine weiteren Komponenten erforderlich.

2. Netzteile

CompuNet

Dieses preiswerte Steckernetzteil dient als Stromversorgung für das *CompuLAB* - Interface. Der Anschluß erfolgt über einen einfachen Hohlstecker.

NT101

Dieses peiswerte Steckernetzteil dient als Stromversorgung für das *CompuLAB* - Interface, die *miniRS-Box* und das *SIOS*-Interface.

Die *PowerBox* erfordert eine Stromversorgung mit einer höheren Leistung. Dieses Netzteil ist dafür besonders geeignet. Es besitzt ein stabiles Metallgehäuse und liefert eine Ausgangsspannung von 12-15 V. Die Einstellung der Sapnnung erfolgt mit Hilfe eines kleinen Schraubendrehers von der Seite des Gerätes aus. Der Ausgangsstrom beträgt max. 2A.

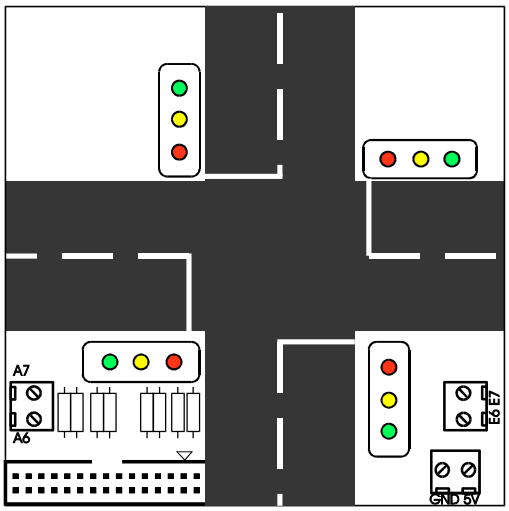
3. Funktionsmodelle

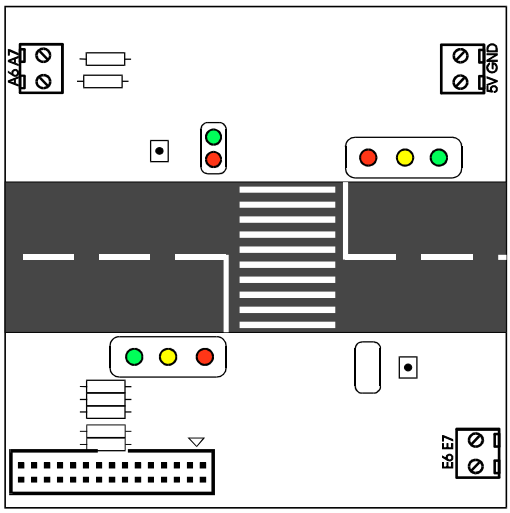
Ampelanlagen

Die Modelle bestehen aus bedruckten Platinen, bei denen die Ampeln durch Leuchtdioden dargestellt werden. Es stehen zwei Ausführungen zur Verfügung:

- *Kreuzungsampel* mit 4 Ampeln
- *Fußgängerampel* mit Drucktastern

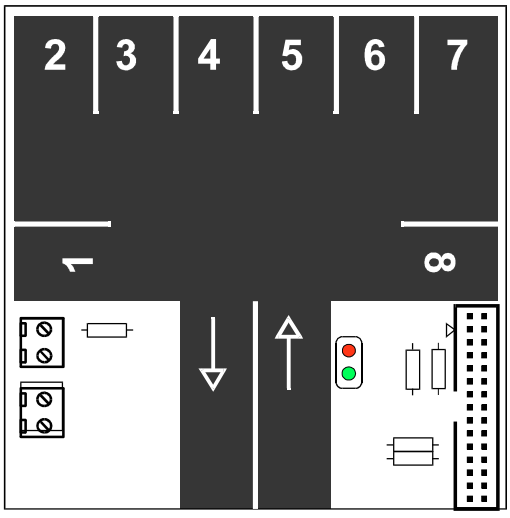
Bei beiden Modellen wurden zusätzliche Ein- und Ausgangsleitungen herausgeführt, die eine Zusammenschaltung mehrerer Anlagen ermöglichen.





Parkhaus

Die Vorgänge in einem *Parkhaus* mit automatisch arbeitender Ein- und Ausfahrtregelung sollen simuliert werden. Das Modell hat 8 Einstellplätze und eine Ampel an der Einfahrt. Über Sensoren (Reed-Kontakte) werden die Fahrzeugbewegungen registriert.



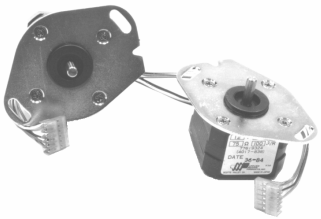
Der heiße Draht

Bei diesem Fertiggerät handelt es sich um ein Geschicklichkeitsspiel, das man häufig auf Schulfesten und ähnlichen Veranstaltungen finden kann. Die Aufgabe besteht darin, einen Ring von der Startposition zum Ziel um einen gebogenen Draht herumzuführen, ohne ihn zu berühren. Mit einem Computerprogramm kann die Zeit automatisch gemessen, jeder Fehler festgestellt und mit einer bestimmten Strafzeit geahndet werden.

Der Programmieraufwand ist so gering, daß sich dieses Beispiel sehr gut für den Einstieg ins algorithmische Problemlösen eignet.

Schrittmotor *)

12 V-Typ mit 96 Steps/Umdre-hungen



Aufzug *)

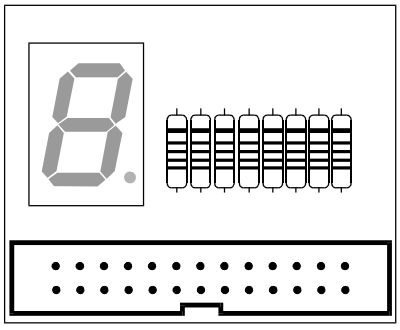
Durch einen Schrittmotor wird in einem Modellhaus ein Aufzugkorb auf- und abwärts bewegt. Der Schrittmotor kann so gesteuert werden, daß der Aufzug per Tastendruck bestimmte Etagen anfährt. Das Modell wird über einen Mehrfachstecker angeschlossen und hat die Ausmaße ca. 435 (Höhe) x ca. 265 (Breite).

) Diese Modelle erfordern die *PowerBox

7-Segment-Anzeige

Die digitalen Ausgänge des *Modul-Bus* werden mit den Balken einer Leuchtdiodenanzeige verbunden. Für dieses einfache Modell können Prozeduren entwickelt werden, die Ziffern darstellen.

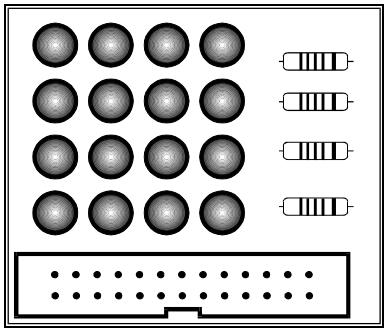
Der Anschluß erfolgt über das Modellanschlußkabel.



Matrix

Mit der *Matrix* können Lichtbilder und Lauflichter programmiert werden. Das Modell besteht aus 16 Leuchtdioden, die in einer 4x4 Matrix angeordnet sind.

Der Anschluß erfolgt über das Modellanschlußkabel.



Codekartenleser

Dieses Gerät besteht aus 8 Fototransistoren und einer integrierten Beleuchtung. Mit Hilfe von Codekarten (Pappkarten mit bestimmtem Lochmuster) können Daten in den Computer eingegeben werden.

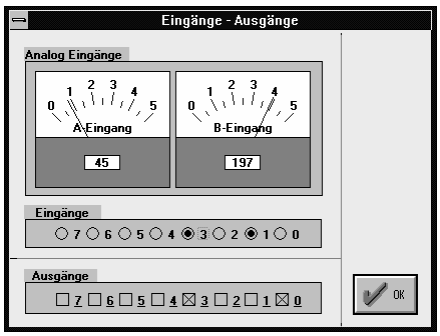
Codekarten:
Bei diesem Artikel handelt es sich um fertig zugeschnittene und markierte aber nicht gelochte Codekarten aus festem Karton. Sie finden z.B. in dem Projekt *Supermarkt* Verwendung.

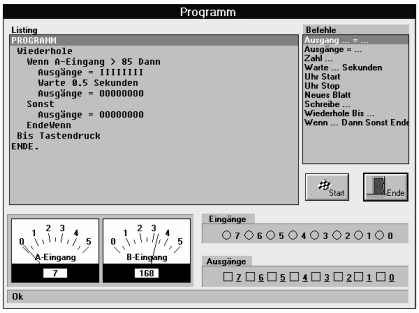
Modellanschlußkabel

Zum Anschluß von Modellen, aber auch von komplizierten Versuchsaufbauten.

4. Software

Do-it





Dieses Programm stellt unter WINDOWS eine universelle Werkzeugkiste zum Messen und Auswerten zur Verfügung. Im einzelnen sind folgende Module integriert:

- Eingänge - Ausgänge
- Zeit-Schreiber
- Koordinaten-Schreiber
- Eingangs-Schreiber

Daneben enthält *Do-it* eine Programmierumgebung, bei der nicht nur Ausgänge gesetzt und analoge wie digitale Eingänge gelesen werden können, sondern in der sich auch Schleifen und Fallunterscheidungen programmieren lassen.

Dieses Programm wird jedem *CompuLAB*-Set als Einzellizenz beigelegt. Es ist auch als Klassenraumlizenz für dieses System erhältlich. Darüber hinaus ist auch eine Version für die *miniRS-Box* bzw. *Compact-Box* erhältlich.

Jetzt ist auch *Do-it* für DOS erhältlich. Nähere Informationen auf Anfrage.

Prozeßsprachen

Bei den Prozeßsprachen handelt es sich um Makrobefehle auf Programmiersprachenebene, mit denen die Grundfunktionen des Interface-Systems ausgeführt werden können. Sie bestehen u.a. aus einem Befehlswort und Übergabeparameter(n).

Als Prozeduren bzw. Funktionen stehen u.a. zur Verfügung:

zum Setzen der digitalen Ausgänge:
DEZAUS, BINAUS, BITAUS

zum Lesen der digitalen Eingänge:
DEZEIN, BINEIN, BITEIN

zum Lesen der analogen Eingänge:
AEIN, UEIN

Die Beispiele, die sich auf der Diskette zu diesem Buch befinden, setzen die Programmiersprache (Pascal oder Comal) und Prozeß-Pascal bzw. Prozeß-Comal voraus. Weitere Versionen auch für andere Rechner können auf Anfrage angeboten werden.

Literaturangaben

[1] Bund-Länder-Komission für Bildungsplanung und Forschungsförderung (Hrsg.): Rahmenkonzept für die Informationstechnische Bildung in Schule und Ausbildung. Reihe „Materialien zur Bildungsplanung“. Bonn: BLK, 1984

[2] Der Kultusminister des Landes Nordrhein-Westfalen (Hrsg.): „Vorläufige Richtlinien zur Informations- und Kommunikationstechnologischen Grundbildung in der Sekundarstufe I“. Eine Schriftenreihe des Kultusministeriums. Düsseldorf 1990

[3] Der Kultusminister des Landes Nordrhein-Westfalen (Hrsg.): „Unterrichtsempfehlungen für den Wahlpflichtunterricht Informatik - Hauptschule“. Eine Schriftenreihe des Kultusministeriums. Düsseldorf 1994

[4] Bund-Länder-Komission für Bildungsplanung und Forschungsförderung (Hrsg.): Gesamtkonzept für die informationstechnische Bildung in Schule und Ausbildung. Heft 16 der Reihe „Materialien zur Bildungsplanung“. Bonn: BLK, 1987

[5] Landesinstitut für Schule und Weiterbildung (Hrsg.): „Orientierungshilfe zur Ausstattung von Allgemeinbildenden Schulen mit Hard- und Software“ Beratungsstelle für Neue Technologien. Soest 1993 - Seite 34