Handbuch

zum Praktikum

Mechatronische Systeme

an der

Friedrich-Alexander-Universität Erlangen-Nürnberg

Version 2013

Hinweis: Dieses Handbuch soll einen Einstieg in die notwendigen Themenbereiche für das Praktikum Mechatronische Systeme ermöglichen. Die Zusammenstellung der technischen Umsetzungsvarianten erhebt jedoch keinen Anspruch auf Vollständigkeit.

Inhaltsverzeichnis

1 Methodische Herangehensweise an die Aufgabenstellung

Die Entwicklung eines technischen Systems unter den gegebenen Randbedingungen erfordert eine systematische und methodische Herangehensweise an die Aufgabenstellung. Der Entwicklungsprozess wird dadurch überschaubar und transparent. Als prinzipielle Vorgehensweise wird der Ablauf in Bild 1 empfohlen.

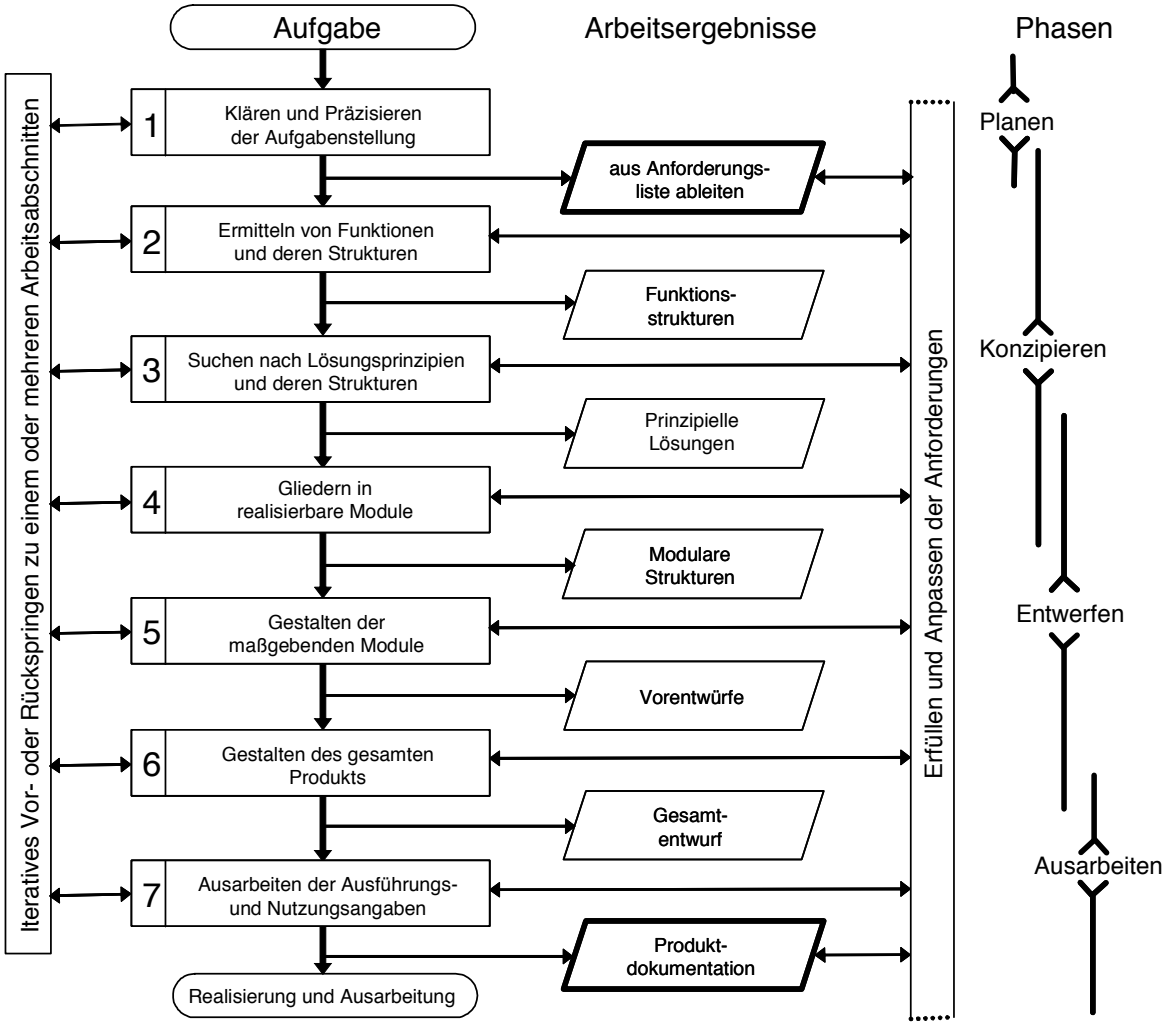

Bild 1: Allgemeines Vorgehen beim Entwickeln nach VDI-Richtlinie 2221.

In einem ersten Schritt wird die Gesamtfunktion in Teilfunktionen niedrigerer Komplexität zerlegt, z.B.:

- Antrieb
- Sensorik / Objekterkennung
- Positionierung des Gesamtsystems

Das Verknüpfen dieser Teilfunktionen über Stoff-, Energie- und Signalflüsse führt zu einer Funktionsstruktur.

Im nächsten Schritt sind für die Teilfunktionen Lösungen zur Realisierung zu suchen. Diese lassen sich zur Strukturierung und in Vorbereitung auf eine Bewertung in einem morphologischen Kasten zusammenfassen (Bild 2).

Lösung / Funktion	1	2	3	…	n	n+1
Hindernis erkennen	Tastsensor 	Licht-schranke 	…		…	…
Bewegung übertragen	Räder 	Ketten 	…		…	…
Fahrzeug lenken	synchron 	Omni-direktional 	…		…	…
…	…	…	…		…	…

Bild 2: Morphologischer Kasten für die Teilfunktionen am Beispiel eines Roboterfahrzeugs.

Mit Hilfe des morphologischen Kastens lassen sich Gesamtlösungen zusammenstellen, indem Pfade gebildet werden. Diese Gesamtlösungen sind hinsichtlich der zu erfüllenden Funktionalität und der zu beachtenden Randbedingungen zu bewerten, um eine geeignete Lösung auswählen zu können.

2 Mikrocontroller-Board

2.1 Aufbau

Für das Praktikum „Mechatronische Systeme“ werden vom Lehrstuhl für Zuverlässige Schaltungen und Systeme (LZS) mehrere Hardware-Module angeboten. Mit diesen Platinen sind verschiedene Funktionalitäten realisierbar. Durch geeignete Kombination der Module und die entsprechende Programmierung der Mikrocontroller kann ein Großteil der geforderten Funktionalität ohne eigens zu realisierende Schaltungen implementiert werden. In Bild 3 ist ein Blockdiagramm eines möglichen Aufbaus zu sehen.

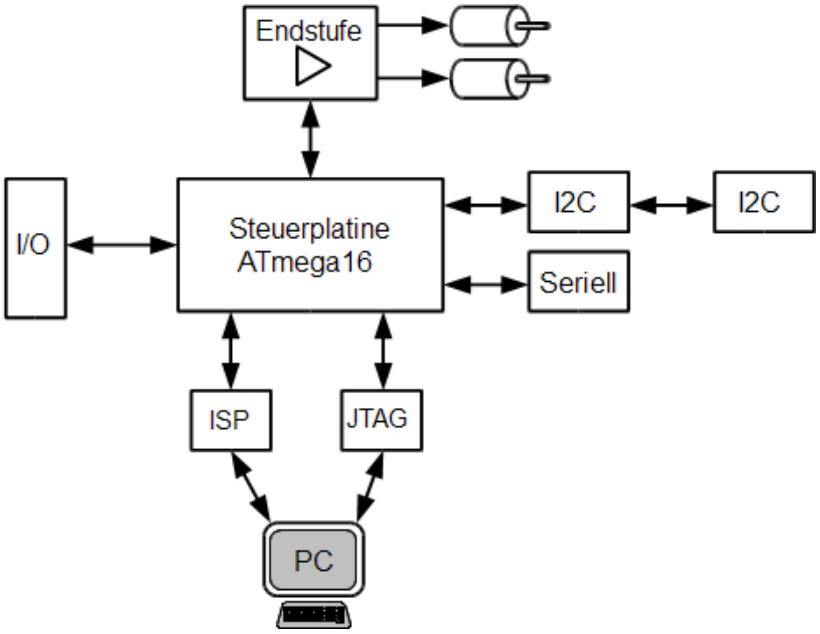

Bild 3: Aufbau des Mikrocontroller-Boards als Blockschaltbild.

Die zentrale Platine ist die Steuerplatine, auf der sich ein ATmega16 Mikrocontroller [1] für die Programmierung der gewünschten Funktionalität befindet. Diese kann über Steckverbinder mit den anderen Modulen verbunden werden. Die Ein- und Ausgänge des Mikrocontrollers sind ebenfalls, nach Funktionalität geordnet, auf verschiedene Steckverbinder geführt. In der folgenden Tabelle sind die Steckverbinder mit einer möglichen Verwendung aufgelistet. Tabelle 1 führt die zur Verfügung gestellten Steckverbinder und deren Anschlussmöglichkeiten auf (alle PINs können unabhängig davon als digitale I/Os verwendet werden):

Tabelle 1: Steckverbinder Steuerungsplatine.

Schnittstelle	Funktion
I2C	Sensoren, Portverteilung
Seriell	RS232, RS454
Endstufe	PWM, Motortreiber
ISP	Programmcode übertragen, Fuses setzen, SPI
JTAG	Programmcode übertragen, Debuggen
I/O	Analoge und digitale I/Os, Interrupt

2.2 Steuerplatine

Die Steuerplatine besteht aus einem Mikrocontroller mit minimaler Beschaltung, einer Spannungsversorgung und Steckverbindern. Über diese Steckverbinder werden die Ein- bzw. Ausgänge des Mikrocontrollers externer Hardware zur Verfügung gestellt. Mit speziellen Steckverbindern kann der Mikrocontroller konfiguriert und das ausführbare Programm geladen werden. Die Betriebsspannung der Steuerplatine ist an jedem Steckverbinder verfügbar, um externe Schaltungen zu versorgen.

2.2.1 Spannungsversorgung

Bild 4 zeigt das Schaltbild des Spannungsreglers auf der Steuerplatine. Als Eingang dient ein externes Netzteil, das über den Steckverbinder X7 mit der Steuerplatine verbunden wird. Das externe Netzteil kann wahlweise eine Gleich- oder Wechselspannung bereitstellen, die zwischen 7,5 V und 12 V liegen muss. Der Brückengleichrichter B1 dient im Falle der Versorgung mit einer Gleichspannung als Verpolschutz, ansonsten zur Gleichrichtung der negativen Halbwelle der Wechselspannung. Die Versorgungsspannung VCC (5 V) wird durch den Längsregler IC2 [2] erzeugt und durch die Kondensatoren C4, C5 und C6 stabilisiert.

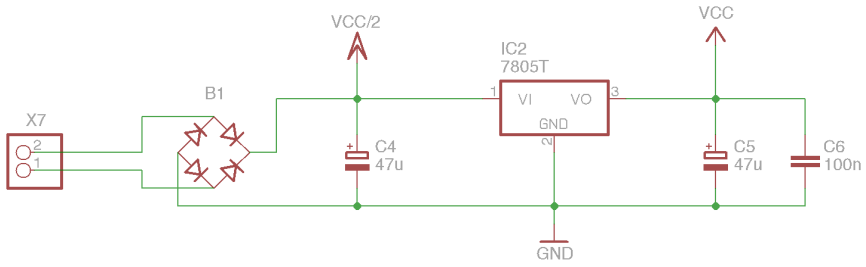

Bild 4: Spannungsversorgung.

Bei der Versorgung von externer Hardware durch die Steuerplatine ist die maximale Stromstärke von 1 A und die maximale Verlustleistung am Spannungsregler (IC2) zu berücksichtigen. Da diese von der zugeführten Spannung abhängig ist, kann hier kein Richtwert angegeben werden. Die Temperatur am Baustein sollte 80 °C nicht überschreiten. Notfalls muss eine Berechnung der maximal möglichen Verlustleistung über die in Anhang A vorgestellte Methode durchgeführt werden.

2.2.2 Mikrocontroller

Als Mikrocontroller wird der ATmega16 aus der AVR-Familie der Firma Atmel eingesetzt. AVR-Mikrocontroller basieren auf einer 8-bit RISC-Architektur (Reduced Instruction Set Computer). Der Prozessortakt wird intern nicht geteilt, was bei einem 16 Mhz Quarz einen Befehlsdurchsatz von bis zu 16 Millionen Befehlen pro Sekunde ermöglicht. Alle AVRs sind mit einer ISP-Schnittstelle (In-System-Programming) ausgestattet. Damit kann der Programmspeicher des Mikrocontrollers in der Zielhardware neu beschrieben werden. Zum Debuggen des Programms und zum Beschreiben des Programmspeichers hat der ATmega16 eine JTAG Schnittstelle.

Der ATmega16 bietet folgende Funktionalität:

- 131 Instruktionen
- 16 MHz max. Taktfrequenz
- 16 kByte ISP Flash-Speicher
- 512 Byte EEPROM
- 1 kByte SRAM
- 32 programmierbare digitale Ein-/Ausgänge
- 2 8-bit Timer/Counter (Zeitgeber/Zähler)
- 1 16-bit Timer/Counter
- 4 PWM (Puls-Weiten-Modulation) Ausgänge
- 8 10-bit Analog-Digital-Wandler Eingänge
- 1 I2C-Bus (Inter IC Bus)

- 1 USART (Universal Synchronous/Asynchronous Receiver Transmitter)
- 1 SPI-Schnittstelle (Serial Peripherals Interface)
- 1 JTAG-Schnittstelle
- 1 Watch-Dog-Timer
- 1 Analogkomparator

Der ATmega16 wird in verschiedenen Gehäuseformen hergestellt. Um den leichten Austausch eines defekten Mikrocontrollers zu ermöglichen, wird im „Praktikum Mechtronische Systeme“ das relativ große PDIP40 (Plastic Dual Inline Package) in Verbindung mit einem Sockel eingesetzt. Durch die großen Pins und deren Abstände zueinander kann ein Signalverlauf direkt am IC durch einen Tastkopf abgegriffen werden.

Der auf dem Board verbaute ATmega16 hat 40 Anschlüsse, von denen 32 programmierbare Ein- bzw. Ausgänge (User I/Os) sind. Diese sind in Ports mit jeweils 8 Pins gruppiert (PortA-D) und somit innerhalb des Mikrocontrollers auf ein Register abgebildet. Die Pins sind als digitale Ein- und Ausgänge vom Programmierer nutzbar. Zusätzlich bieten einige IOs noch Sonderfunktionen, die je nach Bedarf genutzt werden können. Tabelle 2 gibt einen Überblick der alternativen Pinbelegungen.

Tabelle 2: Sonderfunktionen ATmega16.

Port	**Pin**	**Funktion**
PORTA	PA0-PA7 (ADC0)-(ADC7)	Eingänge für 10-bit Analog-Digital-Wandler
PORTB	PB0 (T0)	Eingang für Zähler 0
	PB1 (T1)	Eingang für Zähler 1
	PB2 (AIN0)	positiver Eingang des Analogkomparators
	PB3 (AIN1)	negativer Eingang des Analogkomparators
	PB4-PB7 PB4 (SS) PB5 (MOSI) PB6 (MISO) PB7 (SCK)	SPI (Serial Peripherals Interface) Slave Select Master Out Slave In Master In Slave Out Serial Clock (Taktgeber

Port	**Pin**	**Funktion**
PORTC	PC0-PC1 PC0 (SCL) PC1 (SDA)	I2C (Inter IC Bus - Serieller Zweidrahtbus) Serial Clock (Taktgeber) Serial Data
	PC2-PC7	JTAG-Schnittstelle
PORTD	PD0-PD1 PD0 (RXD) PD1 (TXD)	USART (RS232-Schnittstelle) serieller Eingang serieller Ausgang
	PD2 (INT0)	Eingang für externen Interrupt 0
	PD3 (INT1)	Eingang für externen Interrupt 1
	PD4 (OC1B)	PWM-Ausgang
	PD5 (OC1A)	PWM-Ausgang
	PD6-PD7	

Neben den programmierbaren Ein- bzw. Ausgängen hat der Mikrocontroller noch weitere Anschlüsse, die zum Betrieb notwendig sind:

- RESET - Neustart des Programms
- VCC - Spannungsversorgung
- AVCC - Spannungsversorgung analoge Schaltungsteile
- GND - Masse, Bezugspotential
- AREF - Referenzspannung AD-Wandler
- XTAL1, XTAL2 - Anschlüsse für einen Quarz (mit 16 MHz bestückt)

Bild 5 zeigt das Gehäuse des im Praktikum eingesetzten Mikrocontrollers mit der entsprechenden Zuordnung zwischen Pins und deren Funktion.

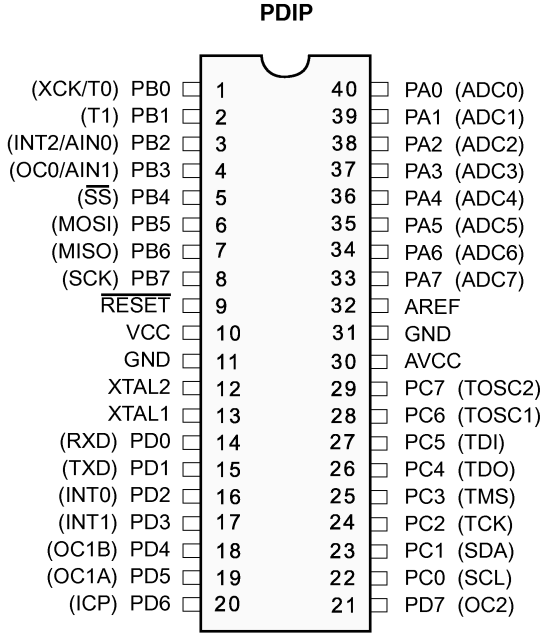

Bild 5: Pinbelegung ATmega16.

2.2.3 Schnittstellen

Zur Implementierung eines mechatronischen Systems sind neben dem Controller und dessen Programmierung noch weitere Komponenten, wie zum Beispiel Sensoren und Aktoren oder Kommunikationsschnittstellen notwendig. Da sich diese je nach Aufgabenstellung stark unterscheiden können, wurden einzelne häufig benötigte Module entwickelt, die über die im Folgenden dargestellten Schnittstellen mit der Steuerplatine verbunden werden können. Die einzelnen Module werden in Abschnitt 2.3 genauer beschrieben. Bild 6 zeigt einen Überblick der Schnittstellen und die Nummerierung der Pins.

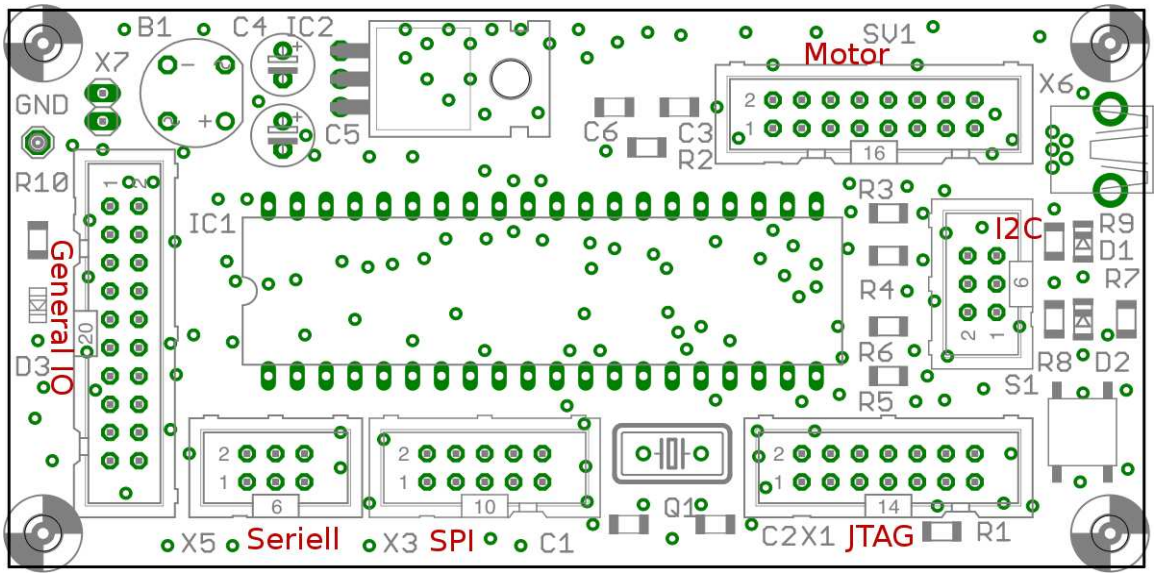

Bild 6: Schnittstellen der Steuerplatine.

Allgemeine I/O Schnittstelle

Der I/O Stecker X2 dient zur Anbindung eigener Schaltungen an die Steuerplatine. Über diese Schnittstelle können Sensoren, Aktoren, Ein-, Ausgabegeräte aber auch komplexe Schaltungen auf anderen Platinen angesteuert und ausgewertet werden. Zu diesem Zweck wurden möglichst viele allgemein verwendbare Pins des ATmega16 zur Verfügung gestellt. Der Steckverbinder stellt die Spannungsversorgung der Steuerplatine (VCC und GND) zur Verfügung. Wie bereits dargestellt, muss die gesamte Verlustleistung des Spannungsreglers auf der Steuerplatine berücksichtigt werden. Für empfindliche Schaltungen könnte sich der Widerstand der Flachbandkabel negativ auswirken. Die Versorgung sollte über einen Kondensator in der externen Schaltung stabilisiert werden. Für analoge Messungen sind alle acht Pins (PORTA) des Analog-Digital-Wandlers herausgeführt. Eine Interrupt gesteuerte Programmierung wird über die Pins INT0 (PB2) und INT1 (PD3) an der I/O Schnittstelle ermöglicht.

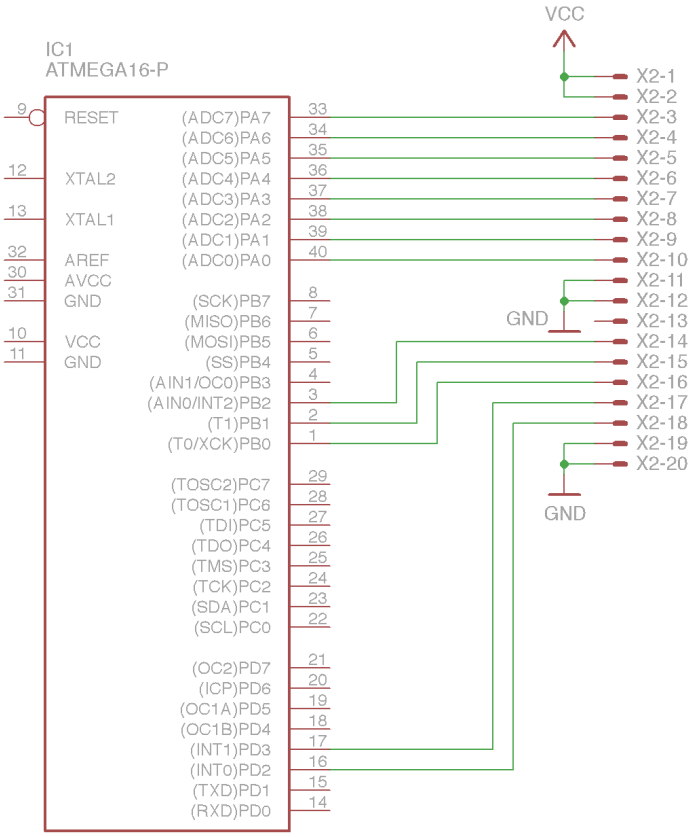

Bild 7: Schaltplan der I/O-Schnittstelle.

Serielle Schnittstelle

Die serielle Schnittstelle stellt den Sende- (TXD) und Empfangskanal (RXD) der integrierten USART-Schnittstelle (PD0 und PD1) des ATmega16 und eine Spannungsversorgung zur Verfügung (siehe Bild 8). Um mit einem RS232 kompatiblen Gerät (z.B. PC COM-Schnittstelle) kommunizieren zu können, ist ein Pegelwandler nötig, der die logischen Pegel des Controllers auf die geforderten +/- 15 V wandelt. Das Modul mit dieser Funktionalität wird in Abschnitt 2.3.4 näher beschrieben.

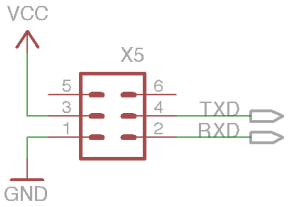

Bild 8: Serielle Schnittstelle.

I2C Schnittstelle

Bei der I2C-Schnittstelle [3] handelt es sich um einen seriellen Datenbus. Dieser Datenbus benötigt zwei Signalleitungen. Gesteuert wird die Kommunikation von einem sog. Master. Diesen Part übernimmt in der Regel der Mikrocontroller. Der Master gibt den Bustakt auf der Taktleitung SCL vor. Die Daten werden über die Datenleitung SDA synchron zum Bustakt nach einem festgelegten Protokoll übertragen. An einem Bus können bis zu 127 verschiedene ICs angeschlossen werden. Um Konflikte zu vermeiden, werden sogenannte Open-Kollektor Treiber eingesetzt, die Pull-Up Widerstände an den Signalleitungen erfordern. Diese Pull-Up Widerstände sind auf der Steuerplatine

und dürfen auf den Erweiterungsmodulen nicht mehr verbaut werden. Für die externen Schaltungen ist an der Schnittstelle - für kleine Leistungen - die geregelte und ungeregelte Versorgungsspannung (VCC, VCC2) der Steuerplatine angeschlossen. Bild 9 zeigt die Pinbelegung der I2C-Schnittstelle und die integrierten Pull-Up Widerstände.

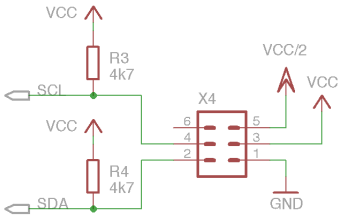

Bild 9: I2C Schnittstelle.

Folgende Auflistung stellt einige gebräuchliche I2C Komponenten vor:

- LM75 Digitaler Temperatur Sensor
- PCF8591 4-Kanal ADC + DAC
- PCF8574(A) 8-bit I/O-Erweiterung
- PCF8577C LCD direkt/duplex Treiber
- NM24C09 8k-Bit EEPROM mit Schreibschutz
- X24C16 16k-Bit EEPROM
- PCF8573 Uhr/Kalender

Motor Schnittstelle

Bild 10 zeigt die Verdrahtung der Motor Schnittstelle, an der sechs IO-Pins des Mikrocontrollers angeschlossen sind. Zwei dieser IO-Pins können mit ihrer Sonderfunktion als PWM-Ausgänge betrieben werden. Mit dieser Funktion kann zum Beispiel die Geschwindigkeit von Motoren gesteuert werden. Zur Versorgung der externen Motorcontroller und Motortreiber, sind an der Schnittstelle die geregelte Spannung VCC, die ungeregelte Spannung VCC2 und die Masse angeschlossen. Allerdings ist bei der Versorgung von externer Hardware durch die Steuerungsplatine die max. Verlustleistung am Spannungsregler (IC2) und der max. Strom durch den Brückengleichrichter (B1) zu berücksichtigen.

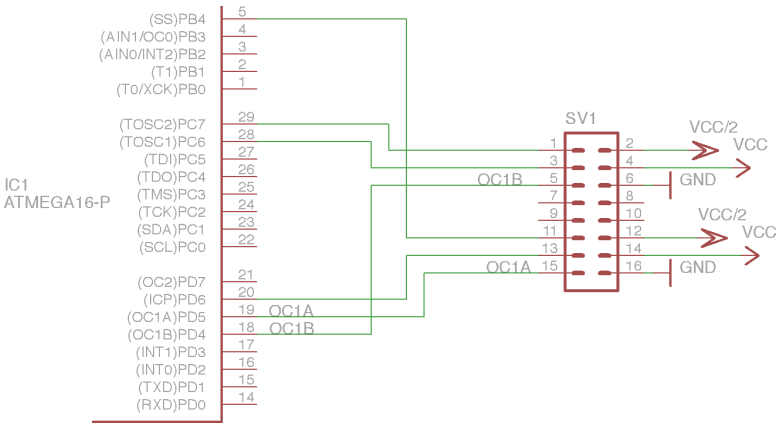

Bild 10: Motor Schnittstelle.

JTAG Schnittstelle

Die JTAG-Schnittstelle (**J**oint **T**est **A**ction **G**roup) ist zum Testen und Debuggen von Hardware vorgesehen. Bild 11 zeigt die Anschlüsse der Stiftleiste. Über diese Schnittstelle kann auf interne Register zugegriffen werden, der Mikrocontroller kann kurz angehalten und die internen Register ausgelesen oder der Programmablauf beobachtet werden. So kann die Funktion des entwickelten Programmcodes verifiziert oder Fehler gefunden werden. Über die Port- oder AD-Register kann auch die Funktionalität von externen Schaltungen überprüft werden. Durch den Zugriff auf den internen Speicher besteht die Möglichkeit den Programmspeicher des Mikrocontrollers zu beschreiben. Hierfür wird allerdings ein JTAG-Programmer benötigt, der im Rahmen des Praktikums zur Verfügung gestellt wird.

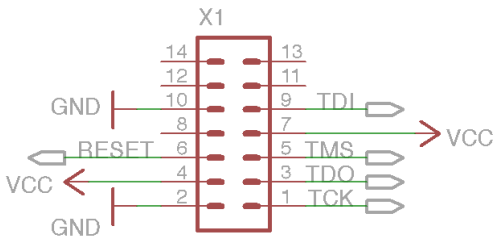

Bild 11: JTAG Schnittstelle.

SPI Schnittstelle

Der Mikrocontroller kann ebenfalls mit Hilfe des Serial Peripheral Interface (SPI) programmiert werden. Der hierfür benötigte Adapter wird im Rahmen des Praktikums nicht bereit gestellt. Bild 12 zeigt die Anschlussbelegung der Schnittstelle. Durch das Reset Signal kann der Mikrocontroller zurückgesetzt werden. SPI ist ein synchrones Datenübertragungsverfahren. Der Takt wird über SCK vom Master zum Slave übertragen. Über MISO (Master-In Slave-Out) werden Daten vom Master zum Slave übertragen, während MOSI (Master-Out Slave-In) die Daten in die andere Richtung führt.

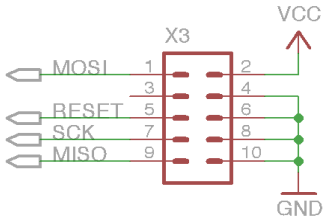

Bild 12: SPI Schnittstelle.

2.3 Erweiterungsmodule

Zur Erweiterung der Funktionalität der Steuerplatine werden noch weitere Module zur Verfügung gestellt, die über die im letzten Kapitel vorgestellten Schnittstellen mit der Steuerplatine verbunden werden können. Dabei handelt es sich um Module, die das Programmieren des Mikrocontrollers auf der Steuerplatine erlauben, und Hardwareinterfaces zu Displays, Motoren etc.

2.3.1 Motor Endstufe

In der Motor Endstufe – Schaltplan siehe Bild 13 - wird ein integrierter Motorleistungstreiber L298 [4] eingesetzt. Dieser Leistungstreiber ist mit zwei Vollbrücken ausgestattet. Mit diesen Vollbrücken können zwei Gleichstrommotoren mit jeweils 2 A angesteuert werden. Hierbei ist aber auf eine ausreichende Kühlung zu achten. Die beiden Vollbrücken lassen sich über Lötbrücken parallel schalten. Dann kann durch die Motor Endstufe ein Gleichstrommotor mit bis zu 4 A angesteuert werden. Auch in diesem Fall muss für eine ausreichende Kühlung des Motorleistungstreibers gesorgt

werden. In diesem Fall können die nicht benutzten Pins des Motorsteckers benutzt werden um eine zweite Motor Endstufe anzusteuern.

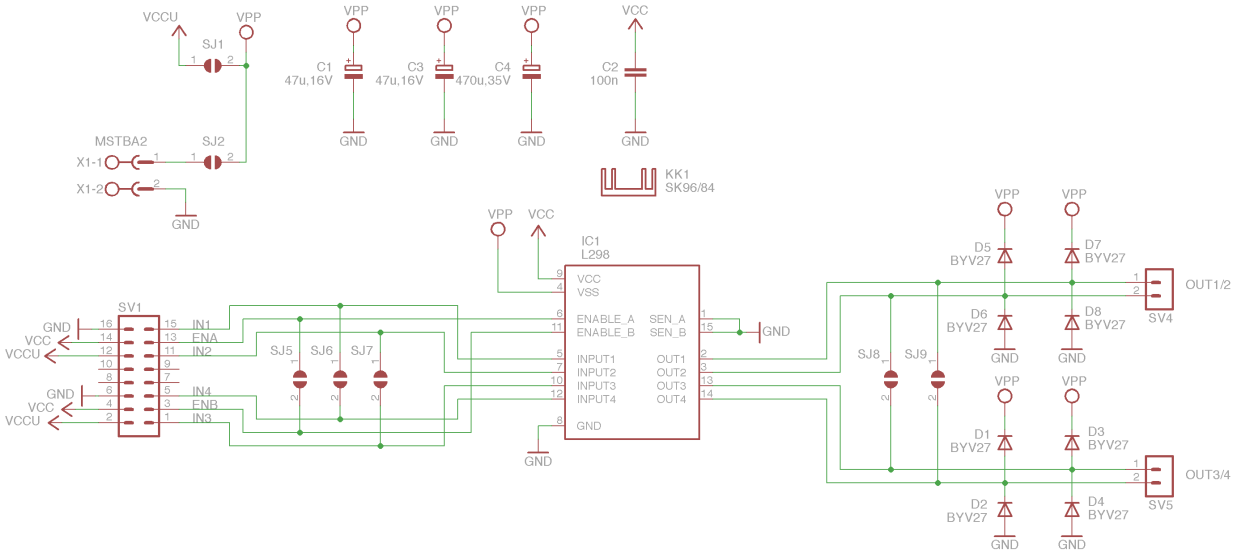

Bild 13: Motor Endstufe.

Hinweis: Bei der Ansteuerung größerer Leistungen ist die Verwendung einer externen Spannungsversorgung am Steckverbinder X1 erforderlich. Die Lötbrücke SJ2 ist hierbei zu schließen. Die Lötbrücke SJ1 muss offen sein. Außerdem ist auf eine ausreichende Kühlung des Motortreibers L298 zu achten.

2.3.2 I2C/LCD Modul

Das I2C/LCD Modul ist mit einem Display ausgestattet, der alphanumerische Zeichen darstellen kann. Bild 14 zeigt den Schaltplan des Moduls. Wenn der Mikrocontroller als I2C Master betrieben wird, können an der I2C Schnittstelle bis zu 127 I2C Slaves (I2C Bausteine) betrieben werden. Um Pins des Mikrocontrollers für andere Funktionen frei zu halten wurde ein LC Display [5] über den I2C Bus angeschlossen. Um das LCD anzusteuern, wurde der I2C Baustein PCF8574T [6] von Philips verwendet, der acht digitale Ein- und Ausgänge bereitstellt. Mit den Lötbrücken A0 bis A2 in Bild 14 wird ein Teil der I2C Adresse des Moduls festgelegt. Die vollständige Adresse ist „0100XXXY“ wobei X durch A2-A0 zu ersetzen ist (Details siehe [PCF8574]) und Y das Schreib-/Lese-Bit darstellt. Die Ansteuerung des I2C/LCD Moduls ist zeitaufwendig und sollte deshalb innerhalb zeitkritischer Softwareteile nicht verwendet werden.

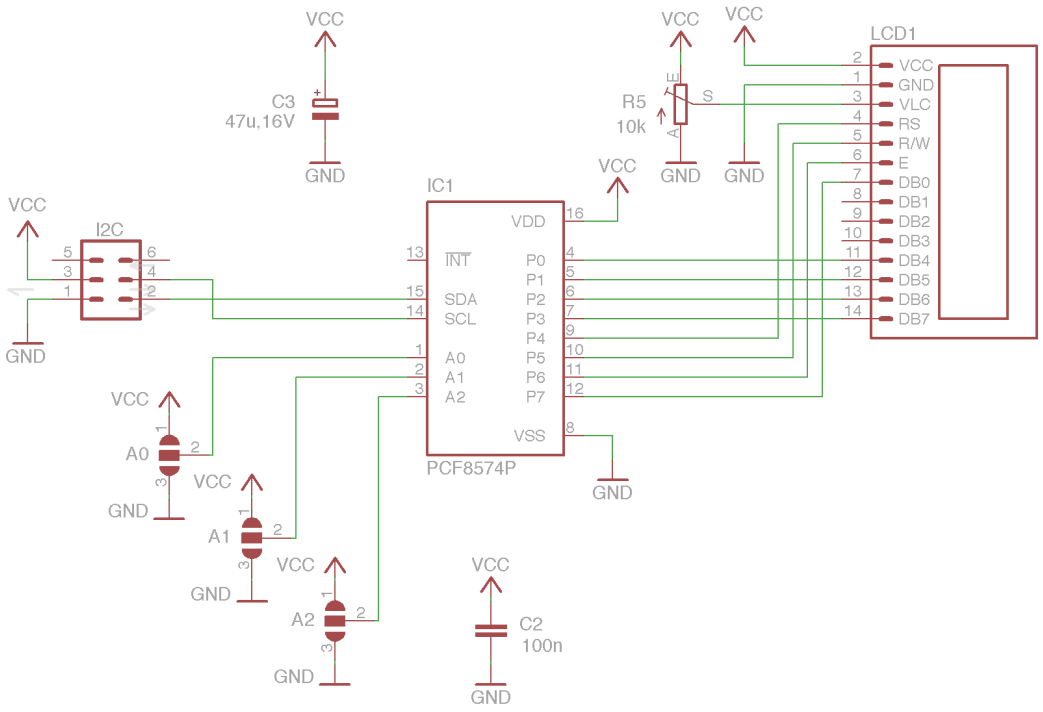

Bild 14: I2C/LCD Modul.

2.3.3 JTAG Modul

Das JTAG-Modul wird an der 14 poligen JTAG Schnittstelle des Mikrocontrollers angeschlossen. Am SUBD-Buchsenstecker des JTAG-Moduls wird, über einen USB auf Seriell Adapter (z.B. Reichelt Artikel- Nr.: DELOCK 61460), die Verbindung zum PC hergestellt. Über diese Verbindung kann mit dem AVR-Studio von Atmel (siehe Softwarebeschreibung) der Mikrocontroller konfiguriert, der Programmspeicher des Mikrocontrollers beschrieben und das Testen und Debuggen der Software erfolgen. Auf das JTAG-Modul wurde ein Bootloader geladen, mit dem über das AVR-Studio manuell die Atmel-Firmware geladen werden kann. Der Bootloader ist unter folgendem Link erhältlich:

http://www.siwawi.arubi.uni-kl.de/avr_projects/evertool/index.html

Hinweis: In der Version 20100122 ist auf der Platine die Beschriftung des Widerstandes R1 und C1 vertauscht.

2.3.4 RS232 Modul

Das RS-232 Modul realisiert eine PC-kompatible serielle Schnittstelle. Die schon angesprochene Pegelwandlung wird von einem MAX232 IC [7] durchgeführt. Bild 15 zeigt den zugehörigen Schaltplan. Mit diesem Modul ist es möglich über ein Terminal Programm wie z.B. Hyperterminal für Windows oder Minicom für Linux, mit dem Mikrocontroller zu kommunizieren. Ist eine Weiterverarbeitung oder Analyse der Daten notwendig, so kann die serielle Schnittstelle auch in selbst entwickelten Programmen genutzt werden.

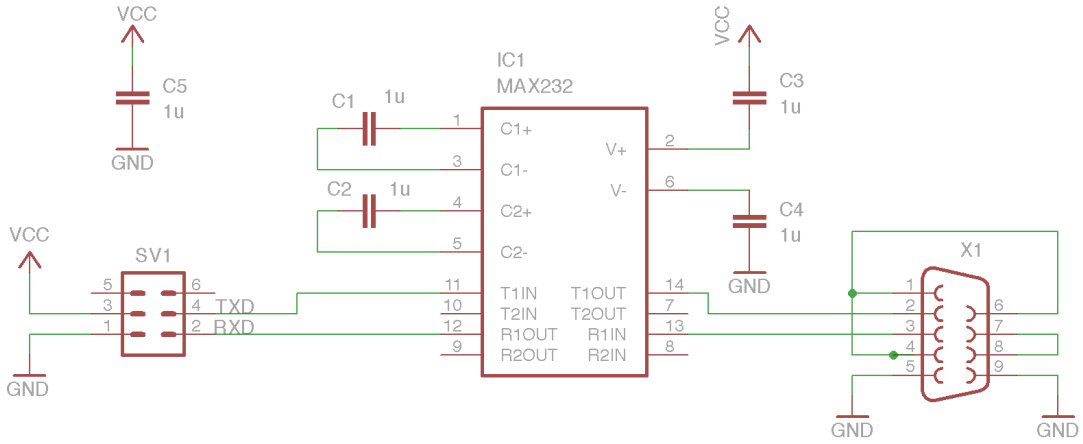

Bild 15: RS232 Modul.

2.4 Software

Zur Entwicklung von Software für die vorgestellte Hardware sind verschiedene Entwicklungswerkzeuge notwendig. Im Folgenden wird auf das AVR-Studio [8] eingegangen, die von Atmel bereitgestellte Integrierte Entwicklungs-Umgebung (IDE), sowie die zusätzlich benötigten Tools.

2.4.1 Softwarewerkzeuge

Um aus C-Code eine auf dem ATmega16 lauffähige Binärdatei zu generieren, ist eine Compiler Suite notwendig. Der C-Compiler generiert aus dem C-Code Objekt Dateien. Der Linker bindet diese mit den (Standard-)Bibliotheken. Durch einen Assembler wird der Maschinen-Code, also die Funktionalität in Maschinenbefehlen, erzeugt und eine binäre Datei erzeugt, die im Speicher des Mikrocontrollers geschrieben und ausgeführt werden kann. Um diese Binärdatei in den Mikrocontroller zu laden wird Software benötigt, die mit dem Programmiergerät kommuniziert.

Die im letzten Absatz vorgestellten Tools stellen den minimalen Satz an Entwicklungswerkzeugen für die C-basierte Softwareentwicklung dar. Um eine effiziente Entwicklung und den Test der erstellten Algorithmen zu entwickeln, ist noch ein Simulator sinnvoll, der den Mikrocontroller in Software simuliert und den Progammablauf dadurch für den Entwickler nachvollziehbar macht. Oftmals erweist sich die Simulation als unzureichendes Mittel, wenn externe Hardware und deren zeitliches Verhalten getestet werden muss. In diesem Fall ist ein so genannter Debugger in Verbindung mit einem Emulator sinnvoll, der auf die echte Hardware zugreift. Er ermöglicht das Anhalten des Controllers auf Anfrage des Benutzers oder an vordefinierten Stellen im Code. Im angehaltenen Zustand können dann die einzelnen Register ausgelesen werden und somit auch indirekt auf externe Hardware zugegriffen werden.

WinAVR ist ein Windows Packet und enthält neben dem GNU C-Compiler AVR-GCC für Atmel AVR Mikrocontroller weitere Zusatzprogramme und die C-Standardbibliothek AVR-LIBC. Das AVR-Studio ist eine Entwicklungsumgebung von Atmel, die eine Projektverwaltung, einen Assembler, einen Simulator und einen Debugger umfasst und die Verwendung verschiedener Programmieradapter und Emulatoren ermöglicht.

Das AVR-Studio unterstützt die Verwendung der WinAVR Werkzeuge. In dieser Kombination können mit dem AVR-Studio C-Programme erstellt, compiliert und auf den Mikrocontroller übertragen werden. Der Debugger kann mit dem integrierten Simulator oder einem Emulator genutzt werden. Zur Emulation und Programmierung kann das in Abschnitt 2.3.3 beschriebene JTAG Modul verwendet werden.

2.4.2 Installation

Vor der Installation müssen die Installationspakete von den unten angegebenen Links geladen werden. Die Programmierbeispiele und die beschriebenen Hardwaremodule sind mit den angegebenen Versionen getestet. Bei Problemen wird empfohlen immer erst die Entwicklungswerkzeuge mit der angegebenen Versionsnummer zu verwenden. Von den Installationspaketen sollte das Paket WinAVR zuerst installiert werden, damit nach der späteren Installation des AVR-Studios die Programmpakete von WinAVR richtig in das AVR-Studio eingebunden werden. Die Installation wird durch einen Installationsassistenten unterstützt.

Hinweis: Die Installation von WinAVR sollte vor der Installation des AVR-Studios erfolgen.

WinAVR Version 20100110:

http://sourceforge.net/projects/winavr/files/

AVR-Studio Version 4.18:

http://www.atmel.com/dyn/products/tools_card_v2.asp?tool_id=2725

oder alternativ ohne Registrierung

http://www.mikrocontroller.net/articles/AVR-Studio

2.4.3 AVR-Studio

In diesem Kapitel wird ein Einstieg in den Umgang mit dem AVR-Studio von Atmel gegeben und notwendige Einstellungen aufgeführt. Neben dieser Einführung sind im Internet zahlreiche Tutorials zum Thema ATMega16 und der hier verwendeten Software zu finden [9][10][11].

Neues Projekt

Ein neues Projekt wird mit dem Dialog „Project->New“ erstellt. Als erste Option wird der Compiler „AVR-GCC“ ausgewählt. Weiterhin muss noch der Name des Projekts und das Verzeichnis angegeben werden, in dem die Daten gespeichert werden (siehe Bild 16).

Bild 16: Projekt Dialog AVR-Studio.

Im nächsten Fenster wird der Emulator oder Programmer und der Mikrocontroller ausgewählt. Folgende Einstellungen sind für die beschriebene Hardware zu wählen:

„Debug platform“: JTAG ICE

„Device“: Atmega16

LZS Bibliothek

Zur Programmierung des Mikrocontrollers sind für die beschriebene Hardware Software-Funktionen vorbereitet. Um diese zu nutzen, müssen die Header-Dateien und die Bibliothek in das Projekt eingebunden werden. Die Header-Dateien und die Bibliothek befinden sich in der Datei „avr-lrs.zip“. Nachdem die Datei entpackt ist, können die entsprechenden Dateien mit „Project → Configuration Options“ eingebunden werden. Im linken Bereich des Fensters wird mit „Include Directories“ der Pfad für die Header Dateien und mit „Libraries“ der Pfad für die Bibliothek festgelegt (Bild 17/18).

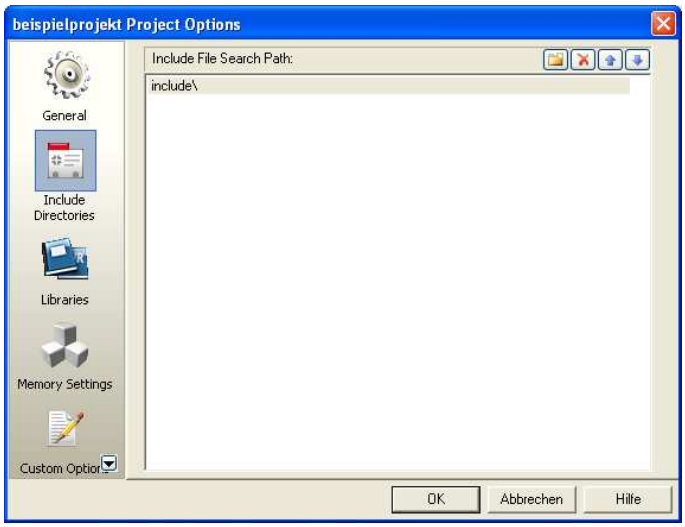

Bild 17: Pfad der Header-Dateien.

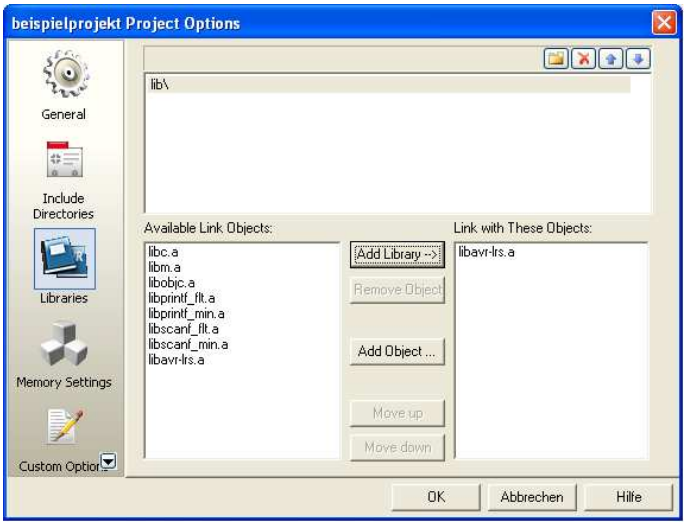

Bild 18: Pfad der LZS Bibliothek.

2.4.4 Programmierung

Mit den Programmen WinAVR und AVR-Studio steht nach der Installation eine umfangreiche Entwicklungsumgebung für die Programmiersprache „C“ zur Verfügung. Neben Programmabläufen die interne Resourcen des Mikrocontrollers verwenden, können auch die Ein- und Ausgänge des Mikrocontrollers und deren Sonderfunktionen programmiert und ausgewertet werden. Diese Funktionalität wird in der Programmierung durch das Lesen und Beschreiben von Registern (spezielle Speicherbereiche) erreicht. Die Beschreibung der Register mit der zugehörigen Funktion kann der Dokumentation des Mikrocontrollers entnommen werden. Die Register des Mikrocontrollers können unabhängig vom Programm durch externe Ereignisse verändert werden.

Für das Setzen und Löschen einzelner Bits in einem Register werden logische Bitoperation in „Standard C“ verwendet. Im folgenden Beispiel wird nur der Zustand des angegebenen Bits mit der angegebenen Bitnummer verändert. Die anderen Bits bleiben unverändert.

```
Register   |=   (1 << Bitnummer);     //Bitnummer in Register gesetzt
Register   &=   ~(1 << Bitnummer);    //Bitnummer in Register gelöscht
```

Beispiel:

```
0000 0000  |=   (1 << 3);            // Ergebnis: 0000 0100
1111 1111  &=   ~(1 << 6);           // Ergebnis: 1101 1111
```

Ports und Pinkonfiguration

Die Kommunikation mit externen Komponenten erfolgt über die Pins des Mikrocontrollers. Diese können zum Beispiel als digitale Ein- oder Ausgänge konfiguriert werden. Der ATmega16 hat vier Ports mit insgesamt 32 Ein- oder Ausgangspins. Die Ports werden mit A, B, C und D (Index „x“) bezeichnet. Die Pins sind von „0“ bis „7“ (Index „n“) nummeriert. Mit dem Port und der Pinnummer ist ein Pin am Mikrocontroller eindeutig referenziert. Dies ist in Bild 5 am Beispiel des Atmega16 ersichtlich.

Um die Funktion eines Ports zu konfigurieren stehen drei Register pro Port zur Verfügung:

DDRx	Datenrichtungsregister
PORTx	Datenregister
PINx	Eingangsregister

Im DDRx wird festgelegt, ob der Pin als Ein- bzw. Ausgang verwendet wird. Wenn der Pin als Ausgang programmiert ist, kann durch PORTx der Ausgang gesetzt oder gelöscht werden. Als Eingang kann am Pin durch das Setzen des zugehörigen Bits im PORTx ein „Pull-up“ Widerstand aktiviert werden, durch das Löschen dieses Bits kann der Eingang „hochohmig“ geschaltet werden. Die Abfrage der Zustände an einem Pin erfolgt über PINx.

Hinweis: Die Abfrage der Eingänge muss über die Eingangsregister „PINx“ erfolgen, da bei den Registern „DDRx“ und „PORTx“ nur der programmierte Zustand nicht aber der logische Wert an den Eingangspins des Mikrocontrollers abgefragt wird!

Tabelle 3 zeigt den C-Code zur Konfiguration der gewünschten Funktion der I/O-Pins.

Tabelle 3: Konfiguration I/O Pins.

Register DDRx	**Register PORTx**	**Funktion**
`DDRx &= ~(1<< Pxn);`	`PORTx &= ~(1<<Pxn);`	Pin n des Ports x als Eingang - hochohmig (Grundeinstellung)
`DDRx &= ~(1<< Pxn);`	`PORTx |= (1<< Pxn);`	Pin n des Ports x als Eingang mit Pull-up
`DDRx |= (1<< Pxn);`	`PORTx &= ~(1<<Pxn);`	Pin n des Ports x als Ausgang Low-Pegel treiben
`DDRx |= (1<< Pxn);`	`PORTx |= (1<< Pxn);`	Pin n des Ports x als Ausgang High-Pegel treiben

x = A..D, n = 0..7

Beispiel:

```
include <avr/io.h>

      DDRB &= ~(1<< PB1);     // Pin PB1 als Eingang
      PORTB |= (1<< PB1);     // Pull-up aktivieren

      if ( PINB & (1<<PB1) )  // führe Aktion aus, wenn Bit PB1 gesetzt ist
      {
            // Aktion
      }
```

Die Bezeichnung der Ports und Pins sind abhängig vom eingesetzten Mikrocontroller. Für den ATmega16 muss die Header Datei „<avr/io.h>“, die deren Definition enthält, eingebunden werden.

Bibliothek „AVR-LIBC“

Das WinAVR Packet stellt für Atmel 8-Bit RISC-Prozessoren die Bibliothek „AVR-LIBC“ bereit. Die Bibliothek und deren Beschreibung kann unter folgender Adresse bezogen werden:

http://www.nongnu.org/avr-libc/

In dieser C-Bibliothek werden die Funktionen des „ANSI-C“-Standards und Teile des Nachfolgers (C99) bereitgestellt. Diese können durch das Hinzufügen der entsprechenden Header-Datei im Programm verwendet werden.

Zum Beispiel können durch das Einbinden der Header-Datei „<util/delay.h>“, folgende Verzögerungs-funktionen verwendet werden:

```
void _delay_ms(double ms)     // Max. Auflösung bis 262.14 ms / F_CPU in Mhz
                              // Max. Verzögerung 6.5535 s
void _delay_us(double us)     // Max. Verzögerung  768 us / F_CPU in Mhz
```

Neben dem Einbinden der Header Datei ist auch die Definition des Taktes für den Mikrocontroller nötig. Diese Informationen sind in der Beschreibung der Bibliothek zu finden und im folgenden Beispiel aufgezeigt. Der Takt kann auch in den Projekteinstellungen „Project → Configuration Options“ unter „General“ angegeben werden.

```
#include <util/delay.h>

#define F_CPU 16000000UL  // 16 Mhz
// #define F_CPU 14.7456E6

      _delay_ms(10);
```

Für das angegebene Beispiel ist mit der Bibliothek nur eine maximale Verzögerung von 16,38 ms (262,14 ms/16) mit hoher Auflösung möglich. Für längere Verzögerungen ist nur noch eine reduzierte Auflösung von 1/10 ms und eine maximale Verzögerung von 6,5535 s möglich.

Programmbeispiel

In diesem Abschnitt wird eine blinkende LED als Einstiegsprogramm eingeführt.

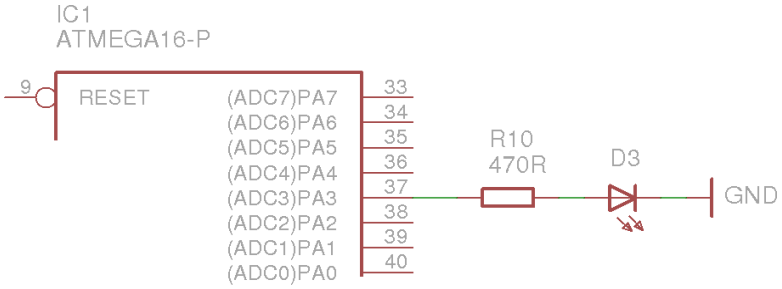

Bild 19: LED Ansteuerung ATmega16.

Bild 19 zeigt die Schaltung und die Verbindung zum Mikrocontroller. Über einen Widerstand und eine LED ist der Pin „PA3“ mit der Masse verbunden. Durch den Vorwiderstand wird ein definierter Strom durch die LED eingestellt. Ein lauffähiges Programmbeispiel wird im Folgenden gezeigt:

```
#include <avr/io.h>
#include <util/delay.h>

void initLED(void)
{
      DDRA        |=    (1<<PA3);         // Pin PA3 als Ausgang
      PORTA       |=    (1<<PA3);   // Pin PA3 auf "1"
```

```
}

void ledOn(void)
{
      PORTA |=    (1<<PA3);/  / Pin PA3 auf "1"
}

void ledOff(void)
{
      PORTA &=~(1<<PA3);      // Pin PA3 auf "0"
}

int main(void)
{
      initLED();
      while(1)
            {
                  delay_ms(500);
                  ledOn();
                  delay_ms(1000);
                  ledOff();
            }
      return 0;
}
```

Um die Definition der Register verwenden zu können, wird die Header-Datei „avr/io.h“ eingebunden. Die Funktion „initLED()“ konfiguriert den Pin PA3 (Port A) des Mikrocontrollers als Ausgang und setzt diesen auf High. Eine Initialisierung der im Programm verwendeten Pins sollte immer erfolgen, um definierte Anfangszustände im Zusammenwirken mit externer Hardware zu erhalten.

Die Funktionen „ledOn()“ und „ledOff()“ setzen den Portpin PA3 auf High bzw. Low. Das Hauptprogramm beginnt mit dem Funktionsaufruf „initLED()“, bevor eine Endlosschleife durchlaufen wird. Durch Verzögerungsfunktionen zwischen Setzen und Rücksetzen des Pins wird der Takt eingestellt. Im obigen Beispiel leuchtet die LED ca. eine Sekunde und erlischt für ca. eine halbe Sekunde.

3 Aktorik

3.1 Gleichstrommotor

Für einfache und gleichförmige rotatorische Bewegungen können permanent erregte Gleichstrommotoren mit einer Nennspannung von etwa 12 V verwendet werden. Der maximale Strom des Motors ist der Haltestrom I_H. Der Haltestrom stellt sich ein, wenn die Welle angehalten wird, während der Motor mit Nennspannung gespeist wird. Als Schutzmaßnahme für den Treiber sollte der Haltestrom des Motors einen kleineren Wert als der maximale kontinuierliche Strom des Treibers aufweisen.

Geeignet sind z.B. Motoren mit einer Mindestleistung von 4 W aus dem Modellbaubereich, die durch hohe Stückzahlen in der Fertigung die Kostenbelastung niedrig halten.

3.2 Servomotor

Servomotoren eignen sich besonders für exakte Positioniervorgänge. Zusätzlich zum Gleichstrommotor kann ein Servomotor einen Winkelgeber mit einer Ansteuerelektronik und einem Getriebe in kompakter Bauweise beinhalten. Die Position der Motorwelle kann durch ein pulsweitenmoduliertes Signal vorgegeben werden, so dass sich dieser Motortyp wegen der einfachen Ansteuerung und Präzision für verschiedene Aufgaben im Bereich der Robotik eignet.

3.3 Schrittmotor

Gegenüber einem Gleichstrommotor bietet der sog. Schrittmotor durch sein Wirkprinzip entscheidende Vorteile bei Positionieraufgaben.

Ein Schrittmotor ist ein Motor, dessen Rotor bei Verwendung einer entsprechenden Ansteuerschaltung bei jedem elektrischen Steuerimpuls um einen bestimmten Winkel bzw. Schritt rotiert. Bei Positionieraufgaben kann deshalb durch Zählen der einzelnen Impulse bzw. Schritte auf einen gesonderten Drehgeber oder Sensor verzichtet werden. Zu beachten ist das Haltemoment des Schrittmotors, das dem maximalen Drehmoment entspricht. Ist das Lastmoment höher als das Haltemoment, kann die Welle ihre Position nicht mehr beibehalten bzw. der vorgegebenen Bewegung durch die Ansteuerung nicht mehr folgen.

Es gibt mehrere Typen von Schrittmotoren, die sich im Wesentlichen durch die Anzahl der verbauten Spulenpaare unterscheiden und die unterschiedlich angesteuert werden können. Viele Schrittmotoren verfügen über 4, 6 oder 8 Anschlussleitungen.

Bild 20 zeigt eine vereinfachte Darstellung eines Motors mit zwei Spulen bzw. zwei Polpaaren. Der Motor kann unipolar angesteuert werden, indem die mittlere Leitung - wie im Bild gezeigt - auf Masse gelegt wird und die anderen Anschlüsse von einer Spannungsversorgung zu- oder abgeschaltet werden. Der Motor kann so angesteuert werden, dass an jeder Spule entweder eine positive oder negative Spannung liegt, ohne Verwendung des mittleren Anschlusses (bipolare Ansteuerung).

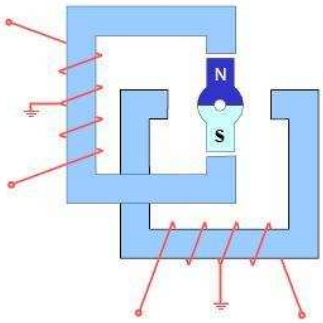

Bild 20: Vereinfachte Darstellung eines Schrittmotors mit Mittelanzapfung.

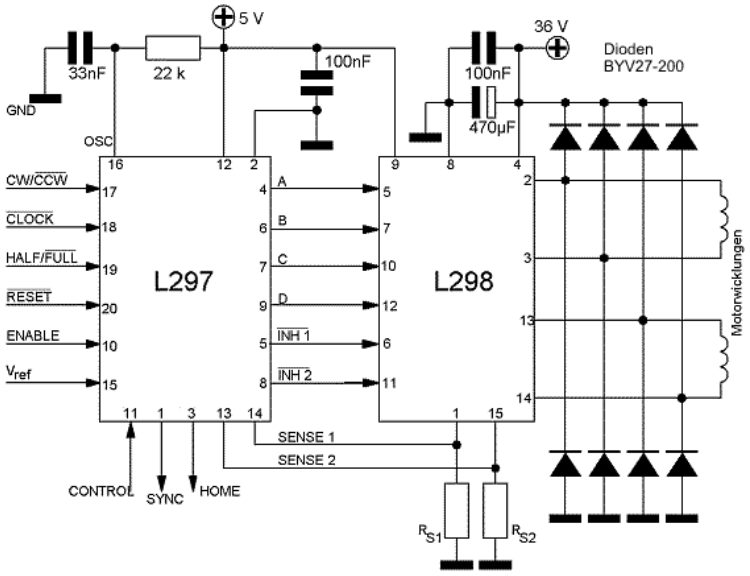

Bild 21: Beispiel einer Ansteuerschaltung für einen Schrittmotor aus dem Datenblatt des Bauteils L298.

Tabelle 4: Einige Signale der Ansteuerschaltung aus Bild 21.

CW/CCW	Gibt die Richtung an, in die der Motor bewegt werden soll.
Clock	Durch einen kurzen Impuls auf diese Leitung wird der Motor einen Schritt bewegt. In einer Schleife kann man nur diese Leitung kurz ein- und ausschalten.
Half/Full	Gewöhnlich legt man den Anschluss auf Masse. Wird er mit 5 V belegt, dann werden immer halbe Schritte durchgeführt, also doppelt so viele Schritte pro Umdrehung (Halbschrittbetrieb).
Vref	Hier muss eine Referenzspannung u_{ref} zwischen 0 und 3 V angelegt werden. Die Spannung lässt sich mit $R_S := R_{S1} = R_{S2}$ (vgl. Bild 21) zu $u_{ref} = I_{Motorstrom} \cdot R_S \cdot 1{,}41$ berechnen.

Ausführliche Informationen finden sich im Herrsteller-Datenblatt des L297 bzw. L298.

4 Sensorik

Grundsätzlich können die Sensoren in zwei Gruppen unterteilt werden: Digitale und analoge Sensoren.

Analogsensoren werden an die Analogeingänge des Kontroller-Boards angeschlossen und liefern bei 12-Bit-A/D-Wandlern Werte zwischen 0 und $2^{12} = 4096$.

Digitale Sensoren liefern nur den Wert 0 (Low) oder 1 (High). Sie werden über sogenannte Pull-up-Widerstände an die Digitaleingänge angeschlossen. Ist der digitale Schalter geschlossen, liegt das Signal auf Masse (Low). Wenn der Schalter geöffnet ist, liegt das Signal auf +5 V (High).

4.1 Optoelektronische Sensoren

Zu den einfachsten Sensoren, die sich an einen Mikrokontroller anschließen lassen, gehören Fotowiderstände, Fototransistoren oder Fotodioden. Das ausgegebene Signal kann direkt interpretiert werden. Videokameras benötigen hingegen sehr spezielle Schaltungen, damit das Videosignal vom Mikrokontroller verwertet werden kann.

4.1.1 Fotowiderstände

- Halbleiterbauelemente, deren Widerstand sich abhängig von der auftretenden Strahlung aufgrund des inneren Fotoeffekts ändert.
- Sind stark temperaturabhängig.
- Eignen sich nur für Messaufgaben mit sehr geringen Anforderungen, z.B. zur Detektion von Schwarz/Weiß-Unterschieden.

4.1.2 Fototransistoren und Fotodioden

- Nutzen den inneren Fotoeffekt in der Sperrschicht, d.h. der Generation von Ladungsträgern in der Raumladungszone.
- Durch die Stromverstärkung des Transistors sind Fototransistoren deutlich empfindlicher (bezogen auf die Fläche) als Fotodioden. Allerdings weisen Photodioden meist eine sehr viel größere empfindliche Fläche auf.
- Das Empfindlichkeitsmaximum bei Siliziumbauelementen liegt im infraroten Bereich.
- Schnellere Antwortzeit als Photowiderstände.

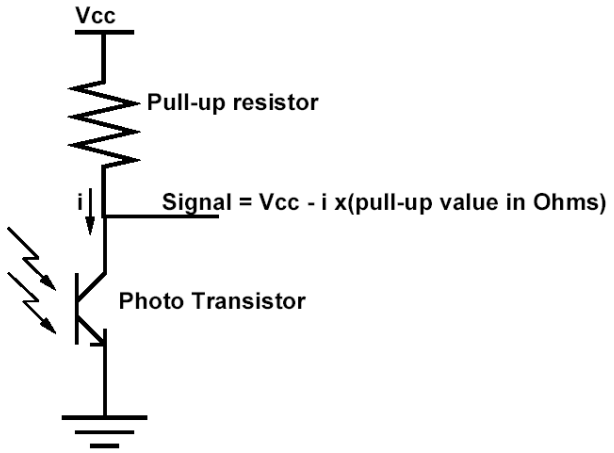

Bild 22: Mögliche Beschaltung eines Phototransistors zur Auswertung des Signals mit einem Mikrocontroller.

Selbst wenn die Wellenlänge der verwendeten IR-Empfangdiode auf die der IR-Sendediode abgestimmt ist, kann Umgebungslicht das Messsignal beeinflussen. Es bestehen grundsätzlich zwei Möglichkeiten den störenden Einfluss des Umgebungslichtes zu minimieren.

1. Tageslichtfilter vor den Empfänger.
2. Modulation des Sendelichts mit Frequenz f_m . Dem Empfänger wird ein Bandpassfilter nachgeschaltet, der genau diese Frequenz f_m herausfiltert.

Beispiel zur Infrarot-Abstandsmessung

- Kombiniert IR-Sender und Empfänger.
- Eignet sich nur für kurze Distanzen.
- Sensorsignal hängt von den Reflektionseigenschaften der Oberflächen der detektierten Objekte ab.

Dieser Sensortyp hat normalerweise einen sehr viel kleineren Messbereich als ein Ultraschall-Entfernungsmesser.

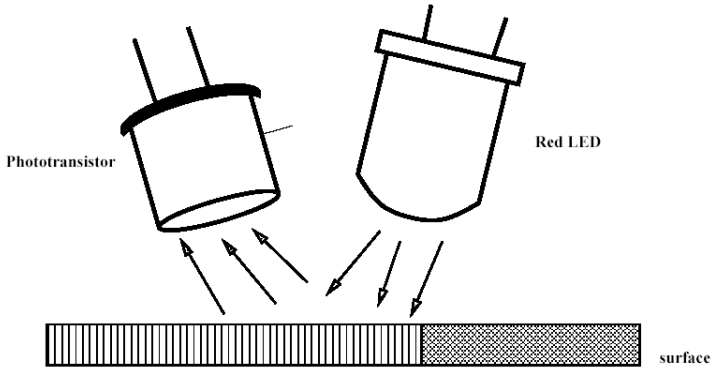

Bild 23: Prinzipielle Anordnung eines IR-Abstandsmessers. Der Abstand ist eine Funktion des vom Hindernis reflektierten und vom Phototransistor aufgenommenen Lichts.

Einige Möglichkeiten zur Umsetzung einer IR-Abstandsmessung

Bei der Firma „Conrad“ sind beispielsweise fertige IR-Abstandsmesser von Sharp der Serie GP2D12 erhältlich, die eine der Entfernung proportionale Ausgangsspannung liefern. Diese Sensoren gibt es in mehreren Ausführungen, sie haben eine Reichweite von 10 bis 80 cm bzw. 4 bis 30 cm und strahlen sehr flach/gebündelt ab.

Preiswerter ist der IR-Detektorbaustein IS471-F von Sharp und einer IR-Sendediode. Der Baustein IS471-F besteht aus einer Wechsellicht-Empfängerschaltung mit integrierter Elektronik und einem Treiber für Infrarotdioden. Die Schaltung ist für Reflexlicht- oder Lichtschrankenanwendungen geeignet. Die Reichweite im Reflexlichtbetrieb beträgt z.B. mit der IR-Sendediode LD274 ca. 10 cm (weißes Papier in Postkartengröße). Mit einer einfachen Zusatzschaltung kann man die Reichweite bis auf über 25 cm im Reflexbetrieb steigern. Hierzu wird ein externer Transistor ZTX750 angeschaltet. Der Ausgang 4 der Schaltung IS471-F liefert negativ gerichtete Taktsignale. Diese Signale steuern den Transistor ZTX750. Dieser wiederum schaltet für jeden Ausgangimpuls die LED LD274 ein. Hierbei wird der LED-Strom durch den Widerstand mit einem Wert von 4,7 Ω auf ca. 600 mA begrenzt. Die nachfolgende Schaltung (Bild 24) ist für 5 V Betriebsspannung ausgelegt.

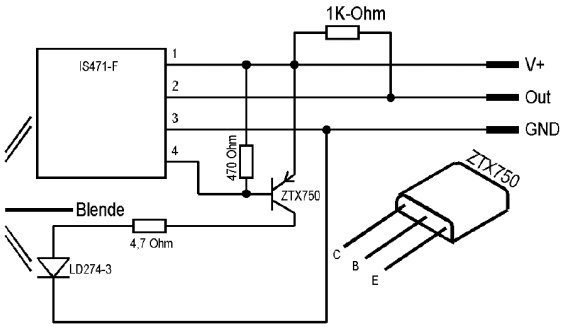

Bild 24: Zusatzschaltung für den IR-Detektorbaustein IS471-F von Sharp.

Beispiel zur Schwarz-Weiß-Detektion mit Infrarot

Als „Augen“ der Schaltung werden hier eine IR-Sendediode und die beiden Fototransistoren, die zu einem kompletten und einem halben IR-Reflexkoppler CNY70 (Hersteller „Temic“) gehören, benutzt.

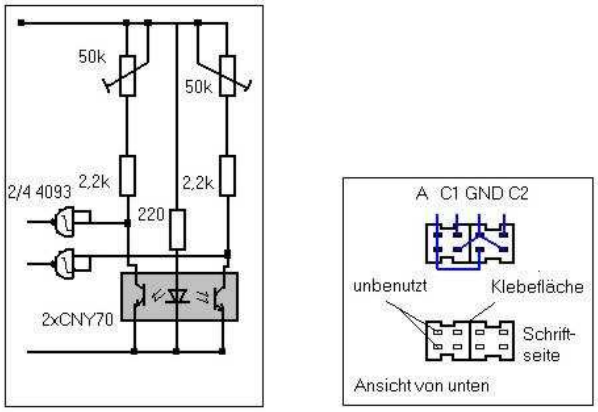

Bild 25: Schaltplan (links) und Verdrahtung der Sensoren CNY70 (rechts).

Mit diesem Abtastsystem kann eine schwarze Linie (ab 5 mm Stärke) auf hellem Grund detektiert werden. Die Kollektoren der Fototransistoren liegen an den Eingängen eines 4-fach-NAND-Bausteins mit Schmitt-Trigger (Typ 4093), damit ein eindeutiger Pegel an den beiden Ausgängen erzielt werden kann. Mit den beiden Potentiometern kann das Schaltverhalten der Koppler den Gegebenheiten des Untergrunds anpasst werden (anfangs in Mittelstellung bringen).

Das Abtastsystem wird so zusammengeklebt, dass eine Sendediode mittig zwischen zwei Empfängern sitzt (siehe Verdrahtungsschema): Trifft das IR-Licht der Diode z.B. auf den schwarzen Strich wird es nicht reflektiert, beide Empfängertransistoren sperren. Weicht das Abtastsystem von der Linie ab, erhält einer der beiden Empfänger von der hellen Fläche reflektiertes Licht, somit sinkt das Kollektorpotential des Fototransistors und bei Erreichen der Triggerschwelle des angeschlossenen Inverters kippt ein Ausgang der Schaltung.

Beispiel zum Rad-Encoder

Ein Rad-Encoder („Drehgeber“) ist ein Sensor, der die Position oder Umdrehungszahl einer Welle misst. Normalerweise wird er auf der Antriebswelle eines Antriebsmotors oder einer Achse befestigt. Er besteht aus einer Gabellichtschranke und einer Lochscheibe bzw. einer Reflexlichtschranke und einer Scheibe mit abwechselnd schwarzem und weißem „Kuchenmuster“. Der Sensor liefert das

Signal entweder in Form eines Codes, der einer bestimmten Ausrichtung der Welle entspricht, oder als Folge von Impulsen.

Eine relativ teure und sehr genaue Version ist bei der Firma „Conrad“ als Kombination aus Lichtschranke Sharp GP1A71R und Taktscheibe 120 erhältlich.

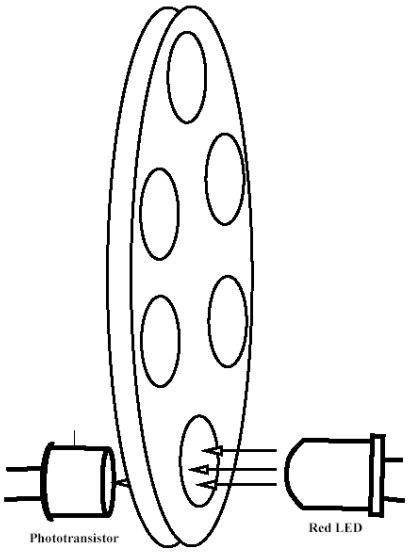

Bild 26: Beispiel eines Rad-Encoders mit Lochscheibe und Gabellichtschranke.

4.2 Taktile Sensoren

Beispiel zu Berührungssensoren

Am einfachsten wird ein Berührungssensor mit einem Mikroschalter realisiert. In der Praxis haben sich diese Art Sensoren bestens bewährt, da sie geringste Störgeräusche und einfachst zu interpretierende Signale liefern. Je nach Anwendungsfall muss ein Prellen des Schalters unterbunden werden.

4.3 Akustische Sensoren

Der wichtigste Vertreter dieser Gruppe ist der Ultraschall-Messwertgeber.

Beispiel zur Ultraschall-Abstandsmessung

- Besteht aus Ultraschallsende- und Empfangsstufe sowie einer Steuerelektronik, um die Laufzeit des Ultraschallsignals zu bestimmen.
- Laufzeit des Ultraschallsendesignals ist proportional zum Abstand.

5 Halbleiterbauelemente

5.1 Dioden

Einige Einsatzmöglichkeiten von Dioden:

1. Verpolschutz für integrierte Schaltkreise.
2. Schutz von Ein- und Ausgängen integrierter Schaltkreise vor Überspannung.
3. Zenerdiode zur Spannungsstabilisierung bzw. Spannungsbegrenzung.
4. Diode als Freilaufpfad beim Schalten induktiver Lasten (z.B. Elektromotoren, Relais).
5. Dioden als Gleichrichter für Wechselspannungen.

5.2 Bipolartransistoren

Einige Einsatzmöglichkeiten von Bipolartransistoren:

1. Der Bipolartransistor als Schalter kleiner Lasten (z.B. Relaisansteuerung, LED-Ansteuerung).
2. Der Bipolartransistor als Verstärker bei der Auswertung von Sensorsignalen.

5.3 Feldeffekttransistoren

Alternativ zu mechanischen Relais können auch Feldeffekttransistoren (MOSFETs) zum Schalten von Lasten verwendet werden. Dabei bieten MOSFETs den Vorteil der Ansteuerung mit einem einem pulsweitenmodulierten Signal, mit dem zum Beispiel die Drehzahl eines Gleichstrommotors gesteuert werden kann. Zur Reduktion der notwendigen Bauelemente für die Ansteuerung der MOSFETs sind n-Kanal-MOSFETs als Lowside-Schalter zu bevorzugen. Einen weiteren Vorteil stellt die kompaktere Bauform von MOSFETs im Vergleich zu mechanischen Relais dar. Allerdings müssen bei MOSFETs Schalt- und Leitungsverluste berücksichtigt werden, weshalb die Entwärmung im Betrieb sichergestellt werden muss.

A Anhang

Tabelle 5: Ausgewählte Bezugsquellen für elektronische und mechanische Bauelemente.

Hersteller / Bezug	Artikel / Bereich	Internetadresse	Katalogart
Mechanik- und Elektronikwerkstatt der Technischen Fakultät	Mechanik, Elektronik, Elektrik	www.mew.uni-erlangen.de	www
CONRAD	Modellbau, Elektronik, Elektrik	www.conrad.de	Pap., www
ELV	Modellbau, Elektronik, Elektrik	www.elv.de	Pap., www
REICHELT	Modellbau, Elektronik, Elektrik	www.reichelt.de	Pap., www
RS COMPONENTS	Elektronik, Elektrik	www.rs-componets.de	Pap., www
FARNELL	Elektronik, Elektrik	www.farnell.com	Pap., www

Tabelle 6: Ausgewählte Zusammenstellung zu Robotikforen im Internet.

Forum	Internetadresse
ROBOTERNETZ	http://www.roboternetz.de
ELEKTRONIK-PROJEKT	http://www.elektronik-projekt.de
ELEKTRONIK-KOMPENDIUM	http://www.elektronik-kompendium.de
LÖTSTELLE	http://www.loetstelle.net

A.1 Leiterkabel

Bei Leiterkabeln muss sowohl der Widerstand, der zu einem Spannungsabfall über dem Kabel führt, beachtet werden, als auch die maximale Stromdichte, die durch die Überhitzung des Kabels begrenzt wird. Der Widerstand eines Kabels kann durch den spezifischen Widerstand des Materials (ρ), die Querschnittfläche des Leiters (A) und dessen Länge (l) berechnet werden. Für Leiter mit einer runden Querschnittfläche kann folgende Formel gebildet werden:

$$R = \frac{\rho}{A} \cdot l = \frac{\rho}{\pi \cdot r^2} \quad \text{mit } \rho_{Cu} = 0{,}0172 \frac{\Omega \cdot \text{mm}^2}{\text{m}}$$

Die maximale Stromdichte unterscheidet sich je nach Anwendungsgebiet beziehungsweise Wärmeabtransport. Bei Transformatoren ist der Abtransport der Verlustleistung in den inneren Lagen behindert. In diesem Fall kann eine maximale Stromdicht von ca. 2,5 A/mm² angesetzt werden. Bei PVC isolierten Kupferleitungen (umgeben von Luft) gilt eine maximale Stromdichte von 10 A/mm² bis zu einem Durchmesser von maximal 4 mm². Ein Flachbandkabel mit einem Leiterquerschnitt von ca. 0,1 mm² wäre demnach mit maximal 1 A belastbar. Wie bei allen Grenzwerten gilt auch hier, dass man sie nicht ausreizen muss. Tabelle 7 zeigt einige typische Werte von Leitern aus Kupfer. Der gezeigte Widerstand gilt für eine Temperatur von 20 °C. Bei höheren Temperaturen (unter Belastung)

steigt der Widerstand. Der maximale Strom beziehungsweise die maximale Stromdichte sinken bei höheren Umgebungstemperaturen.

Tabelle 7: Widerstand und Strombelastbarkeit von Leitern.

Durchmesser	*Querschnitt*	*Widerstand*	*max. Strom bei*	
[mm]	[mm²]	[Ω/m]	2,5 A/mm²	10 A/mm²
0,1	0,008	2,19	0,02	0,079
0,25	0,049	0,35	0,123	0,491
0,35	0,096	0,179	0,241	0,962
0,5	0,196	0,088	0,491	1,963
0,75	0,442	0,039	1,104	4,418
1	0,785	0,022	1,963	7,854
1,5	1,767	0,01	4,418	17,671
2	3,142	0,005	7,854	31,416
2,25	3,976	0,004	9,94	39,761

Der Leitungswiderstand wirkt sich besonders negativ aus, wenn Leistungselektronik wie zum Beispiel Motortreiber und Logik oder Sensoren gemeinsam durch ein Kabel mit geringem Querschnitt versorgt werden. Nimmt man beispielsweise ein Flachbandkabel mit einem Querschnitt von 0,1 mm² und versorgt Leistungselektronik (1 A) und Logik/Sensorik, so schwankt der Spannungsabfall über der Leitung und damit die Versorgungsspannung der Logik/Sensorik um bis zu 0,36 V. Dies kann zu Störungen führen.

A.2 Entwärmung von Halbleiterbauelementen

Für Halbleiterbauelemente gibt es Grenzwerte für die maximal zulässige Betriebstemperatur. Diese sind je nach Produkttyp unterschiedlich und werden in den entsprechenden Datenblättern spezifiziert. Um sicherzustellen, dass ein Bauteil nicht außerhalb der Spezifikation betrieben wird, muss dessen Verlustleistung an die Umgebung abgeführt werden. Durch die Wärmeleitfähigkeit vom Silizium zur Umgebung wird der Temperaturanstieg im Silizium bestimmt.

Zur Berechnung der Temperatur in der Schaltung (Sperrschichttemperatur T_J) wird demnach die Verlustleistung P_D, die Umgebungstemperatur T_A und der Wärmewiderstand von der Sperrschicht zur Umgebung R_{th} benötigt. T_A wird meist aus Sicherheitsgründen auf 45 °C gesetzt. Die Verlustleistung wird durch den betrachteten IC bestimmt. Bei einem Spannungsregler kann die Verlustleistung zum Beispiel aus der Differenz von Eingangs und Ausgangsspannung und der Strombelastung (unter Vernachlässigung der Verlustleistung, die in der Kontrolllogik entsteht) berechnet werden:

$$P_D = \left(U_{inp} - U_{out}\right) \cdot I$$

Für den Wärmewiderstand R gilt:

$$R = \frac{\Delta T}{P} \quad \left[\frac{\mathrm{K}}{\mathrm{W}}\right]$$

In dieser Formel lässt sich die Analogie zu elektrischen Schaltungen erkennen, wenn man die Temperaturdifferenz durch eine Spannungsdifferenz ersetzt und die Leistung durch Strom, so berechnet obige Formel den elektrischen Widerstand. Durch eine einfache Umformung lässt sich die Sperrschichttemperatur T_J berechnen:

$$R_{th} = \frac{T_j - T_A}{P} \rightarrow T_j = T_A + R_{th} \cdot P$$

Um die Sperrschichttemperatur T_J berechnen zu können, wird der Wärmewiderstand benötigt. Im einfachsten Fall kann eine Rechnung ohne Kühlkörper durchgeführt werden. In der Regel wird in diesem Fall der Wärmewiderstand von der Schaltung zur Umgebung im Datenblatt des Bausteins angegeben (R_{thJA}).

Sollte ein Kühlkörper eingesetzt werden, so bildet sich der Gesamtwiderstand von der Sperrschicht zur Umgebung aus einer Serienschaltung von drei Wärmewiderständen. Eine Addition der einzelnen Widerstände ergibt den Gesamtwiderstand. Der erste erstreckt sich von der Sperrschicht zum Gehäuse. Dieser ist dem Datenblatt des ICs zu entnehmen (R_{thJC}). Der zweite ist der Wärmewiderstand, den der Übergang vom Gehäuse zum Kühlkörper darstellt. In Tabelle **Error! Reference source not found.** sind typische Werte für verschiedene Kontaktflächen, im Falle der Behandlung der Oberflächen mit und ohne Wärmeleitpaste. Die Werte gelten nur für planare Flächen. Die angegebenen Größen sind in der Einheit cm²·K/W angegeben und müssen noch durch die Kontaktfläche dividiert werden.

Tabelle 8: Spezifischer Wärme-Kontaktwiderstand.

Kontaktfläche	*Ohne Leitpaste*	*Mit Leitpaste*
Metall auf Metall	1,0	0,5
Metall auf Eloxalschicht	2,0	1,4

Der letzte Wärmewiderstand ist der des Kühlkörpers. Falls ein dedizierter Kühlkörper verwendet wird, ist dessen Wert den Unterlagen zu entnehmen. Als Kühlkörper (für geringe Verlustleistungen) lassen sich auch andere Materialien, wie Leiterplatten oder Metallbleche einsetzen.

Der Wärmeaustausch zwischen der Luft an der Kühlkörperoberfläche und der (weit) entfernten Umgebungsluft wurde hier nicht berücksichtigt. Der nicht immer zu vernachlässigende Anteil an Verlustleistung, der durch die Bauteilanschlüsse (Pins) abgeführt wird wurde ebenfalls nicht berücksichtigt. Es wurde nur das statische Verhalten betrachtet. Um das dynamische Verhalten berechnen zu können, muss die Wärmekapazität berücksichtigt werden.

B Literaturverzeichnis

[1] Atmel, 8-bit AVR Microcontroller with 16 kB In-System-Programmable Flash ATMega16(L), 2009, http://www.atmel.com/atmel/acrobat/doc2466.pdf

[2] STMicroelectronics, L78xx Positive voltage regulators, 2010, http://eu.st.com/stonline/books/-pdf/docs/2143.pdfA

[3] Philips Semiconductor, THE I2C-BUS SPECIFICATION VERSION 2.1, 2000, http://www.nxp.com/acrobat_download2/literature/9398/39340011.pdf

[4] STMicroelectronics, DUAL FULL-BRIDGE DRIVER L298, 2000, http://www.st.com/stonline/-books/pdf/docs/1773.pdf

[5] Displaytech Ltd., LCD MODULE 161A SERIES, 2010, http://www.reichelt.de/?;ACTION-=7;LA=28;OPEN=0;INDEX=0;FILENAME=A500%252FLCD161A.pdf;SID=15ttRBtqwQAQ8AAGoYiUY8738bd9e1979e2a5c79cf5d328db0d55

[6] Philips Semiconductor, PCF8574 Remote 8-bit I/O expander for I2C-bus, 2002, http://www.nxp.com/documents/data_sheet/PCF8574.pdf

[7] Maxim Integrated Products, +5V-Powered, Multichannel RS-232 Drivers/Receivers, 2006, http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf

[8] Atmel, Atmel Products - Tools and Software, 2010, http://www.atmel.com/dyn/Products/-tools_card.asp?tool_id=2725

[9] Mikrocontroller.net, AVR-GCC Tutorial, 2010, http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial

[10] Purdue University, AVR Simulation with the ATMEL AVR Studio 4, 2010, http://www.tech.purdue.edu/ecet/courses/ecet309/Reference_Materials/-Simulation_AVR_Studio_4.pdf

[11] Robomodules, AVR Studio 4 - Tutorial, 2010, http://www.robomodules.de/portal/index.php?-id=199&type=1

Anmerkung 130412 von WLS:

Bei meinem Textvergleich zum Handbuch 2011 stellte ich nur die Änderung auf Seite 32/3. Zeile fest.

Statt “Fehler! Verweisquelle konnte nicht gefunden werden”, jetzt “Fehler: Referenz nicht gefunden”.

Betreff: Re: Handbuch zum Praktikum - Beschreibung der LZS-Boards

Datum: Fri, 12 Apr 2013 12:28:49 +0200

Von: Wolfgang Sörgel <soergel@lzs.eei.uni-erlangen.de>

An: Dirnecker Tobias <Tobias.Dirnecker@leb.eei.uni-erlangen.de>

Hallo Herr Dirnecker,

meines Erachtens sind die Angaben und Beschreibungen im Handbuch 2013 bis auf eine Textstelle alle korrekt.

Die fehlerhafte Angabe ist auf Seite 10, letzte Zeile zu finden: statt "INT0 (PB2)" muss es "INT0 (PD2)" heißen.

--
Feundlichst grüßt
i.A. Wolfgang Sörgel