Entwicklung eines Testgerätes zur Überprüfung von Mikrocontroller-Platinen

Matthias Rock

Bachelorarbeit

Entwicklung eines Testgerätes zur Überprüfung von Mikrocontroller-Platinen

Matthias Rock

09. September 2015

Lehrstuhl für Datenverarbeitung
Technische Universität München

Matthias Rock. *Entwicklung eines Testgerätes zur Überprüfung von Mikrocontroller-Platinen.* Bachelorarbeit, Technische Universität München, München, 2015.

Betreut von Prof. Dr.-Ing. Klaus Diepold und Dipl.-Ing Johannes Feldmaier; eingereicht am 09. September 2015 bei der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München.

Lehrstuhl für Datenverarbeitung, Technische Universität München, 80290 München, `http://www.ldv.ei.tum.de`.

Zusammenfassung

Diese Arbeit beschreibt die Entwicklung und den Aufbau sowie die Funktionsweise eines Gerätes zum Testen von Mikrocontroller-Boards. Ziel dieser Arbeit ist es, ein Testgerät zu entwickeln und aufzubauen, mit dem die wichtigsten Funktionen des Mikrocontroller-Boards „Arduino Leonardo“ überprüft werden können. Hierzu zählen insbesondere die digitalen Ein- und Ausgänge, die analogen Eingänge sowie die SPI- und I^2C-Schnittstelle. Dies ist notwendig für die Sicherstellung der vollständigen Funktionsfähigkeit dieser Platinen in studentischen Projekten bei Lehrveranstaltungen und Workshops am Lehrstuhl für Datenverarbeitung (LDV) an der Technischen Universität München. Entstanden ist ein Universal-Testgerät, das neben den genannten Anforderungen weitere Zusatzfunktionen unterstützt und auf eine Vielzahl anderer Mikrocontroller-Platinen erweiterbar ist. Zu diesem Zwecke werden theoretische Grundlagen von Testgeräten eruiert, Hardwarekomponenten konstruiert und geeignete Testverfahren entwickelt. Diese werden anschließend in der Software implementiert.

Abstract

This thesis describes the development, the construction and the operation of a device for testing microcontroller boards. The aim of this thesis is to develop and build a device which can test the most important functions of the microcontroller board „Arduino Leonardo“. That includes in particular the digital inputs and outputs, the analog inputs and the SPI and I^2C interface. These functions are necessary to ensure the full functionality of these boards in student projects, in courses, and workshops at the Chair for Data Processing at the Technical University of Munich. The result is an universal board tester, which not only fulfills the requirements mentioned, but also provides additional functions and it can be extended to a variety of other microcontroller boards. For this purpose theoretical principles are investigated, hardware components designed, and appropriate test procedures are developed. These are subsequently implemented in the software.

Inhaltsverzeichnis

1 Einleitung

Zu Beginn dieser Arbeit wird das Thema sowie dessen Zielsetzung beschrieben und im Weiteren werden der Aufbau und die technischen Spezifikationen von verwendeten Modulen und Datenbussen bzw. Kommunikationsschnittstellen erläutert.

1.1 Motivation und Zielsetzung

Mikroprozessoren gibt es seit Ende der 1960er Jahre und sie bilden heutzutage einen grundlegenden Kern in vielen technischen Geräten. Ein Mikrocontroller vereinigt neben einem Prozessorkern alle wesentlichen Funktionsgruppen, die für die Ausführung von Programmen auf einem einzigen Chip notwendig sind. Mit Mikrocontrollern lassen sich viele Probleme, für die früher komplexe Hardwareschaltungen notwendig waren, mit geringem Hardwareaufwand softwaretechnisch lösen [1].

1996 wurde von Atmel[1] die Mikrocontroller-Familie AVR entwickelt [2], die seitdem aufgrund ihres schlichten Aufbaus und ihrer einfachen Programmierbarkeit große Beliebtheit erfährt.

Darauf basierend [2] entwickelten Massimo Banzi und David Cuartielles im Jahre 2005 am Interaction Design Institute Iverea in Italien das Open-Source-Projekt Arduino[2], welches ursprünglich für die Steuerung von interaktiven Objekten und Installationen in Kunstausstellungen und Museen gedacht war. Hierbei spielt die Interaktion mit der Umwelt eine bedeutende Rolle. Zwischen der analogen Umwelt und dem digital arbeitenden Mikrocontroller findet die Kommunikation durch Sensoren und Aktoren statt, was auch als Physical Computing bezeichnet wird. Arduino kombiniert Hardware mit Software und ermöglicht heutzutage auch Laien einen einfachen und kostengünstigen Zugang zur Programmierung und Hardware-Entwicklung. Bei Arduino-Boards handelt es sich nicht um Ein-Platinen-Computer mit eigenem Betriebssystem, sondern es wird stets ein Mikrocontroller entsprechend der Aufgabenstellung programmiert. Mittlerweile existieren eine ganze Reihe unterschiedlicher Arduino-Boards und eine Vielzahl von Applikationen. Üblicherweise werden die Boards mit der Arduino-Entwicklungsumgebung, welche für Windows, Linux und Mac verfügbar ist, in einem vereinfachten C-Dialekt programmiert [3].

[1] http://www.atmel.com (zuletzt aufgerufen am 25.08.2015).
[2] http://www.arduino.cc (zuletzt aufgerufen am 25.08.2015).

1.1.1 Problemstellung

Im Rahmen von Lehrveranstaltungen und Workshops des Lehrstuhls für Datenverarbeitung (LDV) an der Technischen Universität München wird den Studentinnen und Studenten die Möglichkeit angeboten, ihr theoretisch erlerntes Wissen durch praktische Aufgabenstellungen zu festigen, indem elektronische Schaltungen mithilfe von Arduino-Boards des Typs „Leonardo“ erstellt werden müssen. Dazu werden die Mikrocontroller der Boards entsprechend der Aufgabenstellung programmiert und die Ein- und Ausgänge mit Hilfe weiterer elektronischer Bauelemente miteinander verschaltet. Hierbei kann es durchaus vorkommen, dass ein Student beispielsweise zwei Ausgänge des Mikrocontrollers miteinander verbindet, was bei unterschiedlichen Pegeln zur Zerstörung einzelner Pins oder des gesamten Mikrocontrollers führen kann. Außerdem kann es auch passieren, dass ein Produktionsfehler die vollständige Funktionsfähigkeit eines Arduino-Boards einschränkt, indem beispielsweise ein Pin nicht richtig verlötet worden ist oder mehrere Pins versehentlich durch den Lötprozess miteinander verbunden worden sind. Da zu erwarten ist, dass nicht jeder entstandene Defekt bemerkt bzw. vom Studenten gewissenhaft gemeldet oder behoben wird, besteht das Problem, dass ein nachfolgender Student, der dasselbe Board verwendet, unnötigerweise viel Zeit dafür aufwenden muss, um den bereits vorhandenen Defekt des Boards zu erkennen und zu beheben. Aus diesem Grund sollte der für die Betreuung zuständige Dozent die Möglichkeit haben, die Funktionsfähigkeit der Arduino-Boards mit möglichst wenig Aufwand testen zu können, damit sichergestellt wird, dass jeder Student ein voll funktionsfähiges Board zur Verfügung gestellt bekommt.

1.1.2 Aufgabenstellung

Zielsetzung dieser Arbeit ist die Entwicklung eines Testgerätes, das dem Anwender ermöglichen soll, die Funktionsfähigkeit von Arduino-Boards des Typs „Leonardo“ möglichst einfach und in angemessener Zeit überprüfen zu können. Zu prüfen sind insbesondere die digitalen Ein- und Ausgänge, die analogen Eingänge sowie die Funktionsfähigkeit der SPI- und I^2C-Schnittstelle. Nach erfolgtem Test muss dem Anwender angezeigt werden, ob das Board voll funktionsfähig ist oder nicht. Anschließend an die Überprüfung soll zudem ein Standard-Programm auf den Mikrocontroller aufgespielt und sichergestellt werden, dass der Bootloader voll funktionsfähig ist und alle Fuses richtig gesetzt sind. Des Weiteren ist ein Report des Testes zu erstellen, in dem die Ergebnisse der Überprüfung aufgelistet werden.

1.1.3 Umfang der Arbeit

Der Umfang dieser Arbeit entspricht im Wesentlichen den gerade beschriebenen Anforderungen gemäß der Aufgabenstellung, jedoch soll es zusätzlich möglich sein, durch entsprechende Adapter und Softwareanpassungen möglichst viele andere Arduino-Boards ebenfalls testen zu können, was bei der Hardware-Entwicklung somit berücksichtigt wer-

den muss. Diese Arbeit beschreibt in erster Linie die Entwicklung, den Aufbau sowie die Bedienung des Testgerätes.

1.2 Verwendete Module

Zum genauen Verständnis der Problemstellung sowie der Umsetzung dieser Arbeit ist es notwendig, den Aufbau und die Spezifikationen der verwendeten Module zu kennen. Aus diesem Grund werden nachfolgend sowohl das zu testende Board als auch zum Bau des Testgerätes verwendete Module beschrieben.

1.2.1 Arduino Leonardo-Board

Das zu testende Arduino Leonardo-Board (siehe Abbildung 5.1 im Anhang und Abbildung 1.1) ist ein Mikrocontroller-Board, das auf einem ATmega32u4 basiert. Es besitzt 20 digitale I/O-Pins, wovon sieben als PWM-Ausgang und 12 als Analog-Eingang fungieren können. Weiterhin enthält es einen 16 MHz Quarz, einen Mikro-USB-Anschluss, einen Spannungsversorgungsanschluss, eine SPI-Schnittstelle sowie einen Reset-Taster. Der verwendete AVR-Mikrocontroller ATmega32u4 besitzt einen eingebauten USB-Controller, der es dem Leonardo-Board ermöglicht, als Human Interface Device (HID) zu agieren. Damit ist es unter anderem möglich, von einem Computer als Maus oder Tastatur erkannt zu werden [4].

Abbildung 1.1: Mikrocontroller-Board *Arduino Leonardo*.

Spannungsversorgung

Die Spannungsversorgung kann über drei unterschiedliche Wege erfolgen. Zum einen über die Hohlbuchse oder den VIN-Pin der Buchsenleiste (7 - 12 VDC) und zum anderen über den Mikro-USB-Anschluss (5 VDC). Die beiden Pins *5 V* und *3.3 V* sind Ausgangspins und stellen geregelte Spannungen zur Verfügung. An IOREF kann ebenfalls die Betriebsspannung von 5 V abgegriffen werden [4].

Speicher

Der ATmega32u4 besitzt 32 KB Flash-Speicher, wovon 4 KB für den Bootloader genutzt werden, sowie 2,5 KB SRAM und 1 KB EEPROM [4].

I/O-Schnittstellen

Jeder der 20 digitalen I/O-Pins kann wahlweise als Eingang oder Ausgang geschaltet werden, wobei der High-Pegel 5 V beträgt. Sieben dieser Pins können alternativ als PWM-Ausgänge und 12 dieser Pins als analoge Eingänge mit jeweils 10 Bit Auflösung verwendet werden. Mit dem Pin AREF kann die analoge Referenzspannung verändert werden, die standardmäßig 5 V beträgt. Zwei der I/O-Pins können zum Senden (*TX*) und Empfangen (*RX*) von seriellen TTL-Daten und zwei weitere für die TWI/I^2C-Schnittstelle verwendet werden. Externe Interrupts sind mit fünf von den 20 I/O-Pins möglich. Die Zuordnung der Funktionen zu den einzelnen Pins kann aus Tabelle 5.1 im Anhang entnommen werden. Zu beachten sei noch, dass Pin *2* und *SDA* sowie Pin *3* und *SCL* jeweils miteinander verbunden sind. Der 6-polige Pfostenstecker stellt die SPI-Schnittstelle dar, dessen Pins unabhängig von den restlichen I/O-Pins der Steckerleiste sind. Die Zuordnung zwischen den Pins des ATmega32u4 und den Pins des Leonardo-Boards ist in Abbildung 5.2 im Anhang ersichtlich [4].

Programmierung

Üblicherweise wird das Board mit der Arduino-Entwicklungsumgebung[3] in einem vereinfachten C-Dialekt programmiert. Durch den Bootloader kann dies auch ohne Programmieradapter mittels einer USB-Verbindung erfolgen [4].

1.2.2 Betrachtung weiterer Arduino-Boards

Der Umfang der Aufgabenstellung umfasst lediglich die Entwicklung eines Gerätes speziell zum Testen von Arduino Leonardo-Boards. Da es jedoch naheliegend ist, dass in Zukunft auch andere Boards getestet werden müssen, ist es sinnvoll, das Testgerät so zu gestalten, dass durch entsprechende Adapter und Softwareanpassungen auch andere Boards

[3] http://www.arduino.cc/en/Main/Software (zuletzt aufgerufen am 25.08.2015).

grundsätzlich unterstützt werden, um somit eine komplette Neuentwicklung vermeiden zu können. Dies muss sich nicht zwangsweise auf Arduino-Boards beschränken, sondern kann auch auf andere Mikrocontroller-Platinen ausgeweitet werden.

Damit der oben genannte Gedanke bei der Entwicklung berücksichtigt werden kann, ist eine kurze Recherche der wichtigsten Kenndaten von zumindest einigen weiteren Mikrocontroller-Platinen unumgänglich. Da die Betrachtung aller vorhandenen Arduino-Boards[4] den Rahmen dieser Arbeit bei Weitem überschreiten würde und es zudem nicht zweckmäßig wäre, werden nachfolgend nur die aus Sicht des Autors relevantesten Boards betrachtet. Hierbei handelt es sich um die Boards Uno[5], Mini[6], Pro Mini[7], Micro[8] und Nano[9]. Die wichtigsten Daten dieser Boards sind übersichtlich in einer Tabelle zusammengetragen (siehe Tabelle 5.2 und 5.3 im Anhang).

1.2.3 Raspberry Pi 2

Der Raspberry Pi[10] ist ein sog. Ein-Platinen-Computer aus dem Jahr 2006. Für den Betrieb des in etwa Kreditkarten-großen Computers wird lediglich noch eine Speicherkarte sowie eine Spannungsversorgung benötigt. Als Betriebssystem werden üblicherweise angepasste Linux-Distributionen verwendet, womit selbst Büroanwendungen und Spiele auf dem Raspberry Pi lauffähig sind. Im Laufe der Zeit sind verschiedene Modelle entstanden, die sich in einigen Spezifikationen voneinander unterscheiden [3].

Seit Februar 2015 existiert der Raspberry Pi 2 Modell B (siehe Abbildung 1.2), der das ursprüngliche B+-Modell ablöst und bei der Entwicklung des Testgerätes in dieser Arbeit zum Einsatz kommen soll. Im Wesentlichen unterscheidet es sich vom B+-Modell des Vorgängers nur darin, dass es mehr Arbeitsspeicher sowie eine deutlich schnellere CPU besitzt. Am Raspberry Pi 2 sind folgende Schnittstellen vorhanden: 4x USB, Full HDMI, Ethernet, 3,5 mm Audio/Video-Anschluss, CSI (Camera interface), DSI (Display interface), MicroSD-Slot sowie 40 GPIO (General Purpose Input Output)-Pins. In Zukunft soll auf diesem Board selbst Microsoft Windows 10 lauffähig sein [5].

Spannungsversorgung

Die Spannungsversorgung erfolgt idealerweise über den Micro-USB-Anschluss (5 V), wobei zu beachten ist, dass das Netzteil für einen stabilen Betrieb mindestens 1200 mA liefern können sollte. Theoretisch ist es auch möglich, den Raspberry Pi über die USB-A-Anschlüsse mit Spannung zu versorgen, jedoch wird hierdurch der interne Überspannungsschutz umgangen [6].

[4] http://www.arduino.cc/en/Main/Products (zuletzt aufgerufen am 25.08.2015).
[5] http://www.arduino.cc/en/Main/ArduinoBoardUno (zuletzt aufgerufen am 25.08.2015).
[6] http://www.arduino.cc/en/Main/ArduinoBoardMini (zuletzt aufgerufen am 25.08.2015).
[7] http://www.arduino.cc/en/Main/ArduinoBoardProMini (zuletzt aufgerufen am 25.08.2015).
[8] http://www.arduino.cc/en/Main/ArduinoBoardMicro (zuletzt aufgerufen am 25.08.2015).
[9] http://www.arduino.cc/en/Main/ArduinoBoardNano (zuletzt aufgerufen am 25.08.2015).
[10] https://www.raspberrypi.org (zuletzt aufgerufen am 25.08.2015).

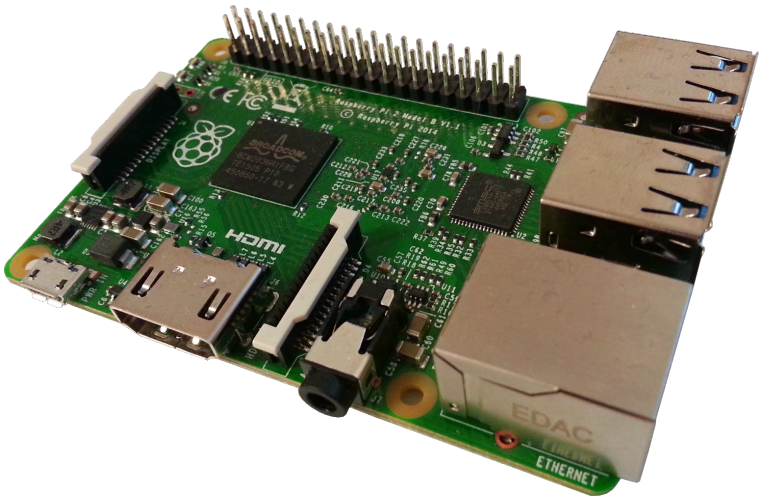


Abbildung 1.2: Der Ein-Platinen-Computer *Raspberry Pi 2*.

GPIO

Die GPIO (General Purpose Input Output)-Schnittstelle bezeichnet die 40-polige Steckerleiste auf dem Raspberry Pi 2, die mehrere unterschiedliche Signale führt. Es existieren insgesamt zwei 3,3 V-Pins, zwei 5 V-Pins, acht GND-Pins, 26 GPIO-Pins sowie zwei Anschlüsse für spezielle I^2C-Identifizierungssignale (ID_SD, ID_SC), mit denen aufgesteckte Erweiterungsmodule für einen automatischen Setup der GPIO-Signale identifiziert werden können. Die GPIO-Pins arbeiten mit einer Spannung von maximal 3,3 V und können wahlweise als digitaler Eingang oder Ausgang geschaltet werden. Einige der GPIO-Pins können auch für andere Funktionen verwendet werden, beispielsweise als PWM-Ausgangssignal, SPI, I^2C oder zur Implementierung einer seriellen Schnittstelle wie z. B. RS232. Die Pinbelegung ist in Abbildung 5.3 im Anhang dargestellt [3].

Ein einzelner GPIO-Pin kann mit bis zu 16 mA belastet werden, wobei zu beachten ist, dass alle GPIO-Pins zusammen nur 50 mA liefern können [6].

1.3 Verwendete Datenbusse

Zur Entwicklung des Testgerätes ist es weiterhin notwendig, ein zumindest grobes Verständnis über die vom Arduino Leonardo-Board unterstützten Datenbusse bzw. Kommunikationsmöglichkeiten zu erlangen. Diese werden nachfolgend erläutert.

1.3.1 I^2C / TWI

Der I^2C-Bus (Inter Integrated Circuit Bus) stammt aus dem Jahre 1982 und wurde für die serielle Verbindung von integrierten Schaltungen innerhalb eines Gerätes entwickelt. Aus

lizenzrechtlichen Gründen existiert auch noch die Bezeichnung TWI (Two Wire Interface), die dasselbe Bus-System darstellt. Die räumliche Ausdehnung eines I^2C-Busses kann lediglich wenige Meter betragen [3].

Für die Kommunikation werden zwei Verbindungen und eine Masse-Leitung benötigt. Jedes I^2C-fähige Bauelement kann entweder als Master oder als Slave arbeiten, verfügt über eine eigene Adresse und kann sowohl Daten senden als auch empfangen. Es ist möglich, dass an einem Bus mehrere Master existieren (Multimasterbus). Für die Datenübertragung sind zwei Leitungen zwischen den Bus-Teilnehmern zuständig: eine Taktleitung (SCL, Serial Clock Line) und eine Datenleitung (SDA, Serial Data Line). Intern sind alle Busteilnehmer Wired-AND-verknüpft, was bedeutet, dass die beiden Datenbussignale, die über Pullup-Widerstände mit der positiven Versorgungsspannung verbunden sind, Low-Signal führen, sobald ein Teilnehmer dieses an den Bus anlegt. Das Taktsignal wird stets von einem Master generiert. Wenn der Slave mehr Zeit benötigt, kann er die Taktleitung auf Low-Pegel halten und so den Master verlangsamen. Bei Inaktivität des Busses befinden sich die Signale SCL und SDA auf High-Pegel. Sobald das Datensignal (SDA) von High auf Low wechselt, während das Taktsignal (SCL) auf High verbleibt, beginnt die Datenübertragung (siehe Abbildung 1.3). Danach wird mit jedem Taktimpuls ein Bit

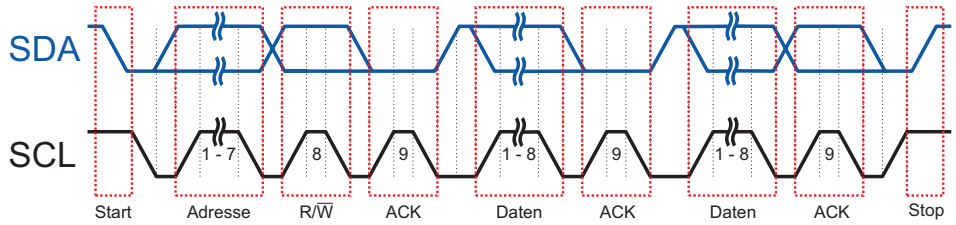


Abbildung 1.3: Beispielhafte Datenübertragung via I^2C [3].

übertragen, wobei das Datenbit nur dann gültig ist, wenn sich der Wert während einer Clock-High-Phase nicht ändert. Zunächst wird eine 7 Bit lange Slave-Adresse gesendet. Danach folgt ein Bit, das die Richtung der Datenübertragung festlegt. Der anschließende 8 Bit lange Datentransfer wird vom Empfänger bestätigt (ACK), indem er SDA auf Low setzt. Das Ende der Datenübertragung wird durch einen Low-High-Übergang von SDA gekennzeichnet, während SCL High-Pegel aufweist [3].

1.3.2 SPI

Das Serial Peripheral Interface (SPI) ist eine synchrone Schnittstelle, die für möglichst hohe Datenraten konzipiert wurde und überwiegend für die Kommunikation zwischen Mikrocontrollern und Peripheriebausteinen genutzt wird. Die räumliche Ausdehnung ist tendenziell kleiner als die des I^2C-Busses und beschränkt sich somit auf relativ kurze Verbindungen. SPI entspricht keinem herkömmlichen Bussystem, da jede Einheit eine eigene

Leitung benötigt und auch nur ein lockerer Standard bezüglich der Spezifikationen existiert [3].

In der Standardkonfiguration werden für die Kommunikation zwischen einem Master und einem Slave zwei Steuer- ($\overline{SS}$, SCK) und zwei Datenleitungen (MOSI, MISO) sowie eine Masse-Leitung benötigt. Mit $\overline{SS}$ (Slave Select) wird der Slave ausgewählt, SCK (Serial Clock) ist die Taktleitung, MOSI (Master Out Slave In) die Datenleitung vom Master zum Slave und MISO (Master In Slave Out) die Datenleitung vom Slave zum Master. Es können beliebig viele Slaves an den Bus angeschlossen werden, jedoch nur ein Master. Der Master erzeugt das Taktsignal (SCK) und wählt über $\overline{SS}$ den Slave aus. Die Signalleitungen eines nicht ausgewählten Slaves sind hochohmig. Über die beiden Datenleitungen wird bidirektional jeweils ein Byte ausgetauscht. Pro Taktperiode wird ein Bit übertragen. Es können grundsätzlich mehrere Bytes hintereinander übertragen werden, jedoch ist in der Spezifikation nicht eindeutig festgelegt, ob zwischen jedem Wort das $\overline{SS}$-Signal kurz auf High-Pegel gesetzt werden muss oder nicht. Wenn das $\overline{SS}$-Signal endgültig auf High-Pegel gesetzt wird, ist die Übertragung beendet. Für die Datenübertragung existieren vier verschiedene Betriebsmodi, die sich darin unterscheiden, wann die Taktleitung (SCK) aktiv ist und bei welcher Taktflanke die Daten übernommen werden [3].

ISP / ICSP

Die SPI-Schnittstelle ermöglicht (sofern vom Mikrocontroller unterstützt) eine In-System-Programmierung (ISP oder auch ISCP: In-Circuit Serial Programming), also das Programmieren eines Mikrocontrollers, ohne ihn aus der Schaltung entfernen zu müssen. Hierfür ist in der Regel zusätzlich noch eine Verbindung zum Reset-Pin des Mikrocontrollers notwendig [7].

2 Entwicklung und Aufbau des Testgerätes

In diesem Kapitel werden zunächst der prinzipielle Aufbau von Testgeräten sowie typische Messgrößen erläutert und anschließend wird die Entwicklung und der Aufbau des in dieser Arbeit behandelten Prüfgerätes vorgestellt.

2.1 Prinzipieller Aufbau von Testgeräten

Apparaturen zum Testen von elektronischen Schaltungen sind in der Industrie längst Standard. Das Einsatzgebiet solcher Testgeräte umfasst nicht nur die Sicherstellung der Funktionsfähigkeit von einfachen Leiterplatten, sondern insbesondere auch den Test von Integrierten Schaltungen und komplexen Steuerungen. Ein allgemeiner und gängiger Begriff für diese Testapparaturen ist die Bezeichnung *Automatic Test Equipment*, der nachfolgend durch *ATE* abgekürzt wird. ATE ist ein sehr allgemein gehaltener Begriff; die einzelnen ATEs verschiedener Hersteller unterscheiden sich zum Teil sowohl in der Bedienung als auch im Funktionsumfang erheblich voneinander. In der Regel enthalten diese Teststationen diverse Spannungsversorgungen, Messgeräte und einen oder mehrere Rechner zur Steuerung des Testablaufes. Der elektrische Kontakt zu dem zu testenden Gerät wird mittels eines sog. *Device Interface Boards (DIB)* hergestellt, das mit dem ATE verbunden ist. Während das ATE zumeist als universell einsetzbare Teststation ausgelegt ist (General-Purpose Tester), muss das DIB speziell an das zu testende Gerät angepasst sein [8].

Nachfolgend werden die wichtigsten Hardwarekomponenten eines ATEs vorgestellt.

2.1.1 DC-Multimeter

Ein wichtiger Bestandteil von Testgeräten sind ein oder mehrere DC-Multimeter zum Messen von Strömen und Spannungen. Meist existiert neben einem schnellen hochauflösenden Multimeter noch ein langsamer arbeitendes ultrahochauflösendes Voltmeter, welches für Kalibrierungszwecke eingesetzt wird. Das Multimeter besitzt üblicherweise einen Instrumentenverstärker mit hohem Eingangswiderstand zum Messen von einfachen Spannungen und Spannungsdifferenzen. Außerdem sind meist noch ein Tiefpassfilter, ein programmierbarer Verstärker zum Skalieren der Eingangsspannung, ein Analog-Digital-Umsetzer (ADU), eine Sample-and-Difference-Schaltung und ein Multiplexer zum Auswählen des Eingangs-Signals vorhanden. Die Sample-and-Difference-Schaltung erlaubt die Messung von Spannungsdifferenzen zweier großer DC-Spannungen mit hoher Genauigkeit, indem zuerst eine Spannung abgetastet und diese dann im zweiten Durchgang von der anderen Spannung subtrahiert wird [8].

2.1.2 Spannungs- / Stromversorgungen

Die meisten Test-Apparaturen besitzen Spannungs- und Stromversorgungen, deren Spannungen bzw. Ströme individuell eingestellt werden können. Für Spannungsversorgungen mit sehr hoher Genauigkeit wird die sog. Vierdraht-Technologie eingesetzt (siehe Abbildung 2.1). Hierdurch wird stets sichergestellt, dass der Spannungsabfall an den Leitungswiderständen die vorgegebene Spannung nicht verfälscht. Dies wird dadurch erreicht, indem die Spannungsdifferenz, die über dem Lastwiderstand abfällt, auf den invertierenden Eingang des Operationsverstärkers (OPV) zurückgeführt wird. Falls diese beispielsweise zu niedrig ist, erhöht der OPV seine Ausgangsspannung so lange, bis die Spannungsdifferenz zwischen den beiden Eingängen 0 V beträgt und somit die Spannung am Lastwiderstand genau dem Sollwert entspricht. Oftmals existieren zudem noch sehr genaue Referenzspannungsquellen, die insbesondere auch als Referenzspannungen für Analog-Digital- und Digital-Analog-Umsetzer verwendet werden, für den Fall, dass die Messung sehr genau erfolgen muss und die herkömmliche Spannungsquelle verrauscht ist [8].

2.1.3 Relais / Relais-Matrizen

Für den Funktionstest ist es notwendig, flexible Verbindungen zwischen unterschiedlichen Signalen und Messgeräten herstellen zu können. Für diesen Zweck gibt es sog. Relais-Matrizen, die dies durch individuelles Schalten unterschiedlicher Relais ermöglichen. Die genaue Architektur dieser Matrizen unterscheidet sich je nach Hersteller und Einsatzgebiet des ATEs. Aufgrund der zum Teil sehr hohen Spannungsspitzen, die beim Schalten von Relais entstehen können, muss stets auf eine entsprechende Schutzbeschaltung beim Bau von Relais-Matrizen geachtet werden (z. B. Freilaufdiode). Auch außerhalb der Matrizen werden elektromechanische Relais häufig eingesetzt, da sie wie andere Schalter auch eine niederohmige Verbindung ermöglichen. Als eine der wenigen beweglichen Teile eines Testgerätes sind sie zugleich als Schwachstelle anzusehen. Nach einigen hundert Millionen Schaltzyklen treten oftmals Defekte auf. Eine zuverlässigere Alternative zum normalen elektromechanischen Relais stellt das Reed-Relais dar, das sich vom inneren Aufbau her vom herkömmlichen Relais unterscheidet. Für eine möglichst lange Lebensdauer sollte stets darauf geachtet werden, keine Spitzenströme zu schalten, die z. B. durch einen kurzgeschlossenen Kondensator auftreten können. Weiterhin ist es sinnvoll, die Schaltkontakte so zu verschalten, dass die am häufigsten verwendete Position bei inaktivem Relais vorherrscht. Mögliche Alternativen zum elektromechanischen Relais sind beispielsweise aus Halbleitern aufgebaute Solid-State-Relais und für Ultrahochfrequenz-Anwendungen sog. MEMS (micro-electro-mechanical systems)-Schalter [8].

2.2 Typische DC-Messgrößen und -verfahren

Bei digitalen Schaltungen ist ein vorrangiges Ziel das Testen der Logik, was durch entsprechende Eingaben und anschließender Messung der Ausgaben erfolgt. Das Messen

von analogen Parametern ist jedoch auch von großer Bedeutung. In diesem Abschnitt werden die gängigsten analogen Messgrößen bzw. -verfahren vorgestellt.

2.2.1 Durchgangsprüfung

Bevor das zu prüfende Gerät getestet werden kann, muss es mittels des Device Interface Boards mit dem ATE verbunden werden. Dies erfolgt in der Regel mit Hilfe von Steckern und/oder speziellen Kontaktier-Vorrichtungen. Hierbei kann es unter Umständen zu fehlerhaften Kontaktierungen kommen. Wenn solch eine Durchgangsprüfung nicht durchgeführt wird, kann es sein, dass funktionierende Geräte vom ATE als defekt erkannt werden, da dieser nicht unterscheiden kann, ob das fehlerhafte Signal z. B. von einer defekten Halbleiterstruktur oder von einem fehlenden Kontakt herbeigeführt wurde. In den meisten Fällen wird die Durchgangsprüfung durch Erkennen von ESD-Schutzbeschaltungen der einzelnen Pins vollzogen. Die ESD-Schutzbeschaltung schützt das Gerät bzw. die Integrierte

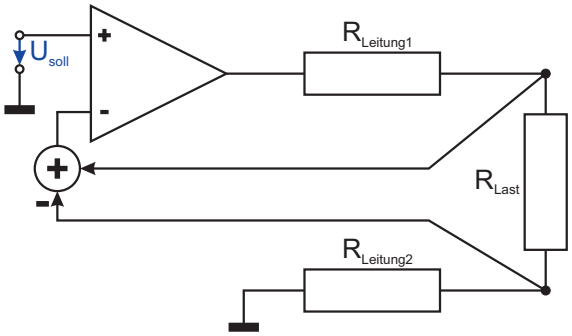


Abbildung 2.1: Hochpräzise Spannungsversorgung mit Vierdraht-Technologie [8].

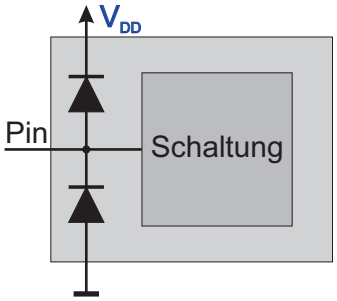


Abbildung 2.2: Mögliche Schutzbeschaltung gegen elektrostatische Entladungen für Pin mittels zweier Dioden [8].

Schaltung vor elektrostatischer Entladung. Das kann beispielsweise mit Hilfe zweier Dioden geschehen (siehe Abbildung 2.2). Diese verhindern, dass die Spannung am Pin den erlaubten Bereich verlässt. Um nun zu testen, ob ein Pin Kontakt zum Testgerät hat, werden alle Versorgungsspannungen des zu testenden Gerätes auf GND-Potential geschaltet und anschließend wird mittels einer Konstantstromquelle ein Strom in den Pin eingeprägt. Anhand der am Pin messbaren Spannung kann nun festgestellt werden, ob der Pin angeschlossen ist oder nicht. Die andere Diode kann getestet werden, indem ein entgegengesetzter Strom eingeprägt wird. Typischerweise beträgt der Strom zwischen 100 μA und 1 mA. Wenn die gemessene Spannung 0 V beträgt, liegt ein Kurzschluss zwischen dem Pin und GND oder V_{DD} vor. Falls die Spannung am Pin der Versorgungsspannung entspricht, bedeutet dies, dass kein Kontakt zum Pin besteht. Für den fehlerfreien Fall sollte eine Spannung gemessen werden, die in etwa 0, 7 V (Schwellspannung der Diode)

entspricht. Wenn die angrenzenden Pins während des Tests ebenfalls auf GND-Potential gelegt werden, kann zusätzlich noch sichergestellt werden, dass kein Kurzschluss zu diesen Pins besteht. Alle Pins, die nicht über eine solche Schutzbeschaltung verfügen, müssen anderweitig getestet werden, z. B. durch Messen eines möglicherweise ausreichend großen Leckstromes [8].

2.2.2 Leckstrom

Jeder Ein- und Ausgangspin verfügt über einen Leckstrom. Bei einem hochohmigen digitalen oder analogen Eingang zeigt sich dieser parasitäre Effekt in Form eines hinein- oder herausfließenden Stromes, wenn an den Pin eine Spannung angelegt wird. Ein Leckstrom kann ebenfalls bei Ausgangs-Pins gemessen werden, die in einem sog. „non-driving"-Modus betrieben werden. Typischerweise liegt dieser Strom unter einem Wert von 1 μA. Einer der Hauptgründe für das Messen dieses Leckstromes ist, dass hiermit oftmals physikalische Defekte festgestellt werden können, falls der gemessene Strom zu weit von der Norm abweicht. In der Regel werden zwei Werte für den Leckstrom gemessen: einmal mit einer angelegten Spannung nahe der Versorgungsspannung (I_{IH}) und einmal bei GND-Potential (I_{IL}). Bei analogen Eingängen erfolgen diese Messungen des Leckstromes oftmals bei definierten Spannungen, die sich aus den Informationen des Datenblattes ergeben [8].

2.2.3 Stromversorgung

Eine einfache Möglichkeit, gravierende Defekte festzustellen, ist die Messung des Stromes des Versorgungsnetzteiles. Viele Defekte resultieren in einem deutlich überhöhten Stromverbrauch oder führen dazu, dass gar kein Strom fließt. Dieser Test wird für gewöhnlich zu Beginn durchgeführt, da hiermit sehr schnell defekte Geräte aussortiert werden können. Die Schwierigkeit bei diesem Test besteht darin, festzulegen, unter welchen Bedingungen der Stromverbrauch gemessen werden sollte und wie hoch dieser sein darf. Es ist generell die sicherste Variante, den Stromverbrauch des Gerätes unter Worst-Case-Bedingungen zu messen [8].

2.2.4 Impedanz

Für analoge Eingänge ist die Eingangsimpedanz bzw. der Eingangswiderstand (bei DC) eine ebenfalls übliche Spezifikation. Für den linearen Fall kann dieser ermittelt werden, indem z. B. nacheinander zwei unterschiedlich große Spannungen angelegt und die jeweiligen Ströme gemessen werden. Die resultierende Eingangsimpedanz berechnet sich mit Hilfe des Ohm'schen Gesetzes aus der Spannungs- und Stromdifferenz. Die Messung der Ausgangsimpedanz erfolgt nach dem gleichen Schema, jedoch empfiehlt es sich in den meisten Fällen aufgrund des niedrigen Impedanz-Wertes, den Strom vorzugeben und die Spannung zu messen [8].

2.2.5 Schwellspannung

Die Input-High-Spannung (U_{IH}) und die Input-Low-Spannung (U_{IL}) definieren die Schwellen für digitale Eingänge zur Unterscheidung zwischen High und Low. Der Bereich zwischen diesen beiden Spannungen ist nicht definiert. U_{IH} und U_{IL} können durch Anlegen verschiedener Spannungen ermittelt werden, jedoch wird der Test meist direkt mit den im Datenblatt angegebenen Werten durchgeführt. Für digitale Ausgänge können die Spannungen U_{OH} und U_{OL} gemessen werden. U_{OH} ist hierbei die minimale garantierte Spannung für einen Ausgang mit High-Pegel (bei einem Strom I_{OH}) und U_{OL} stellt die maximale garantierte Spannung für einen Ausgang mit Low-Pegel dar (bei $I = I_{OL}$) [8].

2.2.6 Kurzschlussstrom

Sofern ein digitaler Ausgang eine eingebaute Strombegrenzung besitzt, können die Kurzschluss-Ströme I_{OSH} und I_{OSL} gemessen werden. Zum Messen von I_{OSL} wird der Ausgang auf Low-Pegel geschaltet und die Betriebsspannung an diesen Pin angelegt. Zum Ermitteln von I_{OSH} wird der Ausgang mit High-Pegel auf GND geschaltet [8].

2.3 Systemansatz

Die Grundlage des Testgerätes bildet ein Raspberry Pi 2. Dieser ist für die Programmierung des zu überprüfenden Mikrocontrollers und für die grundlegende Steuerung des Testes zuständig. Außerdem kann dadurch ein Datenaustausch mit einem Computer per Ethernet stattfinden, wodurch zum Beispiel eine Änderung des auf den Mikrocontroller aufzuspielenden Standardprogramms und des erstellten Report möglich ist. Zum Verbinden des Raspberry Pis mit dem Arduino-Board und zum Herstellen der Funktionsfähigkeit des Testgerätes wird eine entsprechende Platine entwickelt. Der prinzipielle Aufbau des Testgerätes ist in Abbildung 2.3 dargestellt. Die zu erstellende Platine verbindet hierbei alle wesentlichen Komponenten miteinander. Mit dem Raspberry Pi erfolgt die Verbindung durch die 40-polige GPIO-Schnittstelle sowie per USB. Des Weiteren dient die an ein Netzteil angeschlossene Platine zur Stromversorgung des Raspberry Pis und des zu testenden Arduino-Boards. Die Platine soll so konzipiert sein, dass nicht nur das Arduino-Board „Leonardo“, sondern auch eine Vielzahl anderer Mikrocontroller-Boards getestet werden können. In jedem Fall sind hierbei noch die Arduino-Boards Uno, Mini, Pro Mini, Micro und Nano zu unterstützen. Eine Verbindung zwischen dem zu testenden Board und der Platine kann via USB, einem Stromversorgungsanschluss (PWR) und einem 64-poligen Stecker erfolgen. Jedes Board benötigt hierfür einen individuellen Adapter (DIB), der an den 64-poligen Stecker angeschlossen wird und die Verbindung zum Mikrocontroller-Board herstellt. Für den Adapter ist keine zusätzliche Elektronik nötig, sondern dieser besteht im Wesentlichen nur aus Steckern und Leitungen. Neben einem individuellen Adapter muss auch die Software an das zu testende Board angepasst werden. Im Rahmen dieser Arbeit erfolgt dies lediglich für das Arduino-Board Leonardo.

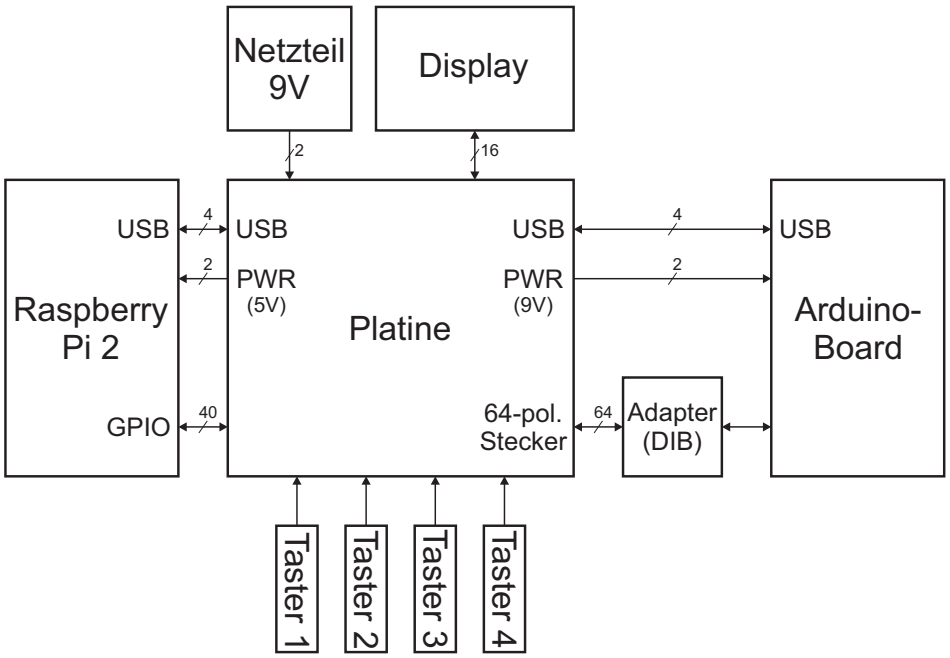


Abbildung 2.3: Prinzipieller Aufbau des Testgerätes.

2.4 Elektronik

Bezüglich der Elektronik wird im Rahmen dieser Arbeit ein Schaltplan entwickelt und ein Platinenlayout erstellt. Dies gilt sowohl für das Testgerät, als auch für den notwendigen Adapter für das Arduino Leonardo-Board.

2.4.1 Anforderungen

Wie bereits in der Einleitung erwähnt, sind die digitalen Ein- und Ausgänge, die analogen Eingänge sowie die Funktionsfähigkeit der SPI- und I^2C-Schnittstelle zu prüfen. Zum Testen der digitalen und analogen Eingänge ist es notwendig, eine analoge Spannung mittels eines Digital-Analog-Umsetzers (DAU) erzeugen zu können. Hierdurch können die analogen Eingänge mit unterschiedlichen Spannungen getestet werden und bei den digitalen Eingängen ist es möglich, die Schwellspannungen zu ermitteln. Ein Analog-Digital-Umsetzer (ADU) ist erforderlich, um Spannungen zu messen. Zum Kontaktieren unterschiedlicher Pins wird eine Auswahlschaltung benötigt. Schutzbeschaltungen wie z. B. Strombegrenzungen sind erforderlich, damit die Zerstörung des Testgerätes beim Anschließen defekter Boards verhindert wird. Außerdem ist ein Stromsensor zum Messen des Versorgungsstromes sinnvoll, da hierdurch festgestellt werden kann, ob ein Versorgungsanschluss defekt ist. Eine Durchgangsprüfung soll ebenfalls möglich sein, um ermitteln zu können, ob die Datenpins des zu testenden Boards richtig kontaktiert sind oder ob

möglicherweise eine fehlerhafte Verbindung vorliegt. Insgesamt muss das Testgerät über genügend Anschlusspins verfügen, damit es universell auch für andere Boards eingesetzt werden kann. Hierbei sind insbesondere die Tabellen 5.2 und 5.3 im Anhang zu berücksichtigen.

2.4.2 Entwicklung des Schaltplanes

Das Funktionsprinzip der zu erstellenden Platine wird nachfolgend anhand von Abbildung 2.4 und 2.5 erläutert. Zunächst wird ersteres betrachtet.

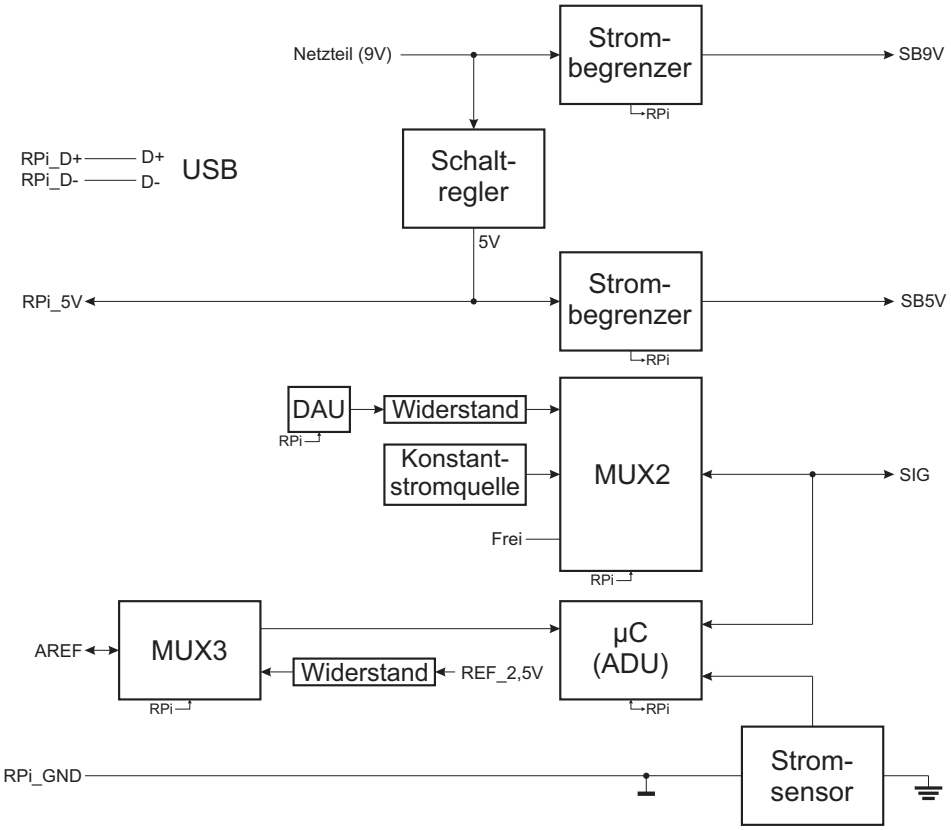


Abbildung 2.4: Blockschaltplan Teil 1 von Platine.

Die Platine wird über ein Netzteil mit einer Spannung von 9 V versorgt. Zum einen wird mit diesen 9 V ein Schaltregler gespeist, der daraus 5 V erzeugt und zum anderen kann damit über einen zwischengeschalteten Strombegrenzer das angeschlossene Mikrocontroller-Board versorgt werden (*SB9V*). Für den Fall eines zu hohen Stromverbrauches, wie es beispielsweise bei einem Kurzschluss der Fall ist, wird die Strombegrenzung aktiv und begrenzt einerseits den Strom, indem die Ausgangsspannung heruntergesetzt

wird und andererseits erfolgt eine Mitteilung an den Raspberry Pi (RPi) durch ein Ausgangssignal. Die Strombegrenzung für die 5 V-Spannung funktioniert analog. Mit *SB5V* kann ebenfalls das angeschlossene Board versorgt werden. Des Weiteren dient der 5 V-Ausgang des Schaltreglers zur Spannungsversorgung des Raspberry Pis. Mit Hilfe eines Digital-Analog-Umsetzers (DAU), der vom Raspberry Pi angesteuert wird, ist es möglich, eine analoge Spannung zu generieren, die über einen Widerstand und einen Multiplexer (*MUX2*) an den ausgewählten Pin (*SIG*) angelegt werden kann. Der Widerstand fungiert in diesem Fall ebenfalls als Strombegrenzung, da die Möglichkeit besteht, dass bei einem angeschlossenen defekten Board der ausgewählte Pin bei einem Eingangs-Test als Ausgang geschaltet ist, was in diesem Fall einen Kurzschluss zur Folge haben würde. Der Widerstand reduziert diesbezüglich den Strom soweit, dass das Testgerät nicht beschädigt wird. Dieser Fall kann mit Hilfe des ADUs im Mikrocontroller (μC) detektiert werden, da die Spannungsdifferenz über dem Widerstand abfällt und die gemessene Spannung somit von der vorgegebenen abweicht. Mit der Konstantstromquelle kann eine Durchgangsprüfung (siehe Kapitel 2.2.1) durchgeführt und somit festgestellt werden, ob ein Datenpin, der über eine ESD-Schutzbeschaltung verfügt, korrekt an das Testgerät angeschlossen ist. Diese Detektion erfolgt ebenfalls über das Messen der Spannung an *SIG*. Für den Fall, dass die Spannung eines ausgewählten Ausgangspins gemessen werden soll, wird *MUX2* auf den freien Pin geschaltet. Mit *MUX3* kann einerseits eine Referenzspannung von 2,5 V an den AREF-Pin des zu testenden Boards angelegt werden und andererseits ist auch eine Spannungsmessung möglich, für den Fall, dass die interne Referenzspannung des angeschlossenen Mikrocontrollers aktiv ist. Der Widerstand dient hierbei ebenfalls als Strombegrenzung, falls die 2,5 V-Spannung an *AREF* angelegt wird und die interne Referenzspannung des angeschlossenen Boards aktiviert ist. In die Masseleitung ist ein Stromsensor integriert, mit dem der Stromverbrauch des angeschlossenen Mikrocontroller-Boards gemessen werden kann. Hierdurch können gravierende Defekte am Board festgestellt werden (siehe Kapitel 2.2.3). Beispielsweise kann somit bei einem gemessenen Strom von 0 A geschlussfolgert werden, dass der Versorgungsanschluss einen Defekt aufweist. Der Grund, weshalb der Stromsensor in die Masseleitung integriert ist, liegt darin, dass hierdurch sowohl der Strom bei einer Versorgungsspannung von 9 V, als auch bei einer 5 V-Versorgung des Boards gemessen werden kann und somit kein zweiter Stromsensor nötig ist.

Nun wird Abbildung 2.5 betrachtet. Das Kürzel *PD* bedeutet, dass die Leitung über einen Pull-Down-Widerstand mit Masse verbunden ist. *PU* kennzeichnet eine Verbindung zu 5 V mittels eines Pull-Up-Widerstandes. Rechts sind der 64-polige Stecker, der Power (PWR)- sowie der USB-Anschluss abgebildet. Über diese drei Anschlüsse wird das zu testende Board mit der Platine verbunden. Am 64-poligen Stecker erfolgt diese Verbindung unter Verwendung eines entsprechenden Adapters. Der PWR-Anschluss stellt die 9 V-Spannungsversorgung durch die Hohlbuchse dar. Über die Relais RE2, RE3 und RE4 kann ausgewählt werden, über welche Schnittstelle das Board mit Spannung versorgt werden soll. Dies ist notwendig, damit überprüft werden kann, ob alle Anschlüsse intakt sind. Für den Fall, dass eine Datenübertragung über USB erfolgen soll, muss das angeschlossene Board auch zwingend hierdurch gespeist werden.

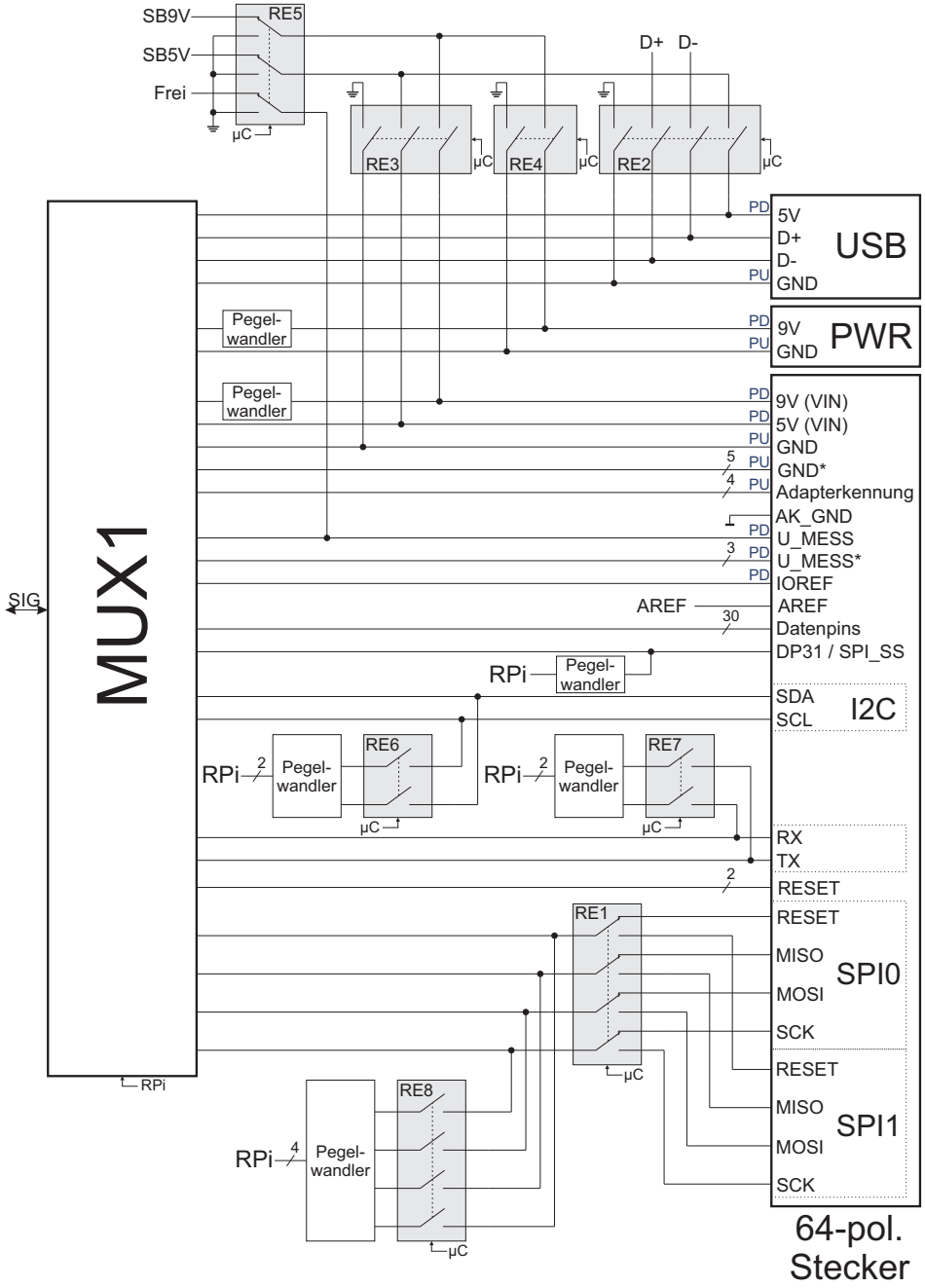


Abbildung 2.5: Blockschaltplan Teil 2 von Platine.

An dieser Stelle erfolgt eine Erläuterung der Signale des 64-poligen Steckers. *9 V* dient zum Anschluss an den VIN-Eingang, der in der Regel mit dem Versorgungsanschluss der Hohlbuchse verbunden ist. Der *5 V*-Pin ist nur bei speziellen Boards notwendig, die über keinen internen Spannungsregler verfügen und direkt mit 5 V betrieben werden. *GND* wird an einen beliebigen GND-Pin angeschlossen. Bei allen anderen GND-Pins erfolgt die Kontaktierung durch *GND**. Über diese Pins ist keine Stromversorgung möglich, sondern hiermit kann lediglich überprüft werden, ob sie mit dem Haupt-GND-Pin verbunden sind. Dies kann beispielsweise wie folgt geschehen: An *GND* wird über RE3 das Masse-Potential angelegt. Da die *GND**-Anschlüsse mit Pullup-Widerständen (PU) verbunden sind, kann an diesen eine Spannung von 5 V gemessen werden, falls sie durch einen Defekt nicht mit dem Haupt-GND-Pin verbunden sind und somit nicht auf GND-Potential gebracht werden können. Über den Multiplexer *MUX1* ist die Auswahl und Messung eines beliebigen Signals möglich (siehe Abbildung 2.4). Die Pins zur *Adapterkennung* sowie *AK_GND* dienen zur automatischen Identifikation des Adapters. Hierdurch kann die Software des Raspberry Pis so implementiert werden, dass der Adapter beim Anstecken automatisch erkannt wird. Durch entsprechende Verdrahtung ist somit eine Unterscheidung von bis zu 15 Boards möglich. *U_MESS* wird mit der Betriebsspannung des zu testenden Mikrocontrollers verbunden und die restlichen 3,3 V- bzw. 5 V-Ausgangspins mit *U_MESS**. Hiermit kann die jeweilige Spannung gemessen bzw. aufgrund der Pull-Down-Widerstände auch festgestellt werden, ob eine Verbindung zum Pin besteht. *IOREF* sowie *AREF* werden mit den gleichnamigen Pins verbunden, sofern diese auf dem Board existieren. Die Verbindung aller Pins mit integrierten Schnittstellen erfolgt entsprechend der jeweiligen Bezeichnungen. Die restlichen Pins werden an die *Datenpins* angeschlossen. Die Relais RE6, RE7 und RE8 dienen dazu, die Verbindung der Schnittstellen zwischen dem zu testenden Board und dem Raspberry Pi zu unterbrechen. Dies ist notwendig, da die Schnittstellen zumeist auch als normale I/O-Pins verwendet werden können und ein entsprechender Test bei bestehender Verbindung zum Raspberry Pi nicht richtig möglich wäre. Mit RE1 kann zwischen SPI0 und SPI1 umgeschaltet werden. Das dient der Kompatibilität mit dem Arduino Uno, da dieser noch einen zusätzlichen SPI-Anschluss für den USB-Controller besitzt (siehe Tabelle 5.2 im Anhang). Mit dem Relais RE5 können die entsprechenden Signale für den Durchgangstest auf GND-Potential gelegt werden. Alle Relais werden durch den internen Mikrocontroller (μC in Abbildung 2.4) angesteuert, der wiederum die auszuführenden Befehle vom Raspberry Pi erhält.

Schaltplan

Auf Grundlage des zuvor erläuterten Blockschaltplanes wurde ein funktionsfähiger Schaltplan entwickelt. Dieser erstreckt sich über drei Seiten und befindet sich im Anhang in Kapitel 5.3 sowie auf der beigelegten DVD (DVD/Schaltpläne/Testgerät - Schaltplan.pdf). Alle nachfolgend erwähnten Datenblätter sind unter DVD/Datenblätter/*.pdf zu finden.

Zunächst wird Seite 1 des Schaltplanes betrachtet. Das 9 V-Netzteil wird an die Hohlbuchse *K10* angeschlossen. An *K11* erfolgt der Anschluss eines Schalters, mit dem das

Testgerät an- und ausgeschaltet werden kann. Der Schaltregler *IC14* erzeugt mit der gegebenen Beschaltung aus den 9 V eine Ausgangsspannung von 5 V, die mit bis zu 3 A belastet werden kann. Die gegebene Beschaltung ergibt sich weitestgehend aus dem zugehörigen Datenblatt. Im Schaltplan wurde ebenfalls berücksichtigt, wie genau die Leiterbahnen für einen möglichst stabilen Betrieb angeschlossen werden müssen. So sollten beispielsweise die Masse-Anschlüsse der einzelnen Bauteile sternförmig in einem Punkt miteinander verbunden sein. Dies ist ebenfalls dem Datenblatt zu entnehmen. Die USB-Buchse *K2* bildet den Spannungsversorgungsanschluss des Raspberry Pis. Über *K5* wird die Platine mit der GPIO-Schnittstelle des Raspberry Pis verbunden. Hierüber werden einige Bauteile der Platine mit 3,3 V versorgt. Ebenfalls über *K5* werden Verbindungen zu der SPI- und den I^2C-Schnittstellen sowie zum UART des Raspberry Pis hergestellt. Die interne Kommunikation zwischen mehreren Bauteilen des Testgerätes erfolgt über den I^2C-Bus. Da ein angeschlossenes defektes Gerät diesen Bus blockieren könnte, wird das zu testende Mikrocontroller-Board an eine separate I^2C-Schnittstelle angeschlossen. Weiterhin werden die Multiplexer auf Seite 2 und 3 des Schaltplanes sowie das Relais *RE9* direkt über die GPIO-Pins angesteuert. Letzteres ist dafür zuständig, die Spannung der Pegelwandler auf der linken Seite des Schaltplanes zwischen 3,3 V und 5 V umzuschalten, damit sowohl 3,3 V- als auch 5 V-Boards getestet werden können. Als Eingänge für den Rasbperry Pi sind vier Signale der Eingabetaster vorhanden sowie eines, das die aktive Strombegrenzung der 5 V-Versorgungsspannung signalisiert und ein weiteres für die 9 V-Strombegrenzung. Die Pegelwandler mit den Feldeffekt-Transistoren (FETs) *T2* bis *T9* sowie *T13* dienen dazu, den 3,3 V-Spannungspegel des Raspberry Pis in den Pegel des zu testenden Mikrocontroller-Boards umzuwandeln. Dieser kann wie bereits erwähnt zwischen 3,3 V und 5 V umgeschaltet werden. Die zwei Pegelwandler mit *T10* und *T11* wandeln den Pegel auf 5 V für *IC6* und *IC11* auf Seite 2. Die bidirektionale Pegelwandler-Schaltung wurde im Prinzip von [9] übernommen. *IC15* und *IC16* bilden die Strombegrenzungen, die den Strom bei angegebener Beschaltung auf maximal ca. 0,5 A begrenzen. Dies kann dem Datenblatt entnommen werden. Die 9 V-Leitung ist durch einen größeren Widerstand *R30* auf einen etwas niedrigeren Strom begrenzt, da diese andernfalls das Netzteil zu stark belasten würde. Sobald die Strombegrenzungen aktiv sind, werden *Short_9V* bzw. *Short_5V* auf Low-Pegel gesetzt. Über die Anschlussklemmen von *K8* werden die USB-Datenleitungen des Raspberry Pis mit der Platine verbunden.

Auf Seite 2 des Schaltplanes dient *K7* zum Anschluss von Eingabetastern an die Platine. Die zwei Kontakte eines Tasters werden jeweils an Pin 1 und 2 bzw. 3 und 4 usw. angeschlossen. Die Kombination aus Widerstand und Kondensator (beispielsweise *R20* und *C7*) wirkt als Tiefpassfilter und zusammen mit dem nachgeschalteten Schmitt-Trigger (*IC13*) wurde hiermit eine Taster-Entprellung realisiert. Das somit erzeugte Signal wird direkt an den Raspberry Pi weitergeleitet. *IC12a* bildet einen nichtinvertierenden Verstärker mit einem Verstärkungsfaktor von

$$v_U = 1 + \frac{R10}{R11} = 101.$$

Hierbei wird die Spannung, die über einen $0,1\,\Omega$-Shunt in der Masseleitung abfällt, verstärkt. Der maximale Strom, der somit theoretisch korrekt gemessen werden kann (mit einem 10-Bit-ADU), ergibt sich zu

$$I_{\max} = \frac{U_{R9max}}{R9} = \frac{\frac{AREF}{v_U}}{R9} = \frac{AREF}{v_U \cdot R9} = \frac{5\ V}{101 \cdot 0,1\,\Omega} = 0,495\ A$$

mit einer Auflösung von

$$\frac{I_{\max}}{2^{10}} = \frac{0,495\ A}{1024} = 0,483\ mA.$$

Jeder Strom, der darüber hinausgeht, wird ebenfalls fälschlicherweise als dieser Wert gemessen werden. Bei der Auswahl des Operationsverstärkers war ein entscheidendes Kriterium, dass es sich um einen sog. Rail-to-Rail-OPV handelt, d. h. dass die Ein- und Ausgangsspannungen nahezu den gesamten Versorgungsspannungsbereich abdecken können. *IC12b* bildet mit der angegebenen Beschaltung eine einstellbare Konstantstromquelle. Über die beiden Widerstände *R21* und *R22* fällt stets, sofern möglich, eine Spannung von 2,5 V ab. Der minimal einstellbare Strom berechnet sich zu

$$I_{\min} = \frac{2,5\ V}{R21 + R22} = 0,09\ mA$$

und der maximale Strom beträgt

$$I_{\max} = \frac{2,5\ V}{R21} = 0,93\ mA.$$

Mit R22 lässt sich dieser Strom stufenlos variieren. *IC6* ist ein Digital-Analog-Umsetzer (ADU), der vom Raspberry Pi über I^2C angesteuert wird. *IC7a* implementiert die beiden Multiplexer *MUX2* und *MUX3* aus dem Blockschaltplan. Das IC wird mit einer Versorgungsspannung von 5 V betrieben und vom Raspberry Pi mit 3,3 V-Pegel angesteuert. In diesem Fall ist kein Pegelwandler notwendig, da es sich um einen HCT-Typen handelt, der somit TTL-kompatibel ist. Dies trifft ebenso auf *IC1 - IC5* zu. *IC9* erzeugt eine Referenzspannung von 2,5 V, die zum einen am Operationsverstärker *IC12b* für eine Spannungsdifferenz von etwa 2,5 V sorgt und zum anderen über *IC7* als Referenzspannung an das angeschlossene Board angelegt werden kann. Die Beschaltung mit dem Potentiometer *R13* ermöglicht eine genaue Einstellung der Spannung und wurde dem Datenblatt entnommen. Der ATmega8 (*IC11*) übernimmt mehrere Aufgaben. Zum einen werden damit analoge Spannungen gemessen (*PC2*, *ADC6*, *ADC7*) und zum anderen das Display und die Relais *RE1* - *RE8* auf Seite 3 angesteuert. Der ATmega8 kommuniziert hierbei mit dem Raspberry Pi über die I^2C-Schnittstelle. An *K6* wird das Display angeschlossen. Die Anschlussbelegung ergibt sich einerseits aus dem entsprechenden Datenblatt und andererseits aus softwaretechnischen Überlegungen. Laut Datenblatt beträgt die minimale Flussspannung

der Hintergrundbeleuchtung 3,0 V und der maximale Strom 45 mA. Hieraus ergibt sich ein minimaler Vorwiderstand von

$$R52_{min} = \frac{5\ V - 3\ V}{45\ mA} = 44,4\ \Omega.$$

Über *R55* kann die Stärke der Beleuchtung reguliert werden und mit *R53* ist es möglich, die Displayspannung von 0 V bis ca. 0,54 V und somit den Kontrast einzustellen. Durch Anschluss eines Programmiergerätes an *K4* kann der ATmega8 mittels ISP programmiert werden. *IC8* dient als Treiber-Array zur Ansteuerung der Relais auf Seite 3.

Nun wird Seite 3 des Schaltplanes betrachtet. *IC1a* - *IC4a* bilden zusammen mit *IC5a* auf Seite 2 den 64-Kanal-Multiplexer MUX1 des Blockschaltplanes. *K1* ist der 64-polige Stecker, an dem der individuelle Adapter angeschlossen wird. An *K3* und *K9* kann das zu testende Mikrocontroller-Board per Hohlstecker bzw. USB mit der Platine verbunden werden. Die Relais *RE1* - *RE8* erfüllen die bereits im Kapitel des Blockschaltplanes erläuterten Funktionen. An jedes Relais ist zum Schutz der Schaltung vor Überspannung jeweils eine Freilaufdiode geschaltet. Die Widerstände *R56* bis *R119* sind ebenfalls essentiell für die korrekte Funktionsweise des Gerätes. Diese erfüllen in erster Linie die Aufgabe, die Ströme der Multiplexer-Eingänge zu begrenzen. Ohne diese Widerstände könnte der Anschluss kapazitiver Lasten an das Testgerät zur Zerstörung der Multiplexer führen. Diese Problematik ergibt sich aus der Überlegung, dass beim Anschluss einer Kapazität beispielsweise zwischen *9 V (VIN)* und *GND* und dem Schalten von Relais 3, es durchaus möglich ist, dass der GND-Kontakt des Relais einen Bruchteil einer Sekunde später schaltet. Da die Kapazität im Einschaltmoment einen Kurzschluss darstellt, bedeutet dies, dass am Multiplexer-Eingang von der GND-Leitung kurzzeitig 9 V anliegen würden. Eine interne ESD-Schutzbeschaltung verhindert dies im Normalfall, jedoch haben Tests ergeben, dass bei einem kurzzeitigen hohen Eingangsstrom intern eine dauerhaft leitfähige Verbindung zwischen VCC und GND hergestellt wird, was dazu führt, dass sich das Bauteil innerhalb kürzester Zeit stark erwärmt, was zur thermischen Zerstörung des Bauteils führen kann.

2.4.3 Erstellung der Platine

Sowohl der Schaltplan als auch das Platinenlayout wurden mit dem Programm *Target 3001!*[1] erstellt. Das Target-File ist unter DVD/Platinen/Testgerät/Testgeraet.T3001 und die exportieren XGerber-Files unter DVD/Platinen/Testgerät/XGerber/* zu finden. Zusätzlich wurden noch ein EAGLE[2]-Script erstellt (DVD/Platinen/Testgerät/Testgeraet.SCR) sowie ein PDF (DVD/Platinen/Testgerät/Platinenlayout.pdf), in welchem alle relevanten Layer abgebildet sind.

Da die Platine in ein fertiges Kunststoffgehäuse eingebaut wird, mussten bei der Erstellung des Platinenumrisses dessen Maße berücksichtigt werden. Als Gehäuse dient ein *TEKO 363*, dessen Datenblatt auf der DVD zu finden ist (DVD/Datenblätter/TEKO363.pdf).

[1] http://www.ibfriedrich.com (zuletzt aufgerufen am 25.08.2015).
[2] http://www.cadsoft.de/ (zuletzt aufgerufen am 25.08.2015).

Das prinzipielle Platinenlayout ist in Abbildung 2.6 zu sehen. An der oberen Seite befin-

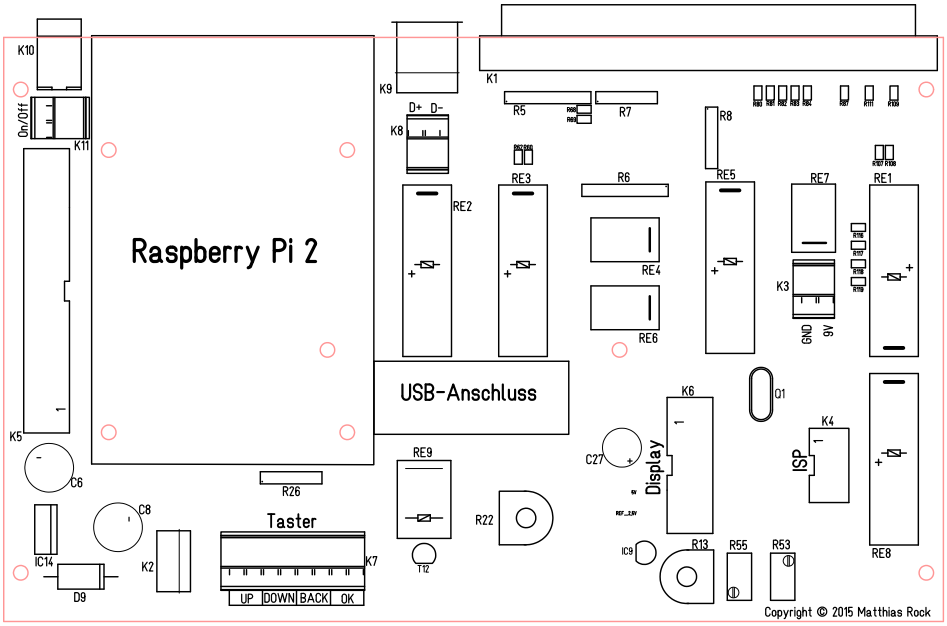


Abbildung 2.6: Prinzipielles Layout von entwickelter Platine.

den sich die Hohlbuchse für die 9 V-Versorgungsspannung sowie die USB-Buchse und der 64-polige Stecker zum Anschluss des zu testenden Boards. Wie in dieser Abbildung zu erkennen ist, existiert auf der Platine eine freie Fläche, auf dem der Raspberry Pi 2 montiert wird. Dies hat einerseits den Vorteil, dass in dem Gehäuse keine zusätzliche Befestigungsmöglichkeit für den Raspberry Pi vorhanden sein muss und andererseits hätte bei anderer Montage unter Umständen ein größeres Gehäuse gewählt werden müssen. Des Weiteren ist hierdurch zusätzlicher Platz für die SMD-Bauteile auf der Unterseite der Platine entstanden. Im Anhang ist jeweils ein Bild der Ober- und Unterseite der Platine aus der 3D-Ansicht in Target zu finden (Abbildung 5.4 und 5.5).

2.4.4 Entwicklung der Adapterplatine für Arduino Leonardo

Aufgabe des Adapters (DIB) ist es, eine Verbindung zwischen den Pins des 64-poligen Steckers des Testgerätes und den Pins des zu testenden Mikrocontroller-Boards herzustellen. Bei der Entwicklung des Schaltplanes sollte folgende Vorgehensweise angewendet werden: Zunächst werden alle Pins miteinander verbunden, die eindeutig einander zuzuordnen sind. Dies ist beispielsweise bei *IOREF*, *AREF*, *RESET* und der Versorgungsspannung *VIN* sowie bei den Schnittstellen-Pins (*I2C_SDA*, *I2C_SCL*, *RX*, *TX*, *S0_RESET*, *S0_MISO*, *S0_MOSI*, *S0_SCK*, *S1_RESET*, *S1_MISO*, *S1_MOSI*, *S1_SCK*) der Fall. Hierbei ist es unerheblich, ob die Schnittstellenpins noch andere Funktionen, wie z. B.

das Messen analoger Spannungen unterstützen. Anschließend wird ein beliebiger GND-Pin mit *GND* verbunden. Alle restlichen GND-Pins werden mit jeweils einem *GND**-Pin verschaltet. Ein Spannungs-Ausgangspin, der mit der Versorgungsspannung des Mikrocontrollers des zu testenden Boards verbunden ist, wird mit *U_MESS* verbunden. Die restlichen Versorgungsspannungs-Ausgangspins (z. B. 5 V und 3,3 V) werden an jeweils einen *U_MESS**-Pin angeschlossen. Danach werden alle übrigen Datenpins mit jeweils einem *DP*-Pin verbunden. Zur Adapterkennung erfolgt die Verschaltung von ein oder mehreren dieser Pins (*AK1 - AK4*) mit *AK_GND*. Diese Kennung muss sich von bereits existierenden unterscheiden.

Prinzipiell kann der Adapter durch Verbinden der Stecker mit einzelnen Leitungen hergestellt werden. Für den Arduino Leonardo wurde eine Platine entwickelt, die ein unkompliziertes Verbinden ermöglicht. Der Schaltplan befindet sich im Anhang in Kapitel 5.3 sowie auf der beigelegten DVD (DVD/Schaltpläne/Adapterplatine_Leonardo - Schaltplan.pdf) und wurde auf Grundlage des Leonardo-Schaltplanes (DVD/Schaltpläne/Arduino_Leonardo - Schaltplan.pdf) entwickelt. Das Target-File ist unter DVD/Platinen/Adapterplatine_Leonardo/Adapter_Leonardo.T3001 und die exportierten XGerber-Files sind unter DVD/Platinen/Adapterplatine_Leonardo/XGerber/* zu finden. Zusätzlich existieren ein EAGLE-Script (DVD/Platinen/Adapterplatine_Leonardo/Adapter_Leonardo.SCR) und ein PDF mit allen darin abgebildeten relevanten Layern (DVD/Platinen/Adapterplatine_Leonardo/Platinenlayout.pdf).

2.5 Aufbau und Montage des Testgerätes

Für den Aufbau des Testgerätes wird zunächst die entwickelte und fertig hergestellte Platine mit allen Bauteilen bestückt. Dies erfolgt anhand der entsprechenden Stückliste (DVD/Stücklisten/Stückliste_Testgerät_Platine.xls). Alle weiteren Bauteile bzw. Elemente, die ebenfalls zum Aufbau des Testgerätes benötigt werden, sind in einer weiteren Stückliste zu finden (DVD/Stücklisten/Stückliste_Testgerät_Sonstiges.xls). Zu sämtlichen hier beschriebenen Arbeitsschritten existieren Bilder, die in DVD/Mechanik/Testgerät - Bilder.pdf zu finden sind. Da sich das Platinenlayout im Laufe dieser Arbeit geändert hat, entsprechen diese Bilder nicht exakt dem hier beschriebenen Gerät.

Bei dem verwendeten Kunststoffgehäuse müssen die vorderen drei Befestigungsbolzen an der Bodenplatte im Inneren des Gehäuses mit geeignetem Werkzeug entfernt werden. Dies kann beispielsweise mit einem Dremel erfolgen und ist deshalb notwendig, um etwa 2 mm zusätzlichen Platz für den Einbau der gesamten Elektronik zu gewinnen. Die Entfernung der restlichen Bolzen ist ebenfalls möglich, jedoch nicht notwendig. Für die Schnittstellen des Gerätes werden Öffnungen an der Gehäuserückseite benötigt. Dies erfolgt in der Regel mit einer CNC-Fräse. Alle hierfür benötigten Dateien befinden sich unter DVD/Mechanik/Gehäuse_Ausfräsungen/*. Die entsprechende Vektorgrafik (*.dxf) für die Ausfräsungen wurde mit Hilfe des Programms *Corel Designer X6*[3] erstellt. Die dazuge-

[3] http://www.corel.com/de/ (zuletzt aufgerufen am 25.08.2015).

hörige Quelldatei (*.des) befindet sich ebenfalls auf der beiliegenden DVD. Mit Hilfe des Programms *SheetCam TNG*[4] (*.job) wurde der entsprechende G-Code erzeugt (*.tap), der zu allen gängigen Fräsprogrammen kompatibel sein sollte. Zum Befestigen der Platine im Gehäuse wird aus einer Aluminiumplatte eine Platinen-Halterung hergestellt. Hierzu wird die Platte mit einer CNC-Fräse entsprechend bearbeitet (DVD/Mechanik/Platinen-Halterung/*). Anschließend müssen Gewinde in die ausgefrästen Befestigungslöcher mit Hilfe eines M3-Gewindeschneiders geschnitten und die Halterung entgratet werden. Bevor die Platine in das Gehäuse eingebaut wird, müssen der AN/AUS-Schalter sowie die beiden Bananenbuchsen in das Gehäuse eingebaut und mit ausreichend langen Leitungen verlötet werden. Anschließend wird die Platinen-Halterung in das Gehäuse eingesetzt und auf die passende Position gebracht. An die Stellen der Löcher werden anschließend 5 mm-Distanzhülsen mit etwas Kleber fixiert (siehe Abbildung 2.7). Danach müssen die vier

Abbildung 2.7: Aufbau von Testgerät (Gehäuse mit Platinen-Halterung).

Kunststoffschrauben zum Befestigen des Raspberry Pis mit Hilfe von jeweils zwei Kunststoffmuttern an die Platine geschraubt und der Kühlkörper - sofern noch nicht geschehen - mit Wärmeleitpaste und einer M3-Schraube an *IC14* befestigt werden. Nun kann die bestückte Leiterplatte mittels Kunststoffschrauben auf der Platinenhalterung fixiert werden. Da eine der vier USB-Schnittstellen des Raspberry Pis mit der Platine verbunden und nicht

[4] http://www.sheetcam.com/ (zuletzt aufgerufen am 25.08.2015).

nach außen geführt werden soll, muss die äußere USB-Buchse ausgelötet und anschließend müssen zwei Leitungen an *D+* und *D-* angelötet werden. Außerdem ist es notwendig, die vier Befestigungslöcher des Raspberry Pis auf 3 mm Durchmesser aufzubohren. Anschließend wird dieser mit vier Kunststoffmuttern an die Platine montiert. Nun werden die auf die passende Länge gekürzten, vorhandenen Leitungen mit darauf gequetschten Aderendhülsen an die entsprechenden Klemmen angeschlossen. Der Raspberry Pi wird durch eine 40-polige Flachbandleitung und ein USB-Kabel mit der Platine verbunden. Bei dem in der Stückliste angegebenen USB-Kabel ist es notwendig, auf beiden Seiten jeweils das Gehäuse zu entfernen. Dies ist problemlos mit einem Schraubendreher oder einem anderen flachen Gegenstand möglich, da es sich hierbei lediglich um zwei aneinander gesteckte Gehäusehälften handelt. Es ist notwendig, das Gehäuse des USB-A-Steckers zu entfernen, da andernfalls der Gehäusedeckel nicht befestigt werden kann.

Abbildung 2.8: Aufbau von Testgerät (Gehäuse mit eingebauter Elektronik).

An dem Deckel des Gehäuses sollen sowohl ein 4-Zeilen-Display, als auch vier Taster befestigt werden. Die hierfür notwendigen Öffnungen werden wiederum mit der CNC-Fräse ausgefräst (DVD/Mechanik/Gehäusedeckel/*). Die vier Befestigungslöcher für das Display müssen anschließend noch mit einem Senker soweit bearbeitet werden, bis die dazu gehörigen Senkkopfschrauben bündig mit der Oberfläche der Aluminiumplatte abschließen. Danach werden die Schrauben jeweils mit Hilfe einer 5 mm-Distanzhülse und zwei Muttern an dem Gehäusedeckel befestigt. Die Unebenheiten auf der Oberfläche des Deckels, die

sich aufgrund der vier Schrauben ergeben, können bei Bedarf noch mit Epoxidharz ausgeglichen werden. Der Gehäusedeckel wird mit einem Aufkleber versehen. Dieser wird auf - idealerweise selbstklebendem - Fotopapier ausgedruckt und nach dem Zuschneiden auf die bearbeitete Aluminiumplatte mitsamt des Display-Frontrahmens aufgeklebt. Anschließend wird eine 16-polige Flachbandleitung an das Display gelötet und dieses mit insgesamt vier Muttern an den Gehäusedeckel geschraubt. Zuvor müssen die vier Befestigungslöcher des Displays auf einen Durchmesser von 3 mm aufgebohrt werden. Das andere Ende der Flachbandleitung wird durch eine Pfostenbuchse mit dem vorgesehenen Wannenstecker verbunden. An die vier Taster werden ausreichend lange Leitungen angelötet, die auf der anderen Seite mit Aderendhülsen versehen und an den entsprechenden Klemmen befestigt werden. Die Kontakte der Taster sollten mit Schrumpfschlauch isoliert und anschließend etwas zur Seite gebogen werden, damit die erforderliche Einbauhöhe minimiert wird. Als letzter relevanter Arbeitsschritt wird eine ca. 100 mm x 60 mm große Kunststofffolie auf die Displayrückseite geklebt. Dies ist deshalb notwendig, da die Einbauhöhe des Displays nahezu identisch mit der noch verfügbaren Höhe ist und mit der Folie gewährleistet wird, dass kein versehentlicher Kontakt zwischen dem Display und dem Raspberry Pi entsteht. Der Gehäusedeckel kann nun theoretisch auf das Gehäuse geschraubt werden, jedoch muss dieses zur ersten Inbetriebnahme noch geöffnet sein.

Abbildung 2.9: Fertiges Testgerät (Vorderansicht).

Sämtliche Elemente, die zum Betrieb des Testgerätes notwendig, jedoch nicht fest mit

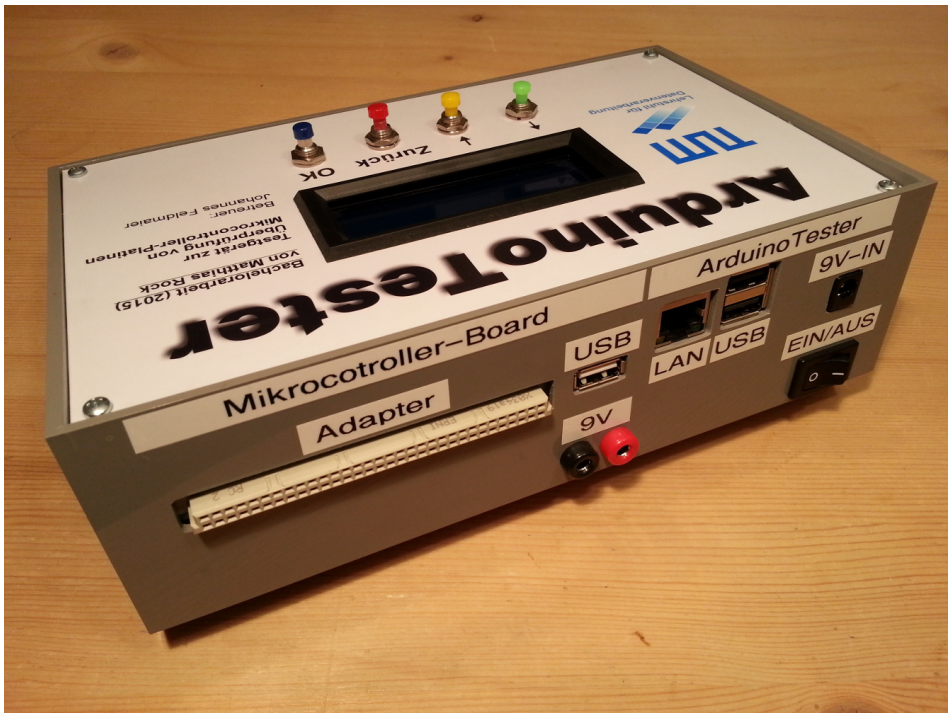


Abbildung 2.10: Fertiges Testgerät (Rückseite).

diesem verbunden sind, können einer separaten Stückliste entnommen werden (DVD/-Stücklisten/Stückliste_Zubehör.xls). Bezüglich der Montage müssen hierbei lediglich die zwei Bananenstecker mit dem Hohlstecker durch zwei Leitungen verbunden werden.

2.5.1 Adapterplatine für Arduino Leonardo

Für den Aufbau des Adapters muss die entwickelte und fertig hergestellte Platine mit allen Bauteilen bestückt werden. Hierfür wird die Stückliste DVD/Stücklisten/Stückliste_Adapter_Leonardo.xls verwendet. Aufgrund der Tatsache, dass das Entfernen eines auf die Adapterplatine aufgesteckten Leonardo-Boards nur schwer möglich ist, ohne die Pins zu verbiegen, wurde ein Hilfsmittel konstruiert, mit dem dies sehr einfach zu bewerkstelligen ist. Dieser *Board-Entferner* wurde mit der CNC-Fräse aus einer 10 mm dicken Aluminiumplatte ausgefräst. Die entsprechenden Dateien sind unter DVD/Mechanik/Leonardo_Board-Entferner/* zu finden. Die Stückliste findet sich unter DVD/Stücklisten/Stückliste_Adapter_Board-Entferner.xls. Es existieren Bilder vom Aufbau des Adapters (DVD/Mechanik/Adapter Leonardo - Bilder.pdf), diese unterscheiden sich allerdings in geringem Maße von dem in dieser Arbeit beschriebenen Platinenlayout.

Abbildung 2.11: Adapterplatine für Arduino Leonardo.

Abbildung 2.12: Hilfsmittel zum einfachen Entfernen des Leonardos von der Adapterplatine.

2.5.2 Hinweis

In dem konkret im Rahmen dieser Arbeit gebauten Gerät ist eine Platine verbaut, dessen Layout von dem hier beschriebenen abweicht. Schaltungstechnisch entspricht es nach diversen Umbaumaßnahmen exakt dem hier vorgestellten Gerät. Damit die Änderungen problemlos nachvollzogen werden können, befinden sich sowohl der ursprüngliche Schaltplan als auch das Platinenlayout ebenfalls auf der beigelegten DVD (DVD/Schaltpläne/Testgerät - Schaltplan_ALT.pdf, DVD/Platinen/Testgerät/Testgerät_ALT/*). Alle Änderungen bzw. behobenen Fehler können in DVD/Platinen/Testgerät/Testgerät_ALT/Platine - Änderungen.pdf nachvollzogen werden.

Selbiges gilt für die Adapterplatine. Der ursprüngliche Schaltplan befindet sich unter DVD/Schaltpläne/Adapterplatine_Leonardo - Schaltplan_ALT.pdf und das Platinenlayout unter DVD/Platinen/Adapterplatine_Leonardo/Adapterplatine_Leonardo_ALT/*.

2.6 Software

Bezüglich der Software muss ein passendes Betriebssystem für den Raspberry Pi 2 ausgewählt und anschließend konfiguriert werden. Außerdem ist es notwendig, eine entsprechende Testsoftware für den Raspberry Pi, den ATmega8 und für das Arduino Leonardo-Board zu programmieren.

2.6.1 Wahl des Betriebssystems für Raspberry Pi 2

Da es eine Vielzahl an Betriebssystemen für den Raspberry Pi gibt, stellt sich die Frage, welches davon am geeignetsten für das geplante Vorhaben ist. Ein sehr entscheidendes Kriterium ist die Geschwindigkeit und insbesondere die benötigte Bootzeit, die zum Zwecke der Anwenderfreundlichkeit relativ kurz sein sollte. Es ist somit ein möglichst schlankes Betriebssystem sinnvoll, das nur die nötigsten Programme enthält. Insbesondere wird

auch keine graphische Benutzeroberfläche (GUI) benötigt. Nahezu alle Betriebssysteme für den Raspberry Pi sind Linux-Distributionen [10] und das am häufigsten verwendete Betriebssystem ist Raspbian[5], eine angepasste Version von Debian[6]-Linux [3]. Es eignet sich besonders für Anfänger, da es bereits zahlreiche vorinstallierte Softwarepakete enthält, welche für diesen Anwendungsfall jedoch nicht benötigt werden. Aufgrund der weiten Verbreitung von Raspbian und der damit verbundenen hohen Anzahl an verfügbarer Software macht es Sinn, sich für ein Derivat dieser Distribution zu entscheiden. Mögliche Kandidaten ohne graphische Benutzeroberfläche sind DietPi[7], MINIBIAN[8] und pipaOS[9] [10], welche zudem auch explizit den Raspberry Pi 2 unterstützen. Ein im Zusammenhang mit einer kurzen Bootzeit häufig genanntes Betriebssystem ist zudem Arch Linux ARM[10] [3], das jedoch kein Raspbian-Derivat darstellt, allerdings aufgrund seiner verhältnismäßig großen Verbreitung ebenfalls als möglicher Kandidat in Frage kommt.

Da zu den genauen Bootzeiten der einzelnen Betriebssysteme keine zuverlässigen Recherchequellen existieren, insbesondere auch nicht für den neuen Raspberry Pi 2, wurden die Bootzeiten der vier genannten Betriebssysteme im Rahmen dieser Arbeit gemessen. Diese sind Tabelle 2.1 zu entnehmen. DietPi scheidet nicht nur aufgrund des langen La-

Betriebssystem	**Bootzeit in Sekunden**
DietPi (v53)	22
Raspbian (Wheezy; GUI deaktiviert)	21
MINIBIAN (2015-02-18)	20
pipaOS (3.3 Debian Wheezy PI2)	12
Arch Linux ARM (01-Apr-2015)	12

Tabelle 2.1: Gemessene Bootzeiten aller betrachteten Betriebssysteme mit Rapsberry Pi 2.

devorgangs aus, sondern auch deshalb, weil das System nach dem Unterbrechen der Stromversorgung während des Betriebes nicht mehr ordnungsgemäß startete. Aufgrund der Tatsache, dass Arch Linux ARM nicht schneller bootet als das Raspbian-Derivat pipaOS, wurde letzteres als Betriebssystem für das Testgerät ausgewählt.

PipaOS

PipaOS ist eine auf Raspbian basierende Linux-Distribution für den Raspberry Pi, die deutlich weniger Speicherplatz benötigt als Raspbian und auch auf dem Raspberry Pi 2 lauffähig ist. Das Betriebssystem ist robust gegenüber Stromausfällen und kann unkompliziert via SSH konfiguriert werden [11].

[5] https://www.raspbian.org (zuletzt aufgerufen am 25.08.2015).
[6] https://www.debian.org (zuletzt aufgerufen am 25.08.2015).
[7] http://fuzon.co.uk/phpbb/viewtopic.php?f=8&t=6 (zuletzt aufgerufen am 25.08.2015).
[8] https://minibianpi.wordpress.com/ (zuletzt aufgerufen am 25.08.2015).
[9] http://pipaos.mitako.eu/ (zuletzt aufgerufen am 25.08.2015).
[10] http://archlinuxarm.org (zuletzt aufgerufen am 25.08.2015).

2.6.2 Installation und Konfiguration von pipaOS

Im Folgenden werden die notwendigen Schritte beschrieben, um ein funktionsfähiges und den Anforderungen entsprechendes Betriebssystem für den Raspberry Pi 2 zu erhalten. Sowohl die Installation als auch die Konfiguration erfolgen in dieser Arbeit über einen Computer mit Windows bzw. einem Linux-basierten Betriebssystem. Bei Verwendung eines anderen Betriebssystems muss ggf. anders vorgegangen werden.

Installation

Zur Installation des Betriebssystems wird ein Kartenlesegerät und ggf. ein SD-mircoSD-Adapter benötigt.

Zunächst wird das gepackte Image von pipaOS heruntergeladen[11] und anschließend entpackt. In diesem Fall handelt es sich um die Version *pipaOS 3.3 Debian Wheezy PI2 2GB*. Diese Datei befindet sich ebenfalls auf der beigelegten DVD (DVD/Software/RaspberryPi/PipaOS-3.3-Wheezy.img.gz). Anschließend muss die microSD-Karte über das Kartenlesegerät mit dem Computer verbunden werden und die entpackte Image-Datei mit einem geeigneten Programm auf die Karte aufgespielt werden. Unter Windows kann dies beispielsweise mit dem Programm *Win32 Disk Imager*[12] erfolgen. Bei einem Linux-basierten Betriebssystem kann wie folgt vorgegangen werden:

```
1 gunzip Datei.img.gz
2 cat /proc/partitions
3 umount /dev/sd...
4 sudo dd if=Dateipfad of=PfadLaufwerk bs=1M
```

Nach dem Entpacken (Zeile 1) werden die Laufwerksbezeichnung sowie die evtl. vorhandenen Partitionen der Speicherkarte festgestellt (Zeile 2). Anschließend werden alle darauf befindlichen Partitionen ausgehängt (Zeile 3) und die Imagedaten auf die SD-Karte kopiert (Zeile 4). Sobald der Vorgang abgeschlossen ist, kann die microSD-Karte aus dem Kartenlesegerät entnommen und in den Slot des Raspberry Pis gesteckt werden.

Konfiguration

Da bei dem Betriebssystem pipaOS standardmäßig SSH installiert und aktiviert ist, wird zur ersten Konfiguration keine Tastatur und kein direkt angeschlossener Monitor benötigt. Für die Einrichtung des Betriebssystems wurde das fertig montierte Testgerät an das Netzteil und per Ethernet an einen Router angeschlossen. Die Konfiguration erfolgte an einem ebenfalls an diesen Router angeschlossenen PC per SSH. Der Standard-Benutzername ist **sysop** und das dazugehörige Passwort lautet **posys** [11]. Ein Ziel der Konfiguration

[11] http://pipaos.mitako.eu/ (zuletzt aufgerufen am 25.08.2015).
[12] http://sourceforge.net/projects/win32diskimager/ (zuletzt aufgerufen am 25.08.2015).

ist es, einen Dateiaustausch via FTP zu ermöglichen sowie die Einrichtung des Raspberry Pis als Fileserver. Zur Kommunikation ist die Aktivierung und Konfiguration der SPI- und I^2C-Schnittstellen notwendig. Installiert werden müssen eine Software zum Flashen von Mikrocontrollern via SPI, ein Compiler für die in C++ 11 geschriebene Testsoftware, ein Programm, das angeschlossene USB-Sticks automatisch mountet sowie die Software WiringPi[13], die einen einfachen Zugriff auf GPIO-Pins vom entwickelten Programm aus ermöglicht. Unter Windows soll das Testgerät zudem über den Hostnamen angesprochen werden können. Weiterhin muss dafür gesorgt werden, dass das entwickelte Testprogramm nach jedem Bootvorgang automatisch gestartet wird.

Alle notwendigen Konfigurationsschritte können im Anhang in Kapitel 5.4 nachverfolgt werden. Das Betriebssystem mit allen durchgeführten Installationen und Konfigurationen befindet sich ebenfalls auf der beigelegten DVD als 7-Zip-gepackte Image-Datei (DVD/-Software/RaspberryPi/ArduinoTester.7z).

2.6.3 Testverfahren / Testablauf

Auf Grundlage der in Kapitel 2.2 erläuterten Messgrößen und -verfahren, der Kenntnis des Aufbaus des zu testenden Boards und der durch die Hardware unterstützten Messmöglichkeiten wurde ein Testablauf entwickelt, der alle relevanten Messungen durchführt und eine zuverlässige Entscheidungsfindung über die Funktionsfähigkeit des Mikrocontroller-Boards ermöglicht. Im Nachfolgenden werden alle durchzuführenden Tests in der entsprechenden Reihenfolge erläutert.

Spannungsversorgungen

Zum Testen der Spannungsversorgungen werden nacheinander alle relevanten Versorgungsspannungs-Relais geschaltet und jeweils nach einer gewissen Verzögerung die Spannung und der Strom, der durch das Board fließt, gemessen. Diese Verzögerung ist notwendig, da die auf dem Board vorhandenen Kapazitäten im Einschaltmoment einen Kurzschluss darstellen und sich somit erst nach dem Ladevorgang ein konstanter Strom einstellt. Mit $1,2\,s$ ist die Verzögerungszeit im Rahmen dieser Arbeit auf einen relativ hohen Wert festgelegt worden. Durch das Optimieren dieses Wertes kann die gesamte Testzeit reduziert werden. Weiterhin wird festgestellt, ob die Strombegrenzung aktiv ist. Wenn der gemessene Strom bei einer Spannungsversorgung zu hoch oder zu niedrig ist, wird diese als defekt erkannt.

GND-Pins

Ziel dieses Tests ist die Überprüfung, ob alle GND-Pins des zu testenden Boards miteinander verbunden sind. Dies wird erreicht, indem eine Spannungsquelle an das Board geschaltet wird und die Spannungen an allen anderen GND-Pins ermittelt werden. Sofern

[13] http://wiringpi.com (zuletzt aufgerufen am 25.08.2015).

eine gewisse Schwelle nicht überschritten ist, wird der Pin als korrekt erkannt. Sollte ein Pin keinen Kontakt zum Haupt-GND-Pin haben, über den die Stromversorgung erfolgt, so wird dieser durch den eingebauten Pullup-Widerstand auf High-Pegel gebracht und dadurch als defekt angesehen.

3,3 V-, 5 V-Pins und IOREF

Bei allen Ausgangsspannungs-Pins (3,3 V-, 5 V-Pins und IOREF) wird jeweils die Spannung gemessen. Sofern diese innerhalb einer definierten Toleranz um den Sollwert liegt, wird der Pin als voll funktionsfähig erkannt.

Durchgangsprüfung

Hier wird festgestellt, ob ein Datenpin mit dem Testgerät verbunden ist. Dazu wird die Konstantstromquelle an die einzelnen Datenpins geschaltet und jeweils die anliegende Spannung gemessen. Wenn eine definierte Schwelle unterschritten wird, so folgt daraus eine korrekte Verbindung zum Pin (siehe Kapitel 2.2.1). Der Wert dieser Schwelle muss sich zwischen der Schwellspannung der ESD-Schutzdioden und der Versorgungsspannung der Konstantstromquelle (= 5 V) befinden. Aufgrund gewisser Toleranzen insbesondere bei den Schwellspannungen der Dioden ist auf einen ausreichenden Sicherheitsabstand zu den Rändern des angegebenen Spannungsbereiches zu achten. In dieser Arbeit ist der Wert auf 1,4 V festgelegt worden.

Digitale Ausgänge / Kurzschlussprüfung

Bei diesem Prüfverfahren werden nicht nur die Pegel der digitalen Ausgänge geprüft, sondern es wird zusätzlich noch festgestellt, ob zwischen zwei oder mehreren Pins womöglich ein Kurzschluss besteht. Prinzipiell läuft der Test wie folgt ab: ein Testmuster, das die Sollzustände aller digitalen Ausgangspins repräsentiert, wird an das Mikrocontroller-Board gesendet. Dieses schaltet die Pins entsprechend der vorgegebenen Pegel. Danach werden diese Pegel durch den ATmega8 vom Raspberry Pi gemessen und mit den Sollwerten verglichen. Bei einer Diskrepanz wird der entsprechende Pin als fehlerhaft detektiert. Für die Generierung der Testmuster wird das Kurzschluss-Testverfahren verwendet, das im nachfolgenden Abschnitt genauer erläutert wird.

Kurzschluss-Testverfahren

Wie bereits zuvor erwähnt, ist es im Rahmen des Testverfahrens der digitalen Ausgänge nicht ausreichend, allein zu prüfen, ob ein vorgegebener Pegel an einem I/O-Pin auch tatsächlich anliegt. Es muss zudem noch getestet werden, ob alle Pins voneinander unabhängig sind, d. h. dass kein versehentlicher Kurzschluss zwischen diesen besteht (z. B. durch eine Zinnbrücke auf der Leiterplatte verursacht). Um sicherzustellen, dass ein digitaler

Ausgangspin voll funktionsfähig ist, muss dieser sowohl auf High- als auch auf Low-Pegel getestet werden. Dies stellt sicher, dass ein Pin weder versehentlich mit GND, noch mit VCC fest verbunden ist. Ein naheliegendes Testverfahren zur gemeinsamen Überprüfung

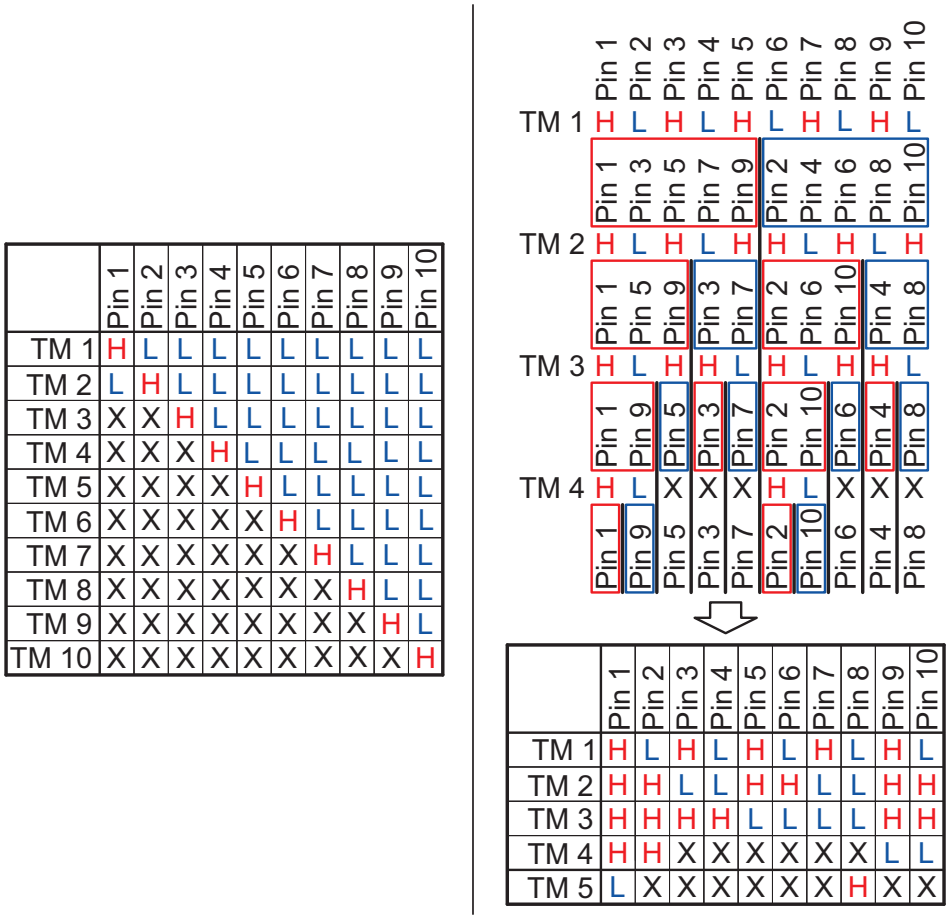


Abbildung 2.13: Ineffizientes (links) und entwickeltes effizientes (rechts) Testverfahren zur Überprüfung von Kurzschlüssen zwischen Pins.

der Pegel mit womöglich vorhandenen Kurzschlüssen ist in Abbildung 2.13 (links) dargestellt. Dem Mikrocontroller wird zunächst das erste Testmuster (TM 1) mitgeteilt, welches anschließend überprüft wird. Pin 1 wird hierbei auf High-Pegel gesetzt und alle restlichen Pins auf Low. Wenn alle Pegel korrekt gemessen werden, kann daraus geschlussfolgert werden, dass Pin 1 mit keinem anderen Pin verbunden ist. Anschließend wird so jeder einzelne Pin alleine auf High gesetzt und die entsprechende Messung durchgeführt. Die Pins links von dem auf High gesetzten Pin müssen hierbei nicht mehr gemessen werden, da hier bereits festgestellt worden ist, dass keine Verbindung zu diesem Pin besteht. Lediglich bei Testmuster 2 (TM 2) muss der Low-Pegel von Pin 1 ermittelt werden, da jeder

Pin mindestens einmal sowohl auf High- als auch auf Low-Pegel getestet werden muss. In Abhängigkeit von der Anzahl der Pins N_{Pins} ($N_{Pins} > 1$) ergibt sich somit für die Anzahl der notwendigen Testmuster (N_{TM})

$$N_{TM} = N_{Pins}$$

und für die Anzahl der notwendigen Messungen (N_{Mess}) nach der gaußschen Summenformel

$$N_{Mess} = \frac{N_{Pins} \cdot (N_{Pins} + 1)}{2} + 1.$$

Da hiermit für die Überprüfung von 20 voneinander unabhängigen Pins insgesamt 20 Testmuster und 211 Messungen notwendig wären, wurde im Rahmen dieser Arbeit ein effizienteres Verfahren entwickelt. Dieses ist ebenfalls in Abbildung 2.13 auf der vorherigen Seite (rechts) zu sehen. Bei dieser Testmethode werden im ersten Durchgang abwechselnd High- und Low-Pegel an die Pins angelegt, wobei mit High begonnen wurde. Durch die Überprüfung von Testmuster 1 (TM 1) kann geschlussfolgert werden, dass alle auf High-Pegel geprüften Pins unabhängig von den Low-Pins sind. In der Abbildung ist dies durch Umsortierung der Pins und mit einer vertikalen Linie dazwischen dargestellt. In beiden voneinander unabhängigen Blöcken wird ebenfalls wieder ein Testmuster angelegt, das mit High beginnt und danach stets den Pegel wechselt. Dieses Verfahren wird so lange weitergeführt, bis alle Pins voneinander getrennt sind. Wenn die so entstandenen Testmuster wieder in eine Tabelle nach den Pins aufsteigend sortiert eingetragen werden, ist das Schema, wonach die Testmuster erstellt werden müssen, leicht erkennbar. Es ist hierbei immer noch ein weiteres Testmuster nötig, da Pin 1 auf Low und ein weiterer Pin auf High getestet werden müssen. Anhand dieser Tabelle ist direkt erkennbar, dass für die Anzahl der erforderlichen Testmuster

$$N_{TM} = \lceil ld(N_{Pins}) \rceil + 1$$

gilt. Die Anzahl der Messungen errechnet sich stets aus dem $ld(.)$ der größten Zweierpotenz, die kleiner als die Anzahl der Pins ist, multipliziert mit der Pin-Anzahl plus einem bestimmten Rest. Dieser Rest wird bestimmt, indem die Pin-Anzahl, die diese Zweierpotenz übersteigt, mit der Zahl 2 multipliziert wird plus den zwei zusätzlichen Messungen für die noch nicht gemessenen High- bzw. Low-Pegel. Zusammengefasst errechnet sich die Menge der erforderlichen Messungen somit zu

$$N_{Mess} = \left(\lceil ld(N_{Pins}) \rceil - 1\right) \cdot N_{Pins} + 2 \cdot \left(N_{Pins} - 2^{\lceil ld(N_{Pins}) \rceil - 1}\right) + 2.$$

bzw.

$$N_{Mess} = (N_{TM} - 2) \cdot N_{Pins} + 2 \cdot \left(N_{Pins} - 2^{N_{TM}-2}\right) + 2$$

Ein Vergleich der benötigten Anzahl an Testmustern und Messungen zwischen den beiden hier vorgestellten Verfahren findet sich in Tabelle 2.2.

		Ineffizientes Testverfahren			Effizientes Testverfahren	
Pinanzahl		**#Testmuster**	**#Messungen**		**#Testmuster**	**#Messungen**
2		2	4		2	4
5		5	16		4	14
10		10	56		5	36
15		15	121		5	61
20		20	211		6	90
30		30	466		6	150
50		50	1276		7	288

Tabelle 2.2: Vergleich der Anzahl der notwendigen Testmuster und Messungen zwischen ineffizientem und effizientem Kurzschluss-Testverfahren.

Digitale Eingänge

Beim Test der digitalen Eingänge wird zu Beginn dem Mikrocontroller-Board mitgeteilt, dass alle dieser Pins als Eingang geschaltet werden sollen und der Digital-Analog-Umsetzer wird auf einen hohen Low-Pegel eingestellt. Diese Spannung ergibt sich aus dem im Datenblatt des zu testenden Mikrocontrollers angegebenen Wert für die maximale Low-Spannung. Von dieser wird als Sicherheitsabstand noch ein kleiner Wert subtrahiert. Nach der Auswahl des jeweiligen Pins wird nicht nur der detektierte Pegel vom Mikrocontroller-Board abgerufen, sondern auch der Messwert des ATmega8. Dies hat den Hintergrund, dass ein defekter Eingang beispielsweise als Ausgang fungieren und somit eine Änderung des vorgegebenen Pegels erzwingen könnte. Wenn der vom Mikrocontroller-Board empfangene Pegel nicht mit dem vorgegebenen übereinstimmt oder falls der Messwert des ATmega8 zu weit vom Sollwert abweicht, so wird dieser Pin als defekt erkannt. In gleicher Weise werden die Pins anschließend mit einem niedrigen High-Pegel getestet.

Analoge Eingänge

Das Prüfen der analogen Eingänge erfolgt in ähnlicher Weise wie der Test der digitalen Eingänge, allerdings wird durch das Mikrocontroller-Board kein Pegel, sondern ein Spannungswert ermittelt. Wenn dieser zu weit von dem vorgegebenen abweicht, so wird dieser Pin als defekt erkannt. Im Rahmen dieser Arbeit wird lediglich ein einziger vorgegebener Spannungswert getestet. Dies kann bei Bedarf auf beliebig viele Werte erweitert werden.

I^2C-Schnittstelle

Die I^2C-Schnittstelle wird überprüft, indem ein Testbyte an das Mikrocontroller-Board gesendet und anschließend wieder empfangen wird. Sofern das empfangene mit dem gesendeten Byte übereinstimmt, wird diese Schnittstelle als voll funktionsfähig erkannt.

2.6.4 Software für Raspberry Pi 2

Die Software des Raspberry Pis ist für die zentrale Steuerung des Testgerätes zuständig. Hiermit werden sämtliche Relais, die Multiplexer, der Digital-Analog-Umsetzer, der ATmega8 und das zu testende Mikrocontroller-Board direkt oder indirekt angesteuert. Alle nachfolgend erläuterten Quelldateien befinden sich unter DVD/Software/RaspberryPi/Programm/Source/*. Im Laufe dieses Kapitels werden der Programmaufbau und -ablauf sowie die wichtigsten Funktionen der Software erläutert.

Programmaufbau / Programmablauf

Der grundsätzliche Aufbau des in C++ 11 geschriebenen Testprogramms ist in Abbildung 2.14 dargestellt. Es existieren vier Klassen, mit denen auf unterschiedliche Hardwarekomponenten zugegriffen werden kann bzw. die spezielle Funktionen enthalten. Eine weitere Klasse erfüllt den Zweck der Fehlerbehandlung. In den nachfolgenden Unterkapiteln werden diese näher erläutert. Eine grobe Übersicht über den Programmablauf ist in Abbildung 2.15 dargestellt. Ein detaillierterer Programmablaufplan ist im Anhang in Kapitel 5.5 sowie auf der DVD (DVD/Software/RaspberryPi/Programmablaufplan - Arduino Leonardo.pdf) zu finden. Im Grunde genommen orientiert sich das geschriebene Programm an diesem detaillierten Ablaufplan, jedoch ist die Software nicht nur speziell auf den Arduino Leonardo ausgelegt, sondern sehr allgemeingültig implementiert, sodass auch andere Boards getestet werden können, ohne die gesamte Software ändern zu müssen.

Main.cpp

In dieser Quelldatei befindet sich einerseits die Hauptfunktion, die nach dem Einschalten des Testgerätes automatisch aufgerufen wird und andererseits wurden hier noch einige weitere Funktionen implementiert. Für jeden der vier Taster existiert jeweils eine Interrupt-Service-Routine (ISR), die aufgerufen wird, nachdem der Taster betätigt worden ist. Diese ISRs ändern im Prinzip ein oder mehrere Variablen, die für die Menüsteuerung relevant sind. Die Funktion *scanAdapter* überprüft, sobald sie aufgerufen wird, solange die vier Leitungen für die Adapterkennung, bis ein angeschlossener Adapter erkannt wird. In der *main*-Funktion werden zu Beginn drei Objekte und ein Zeiger auf ein zu einem späteren Zeitpunkt erstellten Objekt erzeugt. Die zugehörigen Klassen sind in separaten Quelldateien implementiert und werden in den nachfolgenden Kapiteln erläutert. Nach dem Erzeugen der Objekte wird eine Endlosschleife gestartet, in der, je nach Zustand der Menü-Variablen, das jeweilige Menü mit dem dazugehörigen Programmablauf gestartet wird. Es existiert ein innerer und ein äußerer *try-catch*-Block, die zur Fehlerbehandlung dienen. Der innere Block fängt sämtliche darin geworfenen Exceptions ab, die lediglich den normalen Programmfluss stören, jedoch nicht zur Fehlfunktion des Gerätes führen. Alle kritischen Fehler werden an den äußeren Block weitergereicht, der auch für die Fehlerbehandlung im Falle von Problemen bei der Initialisierung zuständig ist. Alle vom äußeren Block abge-

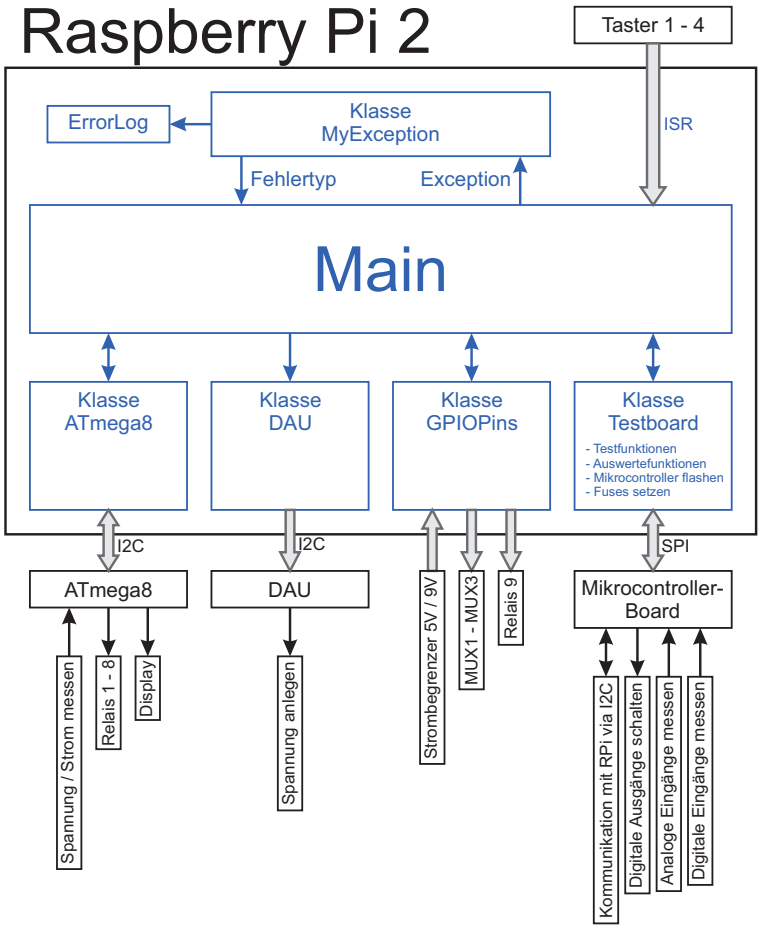


Abbildung 2.14: Aufbau der Software des Raspberry Pis bzw. Aufgaben der einzelnen Klassen.

fangenen Exceptions werden in einer Log-Datei (/home/Testprogramm/Data/ProgramData/ErrorLog.txt) protokolliert und beenden das Programm. Wenn dieser Fall eintritt, reagiert das Testgerät auf keinerlei Eingaben mehr. Sofern der Fehler durch ein Kommunikationsproblem mit dem ATmega8 auftritt, ist es nicht möglich, dies auf dem Display anzeigen zu lassen. Bei den anderen geworfenen Exceptions kann die Anzeige des Fehlerfalls auf dem Display durch eine kleine Programmerweiterung implementiert werden.

Der Ablauf des Testes im entsprechenden Menü erfolgt wie bereits erwähnt prinzipiell nach dem erstellten Programmablaufplan. Während des Testes kann dieser nur an definierten Stellen abgebrochen werden, d. h. wenn der *Zurück*-Taster betätigt wird, kann es unter Umständen noch eine kurze Zeit dauern, bis tatsächlich eine Beendigung des Programms erfolgt.

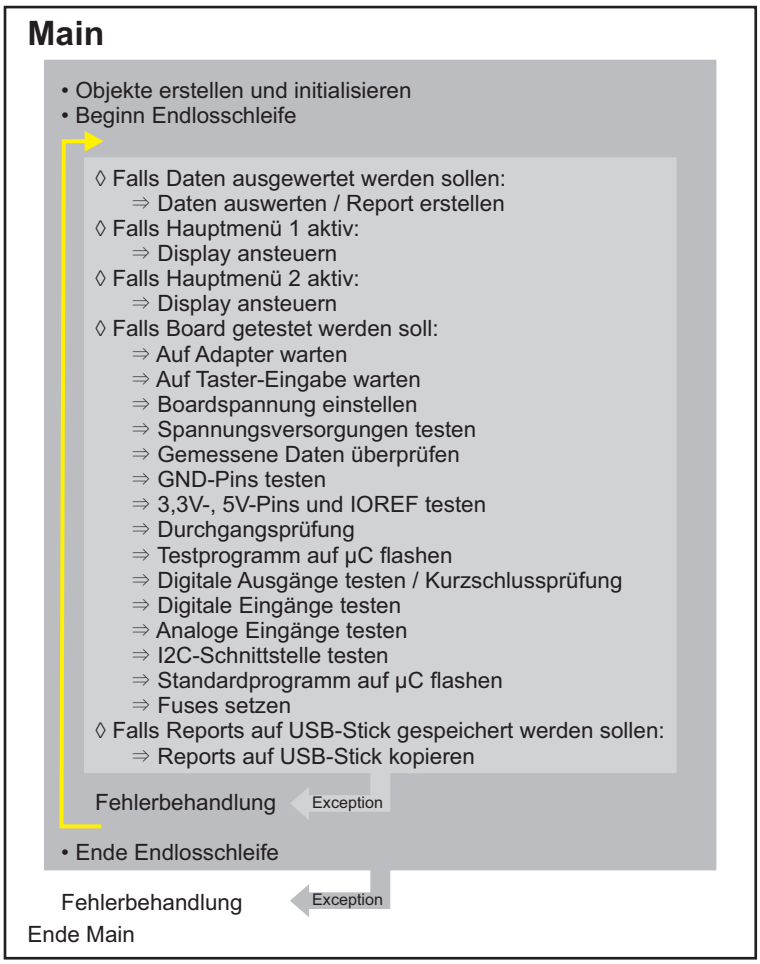


Abbildung 2.15: Programmablauf von Testsoftware für Raspberry Pi 2.

Definitions.hpp

In dieser Datei befinden sich grundlegende Definitionen, die sich in drei Gruppen einteilen lassen: Zum einen Definitionen, die nicht geändert werden sollten, sofern keine Umbaumaßnahmen an der Platine vorgenommen wurden oder größere Softwareänderungen stattfinden. Zum anderen gibt es Definitionen, die sich auf das Testverhalten beziehen und entsprechend angepasst werden können. Und weiterhin existieren Definitionen, die an jedes Gerät individuell angepasst werden müssen, um einen möglichst genauen Test zu gewährleisten. Viele davon ergeben sich aus der Hardwareverschaltung oder wurden zur Kommunikation mit anderen Teilnehmern festgelegt. Die WiringPi-Pinbelegungen ergeben sich durch Eingabe des Befehls

```
1 gpio readall
```

in die Kommandozeile des Raspberry Pis.

Klasse ATmega8

Die Klasse ATmega8 ist für sämtliche Kommunikation mit dem auf der Platine verbauten ATmega8 zuständig. Hierzu zählen die Ansteuerung des Displays, das Schalten der Relais 1 - 8 und das Messen von Spannungen oder Strömen. Mit der Funktion *switchRelais* kann ein bestimmtes an den ATmega8 angeschlossenes Relais geschaltet werden. Hierbei ist zu beachten, dass von den Relais 2 - 4, mit denen jeweils eine bestimmte Spannungsquelle für das angeschlossene Board geschaltet werden kann, stets nur eines aktiv ist. Sobald beispielsweise Relais 3 auf ON geschaltet wird, werden Relais 2 und 4 automatisch auf OFF geschaltet. Der Grund hierfür ist die Tatsache, dass eine Strommessung bei mehreren aktiven Spannungsversorgungen aufgrund des Stromsensors in der Masseleitung nicht möglich ist bzw. keinen Sinn ergibt und hierdurch ein versehentliches Aktivieren mehrerer Versorgungen beim Erstellen des Testablaufes vermieden werden kann. Mit der Funktion *measure* kann eine Spannung bzw. indirekt auch der durch das Board fließende Strom gemessen werden. Dazu wird zunächst dem ATmega8 mitgeteilt, an welchem Pin die Spannung gemessen werden soll. Nach einer kurzen Wartezeit werden dann die durch die Messung entstandenen 2 Bytes (10 Bit Auflösung) empfangen. Mit *setDisplayText* kann zeilenweise der 20 Zeichen lange Text für das Display festgelegt werden. Gesendet wird stets der gesamte Display-Inhalt mit *sendDisplayText*. Die elementaren Funktionen zum Senden und Empfangen der Bytes via I^2C sind so implementiert, dass bei einer fehlgeschlagenen Kommunikation mehrere erneute Versuche stattfinden. Wenn dies nicht gelingt, wird eine Exception geworfen, die das gesamte Programm beendet.

Klasse DAU

Mit der Klasse DAU kann dem Digital-Analog-Umsetzer mitgeteilt werden, welche Spannung an dessen Ausgang anliegen soll. Die Auflösung beträgt 10 Bit. Als Referenzspannung dient die 5 V-Versorgungsspannung.

Klasse GPIOPins

Über ein Objekt dieser Klasse können sämtliche GPIO-Pins des Raspberry Pis angesteuert bzw. abgefragt werden, mit Ausnahme der SPI- und I^2C-Schnittstellen sowie der Taster. Für jeden einzelnen der drei Multiplexer existiert jeweils eine Funktion, um diesen auf das entsprechende Signal einzustellen. Ebenfalls kann das Relais 9, das zur Einstellung der Board-Spannung dient, geschaltet werden. Die Funktion *checkShort* dient zur Überprüfung der beiden Signale der Strombegrenzer.

Klasse MyException

Im definierten Fehlerfall wird eine Exception dieser Klasse geworfen. Hiermit ist es möglich, sowohl den Fehlertypen auszulesen, als auch die Fehlermeldung zusammen mit dem Dateinamen und der Zeile, in der der Fehler aufgetreten ist, in eine Log-Datei zu schreiben.

Klasse Testboard

Durch ein Objekt dieser Klasse findet sämtliche Kommunikation mit dem zu testenden Mikrocontroller-Board statt. Die Klasse beinhaltet alle Testmethoden sowie Funktionen zur Auswertung der Testergebnisse. Im Konstruktor erfolgt die Definition der Pins des jeweiligen Mikrocontroller-Boards. Beispielsweise muss hier festgelegt werden, an welchen Pins des 64-poligen Steckers die analogen Eingänge angeschlossen sind. Die Einzelheiten ergeben sich aus dem kommentierten Quellcode. Ebenfalls werden im Konstruktor die Variablen, in denen die späteren Messwerte gespeichert werden, entsprechend der Pinbelegung des Boards initialisiert. Die Methoden *m_sendToMCB* und *m_receiveFromMCB* dienen zum Senden und Empfangen von Bytes via SPI. Da die vollduplexfähige SPI-Schnittstelle im Gegensatz zum I^2C-Bus über keinerlei Bestätigungssignal verfügt, welches dem Sender den Empfang des Bytes bestätigt, sind diese Methoden so programmiert, dass beim Übertragen eines Bytes stets danach noch ein weiteres Bestätigungs-Byte empfangen wird. Wenn der Raspberry Pi als Sender fungiert, wird im ersten Schritt das zu übertragende Byte gesendet und anschließend selbiges wieder empfangen. Sofern das empfangene mit dem gesendeten Byte übereinstimmt, kann daraus gefolgert werden, dass das Byte korrekt transferiert worden ist. Beim Empfang eines Bytes sendet der Raspberry Pi zeitgleich ein Bestätigungsbyte, das im zweiten Schritt wieder empfangen und auf Übereinstimmung mit dem ursprünglich gesendeten Byte geprüft wird.

Zur Überprüfung der gemessenen Werte existieren zwei Funktionen. Die Methode *Check_DATA_Pwr* wird direkt im Anschluss an den Test der Stromversorgungen aufgerufen, da hier bereits festgestellt werden muss, ob das Board prinzipiell funktionsfähig ist oder ob sämtliche Stromversorgungs-Anschlüsse defekt sind. Sofern mindestens eine funktionierende Spannungsversorgung existiert, wird diese zugeschaltet und der Test fortgesetzt. Die Hauptfunktion zur Auswertung ist *Check_DATA_Report*. Hier werden alle gemessenen Daten überprüft und anschließend wird ein Report erstellt, der die Testergebnisse übersichtlich darstellt.

2.6.5 Software für ATmega8

Die Software des ATmega8 ist in C geschrieben und unter DVD/Software/ATmega8/* zu finden. Die Hauptaufgabe des Programms liegt in der Ansteuerung des Displays sowie dem Schalten der Relais und dem Messen von analogen Spannungen. Gesteuert wird der ATmega8 durch den Raspberry Pi via I^2C. Hierfür ist eine geeignete Interrupt-Service-Routine implementiert worden. Für die Ansteuerung des Displays wurden bereits beste-

hende Methoden von [12] übernommen (lcd-routines.c, lcd-routines.h), die in geringem Umfang abgeändert und an das verwendete Display angepasst worden sind. Mit der Funktion *switch_relais* kann eines der acht angeschlossenen Relais in den gewünschten Zustand geschaltet werden. Zu beachten sei wiederum, dass stets nur eines der drei Spannungsversorgungs-Relais (RE2 - RE4) aktiv ist. Wenn beispielsweise Relais 4 auf ON geschaltet wird, werden die anderen beiden automatisch inaktiv. Bei Bedarf kann dies geändert werden. In der *main*-Funktion wird während der Bootzeit des Raspberry Pis das Display mit einem Ladebildschirm angesteuert, der so lange aktiv ist, bis zum ersten Mal Display-Daten vom Raspberry Pi empfangen werden. Sobald dies erfolgt ist, springt das Programm in eine Endlosschleife, in der fortlaufend ein Steuerbyte bzw. Display-Daten überprüft werden, sofern es sich um neu empfangene Daten handelt.

2.6.6 Testsoftware für Arduino Leonardo

Für den Test des Mikrocontroller-Boards ist es notwendig, ein spezielles Testprogramm auf dieses aufzuspielen, welches dazu in der Lage ist, die Messungen durchzuführen, die Pins entsprechend zu schalten und mit dem Raspberry Pi zu kommunizieren. Die Software ist ebenfalls in C programmiert und befindet sich unter DVD/Software/ArduinoLeonardo/Testprogramm/*. Da die Pins der SPI-Schnittstelle des Arduino Leonardos vollkommen unabhängig von den restlichen Datenpins sind, verläuft die gesamte Kommunikation zwischen dem Raspberry Pi und dem Leonardo via SPI. Bei anderen Mikrocontroller-Boards, bei denen ein SPI-Pin zugleich auch beispielsweise als digitaler Ausgang genutzt werden kann, ist es notwendig, dass auch eine Kommunikation über eine andere Schnittstelle - z. B. via I^2C - unterstützt wird, um auch die SPI-Pins prüfen zu können. Dieser Fall wurde im Rahmen der Arbeit nicht implementiert.

Von der Funktionsweise arbeitet die *main*-Funktion ähnlich der des ATmega8. Es wird in einer Endlosschleife überprüft, ob ein neues Steuerbyte empfangen wurde und gegebenenfalls wird dieser Befehl anschließend ausgeführt. Implementiert werden musste jeweils eine Interrupt-Service-Routine sowohl für die SPI-Schnittstelle, als auch für den I^2C-Bus. Zum Schalten und Abfragen der Pins existieren mehrere Funktionen, deren Funktionsweisen im kommentierten Quellcode leicht nachzuvollziehen sind.

2.7 Inbetriebnahme

Nach dem Zusammenbau des Gerätes und der Installation und Konfiguration des Betriebssystems müssen noch einige Arbeitsschritte erledigt werden, um die Funktionsfähigkeit des Testgerätes herzustellen. Das Gerät wird an das Netzteil sowie an ein lokales Netzwerk bei geöffnetem Gehäusedeckel angeschlossen und eingeschaltet.

Mit einem Spannungsmessgerät wird am Testpunkt *REF_2,5V* die Spannung gemessen und mit einem Schraubendreher das Potentiometer R13 so lange verstellt, bis möglichst genau 2,5 V anliegen. R22 wird an den linken Anschlag gedreht. Dies hat zur Fol-

ge, dass bei der Durchgangsprüfung der maximal einstellbare Strom fließt (ca. 0,93 mA). Das ist deshalb möglich, da die Schutzdioden von AVR-Mikrocontrollern mit maximal ca. 1 mA durchflossen werden können [13]. Sollten auch andere Mikrocontroller-Boards getestet werden, für deren Schutzdioden dieser Strom zu hoch ist, so muss ggf. der Wert verringert werden. Bei einem zu starken Verringern des Stromes wird unter Umständen nicht mehr die Höhe der Schwellspannung erreicht, sondern ein zu niedriger Wert bei der Durchgangsmessung ermittelt. Mit R53 kann in geringem Maße der gewünschte Kontrast des Displays und mit R55 dessen Helligkeit eingestellt werden.

Die Software für den ATmega8 wurde in C unter AVR-Studio 4[14] mit installiertem WinAVR[15] auf einem Windows-System erstellt. Um das Programm auf den ATmega8 zu flashen, wird ein an den Computer angeschlossener ISP-Programmer mit K4 auf der Platine verbunden. Die zu überspielende *.hex-Datei befindet sich unter DVD/-Software/ATmega8/default/Programm.hex. Nach dem Flashen muss noch sichergestellt sein, dass alle Fuses richtig gesetzt sind. Die korrekten Einstellungen sind Abbildung 5.6 im Anhang zu entnehmen. Nun wird das Testgerät auf dem verwendeten Computer als Netzlaufwerk hinzugefügt oder alternativ eine FTP-Verbindung aufgebaut. Anschließend werden alle Dateien von DVD/Software/RaspberryPi/Programm/* nach /home/Testprogramm/ kopiert. Die Datei /home/Testprogramm/Data/Microcontroller-Board_Program/TestSoftware/ArduinoLeonardo.hex entspricht der Datei DVD/Software/ArduinoLeonardo/Testprogramm/default/ArduinoLeonardo.hex, welche aus dem im vorherigen Kapitel beschriebenen Quellcode erstellt worden ist. Das Standard-Programm, das nach einem Test automatisch auf den Arduino Leonardo aufgespielt wird (home/Testprogramm/Data/MicrocontrollerBoard_Program/StandardSoftware/ArduinoLeonado.hex), stammt von der Software ARDUINO 1.6.5[16]. Nach dem Entpacken ist diese Datei unter arduino-1.6.5-r2/hardware/arduino/avr/bootloaders/caterina/Caterina-Leonardo.hex zu finden. Hierbei handelt es sich um ein Standard-Blink-Programm mit Bootloader, welcher dafür sorgt, dass das Board direkt mit der Arduino-Software via USB programmiert werden kann.

Um eine möglichst genaue Messung des Gerätes zu gewährleisten, wird die Ausgangsspannung des Schaltreglers LM2676 am Testpunkt *5 V* ermittelt und in der *Definitions.hpp* unter *AT_BOARD_VOLTAGE_IN_mV* eingetragen. Der Wert *CURRENT_CHANGE_FACTOR* ist ein Korrekturfaktor für die Strommessung, der aufgrund der Leitungswiderstände und der Toleranzen der an der Strommessung beteiligten Bauteile ermittelt werden muss. Dazu wird dieser zunächst auf den Wert 100 gesetzt. Nach dem Abspeichern wird das Programm erstellt und ausgeführt (via SSH):

```
1 cd /home/Testprogramm
2 make
3 sudo ./program
```

[14] http://www.mikrocontroller.net/articles/Atmel_Studio (zuletzt aufgerufen am 25.08.2015).
[15] http://sourceforge.net/projects/winavr/files/WinAVR/ (zuletzt aufgerufen am 25.08.2015).
[16] https://www.arduino.cc/en/Main/Software (zuletzt aufgerufen am 25.08.2015).

Nun wird der Adapter ohne aufgestecktes Board an das Testgerät angeschlossen. An die 9 V-Bananenbuchse wird ein Strommessgerät in Reihe zu einem Widerstand (ca. 100 Ω) geschaltet und ein Test durchgeführt. Der gemessene Strom dividiert durch den im Report angezeigten Wert ergibt den Korrekturfaktor, der in Prozent für *CURRENT_CHANGE_FACTOR* eingetragen werden muss. Nach dem Beenden des Programms muss dieses nochmals neu kompiliert und gestartet werden:

```
1 make
2 sudo ./program
```

Nach diesen Schritten ist das Testgerät voll funktionsfähig. Der Autostart des Testprogramms muss an dieser Stelle nicht mehr eingerichtet werden, da dies bereits in Kapitel 2.6.2 erfolgt ist.

2.8 Funktionstest

Zur Feststellung der Funktionsfähigkeit des Testgerätes wurden Tests durchgeführt. Hierfür diente ein Arduino Leonardo-Board, das gezielt manipuliert worden ist. Bei sämtlichen Tests wurde das Ergebnis im erstellten Report mit der Art der Manipulation verglichen. Die Reports sind unter DVD/Funktionstest - Reports/* zu finden.

In Tabelle 5.4 im Anhang ist die Zuordnung der Reports zu den einzelnen Tests, bei denen ein Pin mit GND bzw. 5 V verbunden wurde, ersichtlich. In Tabelle 5.5 im Anhang sind weitere Reports den jeweiligen Tests zugeordnet. Zwischen den getesteten Fehlern und den daraus erstellten Reports konnten keinerlei Diskrepanzen festgestellt werden. Einige Fälle bedürfen allerdings einer Erläuterung. Im Falle von Report 48, bei dem ein Kurzschluss in der USB-Versorgungsspannung getestet wurde, werden alle Stromversorgungen als defekt erkannt. Dies resultiert aus der internen Verschaltung des Arduino Leonardos und nicht aus einer möglichen Fehlfunktion des Testgerätes. Selbiges gilt für Report 49. Für *VIN (9 V)* wird in diesem Fall widersprüchlich ein zu niedriger Strom angezeigt. Dies wiederum resultiert aus dem internen Aufbau des Testgerätes. Bei einem Kurzschluss der 9 V-Versorgung wird die Strombegrenzung aktiviert. Da bis zum Schalten des dritten Versorgungsspannungs-Relais einige Sekunden vergehen, erwärmt sich das Strombegrenzer-IC so stark, dass die interne Schutzbeschaltung den Stromfluss komplett sperrt. Aus diesem Grund beträgt der Strom 0 mA. Bei Report 54 wurden Pin 5 und Pin 7 miteinander verbunden. Im Report wird aber nur Pin 7 als defekt angezeigt. Der Grund hierfür ist die Tatsache, dass Pin 5 bei diesem Board offensichtlich eine niedrigere Ausgangsimpedanz besitzt und den damit verbundenen Pin 7 auf den annähernd gleichen Spannungswert bringt. Pin 5 kann in diesem Fall nicht als defekt erkannt werden.

3 Bedienungsanleitung

Dieses Kapitel beschreibt die Handhabung des Testgerätes und fasst die wichtigsten für die Bedienung des Gerätes relevanten Informationen zusammen.

Zum Testen eines Mikrocontroller-Boards wird das Gerät an das vorgesehene Netzteil angeschlossen und mit dem EIN/AUS-Schalter an der Gehäuserückseite angeschaltet. Sobald der Startvorgang des Testgerätes beendet ist, erscheint ein Menü auf dem Display. Zur Auswahl stehen *Board testen* und *Reports auf USB-Stick speichern*. Mit Hilfe der beiden Pfeiltasten kann der gewünschte Eintrag ausgewählt werden. Durch das Betätigen der OK-Taste wird der entsprechende Befehl ausgeführt.

3.1 Menü-Eintrag *Board testen*

Wenn ein Board getestet werden soll, wird dieses zunächst auf den Adapter gesteckt. Hierbei ist insbesondere darauf zu achten, dass der Prüfstift zum Verbinden des *SPI_SS*-Signals mit dem entsprechenden Testpunkt richtig kontaktiert ist. Dieser Prüfstift ist deshalb notwendig, da der *SPI_SS*-Pin des Mikrocontrollers nicht als Anschlusspin des Leonardo-Boards herausgeführt wird, jedoch zur Kommunikation via SPI zwingend erforderlich ist. Anschließend wird der Adapter mit dem Testgerät verbunden. Zusätzlich müssen noch das USB- und das PWR-Kabel angeschlossen werden. Dieser Schritt kann entweder im Untermenü *Board testen* oder bereits im Hauptmenü erfolgen. Durch Betätigen der OK-Taste wird der Test gestartet. Sobald dieser beendet ist, kann der Adapter vom Testgerät abgesteckt werden. Zum Entfernen des Boards wird das ebenfalls in dieser Arbeit erstellte Hilfsmittel (*Board-Entferner*) verwendet. Hiermit ist es möglich, den Adapter durch leichten Druck in Richtung der Tischplatte vom Board zu trennen, ohne die Pins zu verbiegen. Die erstellten Reports können prinzipiell über zwei unterschiedliche Arten auf den Computer übertragen werden: entweder durch Kopieren auf einen USB-Stick oder durch einen Zugriff auf die Reports via Ethernet. Hierzu sind die nachfolgenden Kapitel zu beachten. Im Falle eines Zugriffs per Ethernet befinden sich die Reports in dem Verzeichnis /home/Testprogramm/Data/Reports/*.

3.2 Menü-Eintrag *Reports auf USB-Stick speichern*

Zum Kopieren der Reports auf einen USB-Stick wird dieser in eine der beiden USB-Buchsen neben dem LAN-Anschluss eingesteckt. Dies muss erfolgen, bevor dieser Menü-Eintrag mit OK bestätigt wird. Das Betätigen der OK-Taste startet den Kopiervorgang. An-

schließend kann der USB-Stick entfernt werden. Nach einem erfolgreichem Kopiervorgang befindet sich nun auf dem USB-Stick ein Ordner, in dem sich sämtliche auf dem Testgerät gespeicherten Reports befinden.

3.3 Verbinden via SSH

Das Verbinden mit dem Testgerät via SSH erfolgt durch

```
1 config@IP
```

mit der entsprechenden IP-Adresse und dem Passwort **config**. Sofern es vom Betriebssystem des Computers unterstützt wird, kann auch eine Verbindung über den Hostnamen erfolgen:

```
1 config@ArduinoTester
```

Bei einem Windows-Betriebssystem kann zum Herstellen einer SSH-Verbindung beispielsweise das Programm *Putty*[1] verwendet werden.

3.4 Datenaustausch via Ethernet

Der Datenaustausch via Ethernet kann prinzipiell über FTP oder den eingerichteten Fileserver erfolgen. Hierzu muss das Testgerät über den LAN-Anschluss an ein Netzwerk angeschlossen werden.

3.4.1 FTP

Eine FTP-Verbindung kann über die **IP**-Adresse bzw. über den Hostnamen **ArduinoTester** mit dem Benutzernamen **config** hergestellt werden. Das Passwort lautet ebenfalls **config**. Dies kann z. B. mit den Programmen WinSCP[2] oder Filezilla[3] erfolgen.

3.4.2 Samba-Fileserver

Durch den auf dem Testgerät eingerichteten Samba-Fileserver kann auf alle Daten, die sich im Verzeichnis /home/Testprogramm/ befinden, über das Samba-Protokoll zugegriffen werden. Dies stellt eine komfortable Lösung dar, um beispielsweise die Testsoftware zu verändern oder auf die erstellten Reports zuzugreifen. Unter Windows wird hierfür im Arbeitsplatz die Netzwerkadresse **ArduinoTester****Testprogramm** hinzugefügt. Bei Linux-basierten Betriebssystemen kann z. B. mittels

[1] http://www.putty.org/ (zuletzt aufgerufen am 25.08.2015).
[2] https://winscp.net (zuletzt aufgerufen am 25.08.2015).
[3] https://filezilla-project.org/ (zuletzt aufgerufen am 25.08.2015).

```
1 smbclient //IP/Testprogramm -U config
```

eine Verbindung hergestellt werden. Auf dem jeweiligen Linux-Rechner muss ebenfalls *Samba* installiert sein.

3.5 Ändern des Programmcodes

Wenn der Programmcode der Testsoftware geändert werden soll, so kann dies z. B. via SSH mit dem bereits installierten Editor *nano* erfolgen. Eine deutlich komfortablere Lösung ist jedoch der Zugriff durch den konfigurierten Fileserver und das Bearbeiten des Quellcodes mit einem Editor des Computers. In dem Verzeichnis /home/Testprogramm/-Source/* befindet sich neben den Quelldateien ebenfalls die Datei Program.pro. Das ist die Projektdatei des Programms *Qt Creator*[4], welches zur Programmierung verwendet wurde.

Falls der Programmcode geändert worden ist und dieser nun ausgeführt werden soll, ist wie folgt vorzugehen:

```
1 sudo kill $(pidof /home/Testprogramm/program)
2 cd /home/Testprogramm
3 make
```

In der ersten Zeile wird das laufende Programm beendet. Der Befehl *make* erstellt die ausführbare Datei aus dem Quellcode. Bei einem Neustart des Testgerätes wird dieses in Zukunft mit der neu erstellten Software geladen. Zum Testen des Programms kann dieses auch manuell mit

```
1 sudo ./program
```

gestartet werden.

3.6 Testen von anderen Mikrocontroller-Boards

Die Hardware des Testgerätes unterstützt prinzipiell eine Vielzahl weiterer Mikrocontroller-Boards (siehe Kapitel 1.2.2), an der Software sind jedoch Änderungen bzw. Erweiterungen erforderlich. Außerdem muss für jedes weitere Modell ein individueller Adapter hergestellt werden. Die Vorgehensweise hierzu ist Kapitel 2.4.4 zu entnehmen.

3.6.1 Erweitern der Software

Für jedes weitere Board, das getestet werden soll, sind die Pin-Zuordnungen im Konstruktor der Klasse *Testboard* vorzunehmen. Als Vorlage dienen die bereits für den Arduino

[4] http://www.qt.io/ (zuletzt aufgerufen am 28.08.2015)

Leonardo eingetragenen Daten, die an der entsprechenden Stelle einzufügen und abzuändern sind. Einzelheiten ergeben sich aus dem kommentierten Quellcode.

Die Kommunikation zwischen dem Raspberry Pi 2 und dem Arduino Leonardo verläuft während des Testes ausschließlich über die SPI-Schnittstelle, da diese Pins vollkommen unabhängig von den restlichen Datenpins sind. Bei anderen Mikrocontroller-Boards, bei denen ein SPI-Pin zugleich auch beispielsweise als digitaler Ausgang genutzt werden kann, ist es notwendig, dass eine Kommunikation über eine andere Schnittstelle unterstützt wird, um auch diese SPI-Pins prüfen zu können. Dieser Fall wurde im Rahmen der Arbeit nicht implementiert und ist daher noch zu erledigen, sofern eine Überprüfung dieser Pins gewünscht ist. Die Software, die auf den Mikrocontroller geflasht wird, ist ebenfalls dem entsprechenden Board anzupassen.

4 Schlussfolgerung

Ziel dieser Arbeit war es, ein Testgerät zu entwickeln und aufzubauen, mit dem die wichtigsten Funktionen eines speziellen Mikrocontroller-Boards - einem Arduino Leonardo - in einfacher Weise getestet werden können. Hierzu zählen insbesondere die digitalen Ein- und Ausgänge, die analogen Eingänge sowie die SPI- und I^2C-Schnittstelle. Aufgrund weitsichtiger Überlegungen ist hieraus ein Universal-Testgerät basierend auf einem weit verbreiteten Ein-Platinen-Computer entstanden, das nicht nur dieses spezielle Board zu testen vermag, sondern auch eine Vielzahl anderer Mikrocontroller-Boards unterstützt. Zusatzfunktionen, wie z. B. das Überspielen eines Standardprogramms, das Setzen der Fuses und die Erstellung eines Reports sind ebenfalls Teil des Funktionsumfangs des Testgerätes. Im Rahmen dieser Arbeit wurden theoretische Grundlagen von Testgeräten eruiert, Hardwarekomponenten konstruiert und geeignete Testverfahren entwickelt, die anschließend in der Software implementiert worden sind. Zu Beginn wurden zum Erlangen eines genauen Verständnisses die verwendeten Hardwarekomponenten, wie auch die Funktionsweisen der relevanten Datenbusse beschrieben. Im Entwicklungs-Teil der Arbeit wurden der prinzipielle Aufbau von Testgeräten sowie typische Messgrößen und -verfahren erläutert. Darauf basierend konnte die Hardware des Testgerätes entwickelt werden, dessen Aufbau bzw. Montage im Anschluss beschrieben worden ist. Bezüglich der Software mussten neben der Installation und Konfiguration des Betriebssystems Testverfahren und -abläufe entwickelt und in einer geeigneten Programmiersprache implementiert werden. Nach dem Aufbau des Testgerätes erfolgte die Inbetriebnahme mit mehreren anschließenden Tests zur Sicherstellung der korrekten Funktionsweise des Gerätes. In der Bedienungsanleitung wurden die wichtigsten für die Bedienung des Gerätes relevanten Informationen zusammengefasst und die Vorgehensweisen zum Ändern bzw. Erweitern des Programmcodes vorgestellt.

Mit dieser Arbeit ist die Grundlage für ein handliches Universal-Testgerät geschaffen worden, welches dazu in der Lage ist, eine Vielzahl an Mikrocontroller-Boards auf korrekte Funktionsweise zu überprüfen. Die Gesamtkosten des Testgerätes (inkl. des Adapters) belaufen sich auf ca. 350 Euro, wovon in etwa 150 Euro für die Herstellung der beiden Leiterplatten einkalkuliert worden sind. Zur Unterstützung anderer Platinen müssen noch die entsprechenden Adapter entwickelt und hergestellt sowie Softwareanpassungen vorgenommen werden. Für die Zukunft ist in jedem Fall eine Weiterentwicklung des Testgerätes sowohl in der Hardware als auch in der Software denkbar und naheliegend.

5 Anhang

5.1 Abbildungen

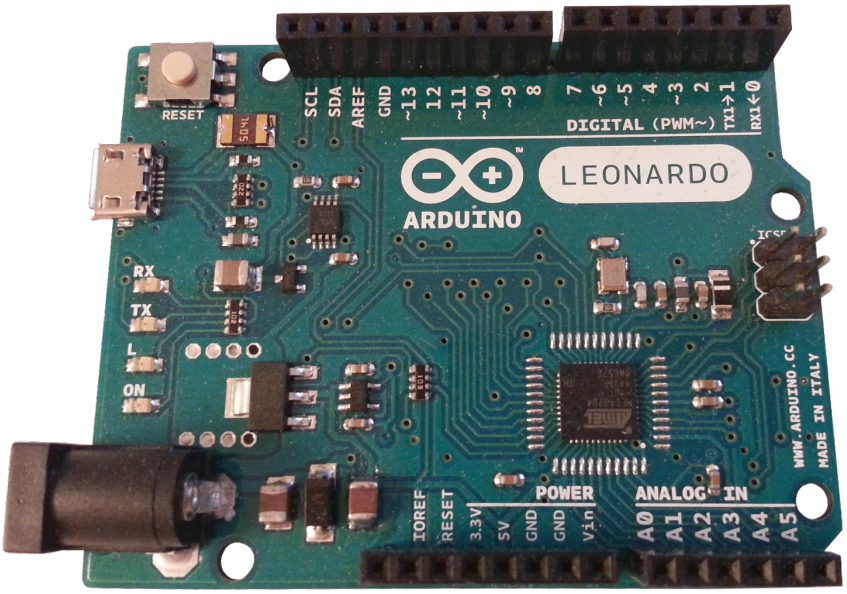


Abbildung 5.1: Arduino Leonardo-Board (Ansicht von oben).

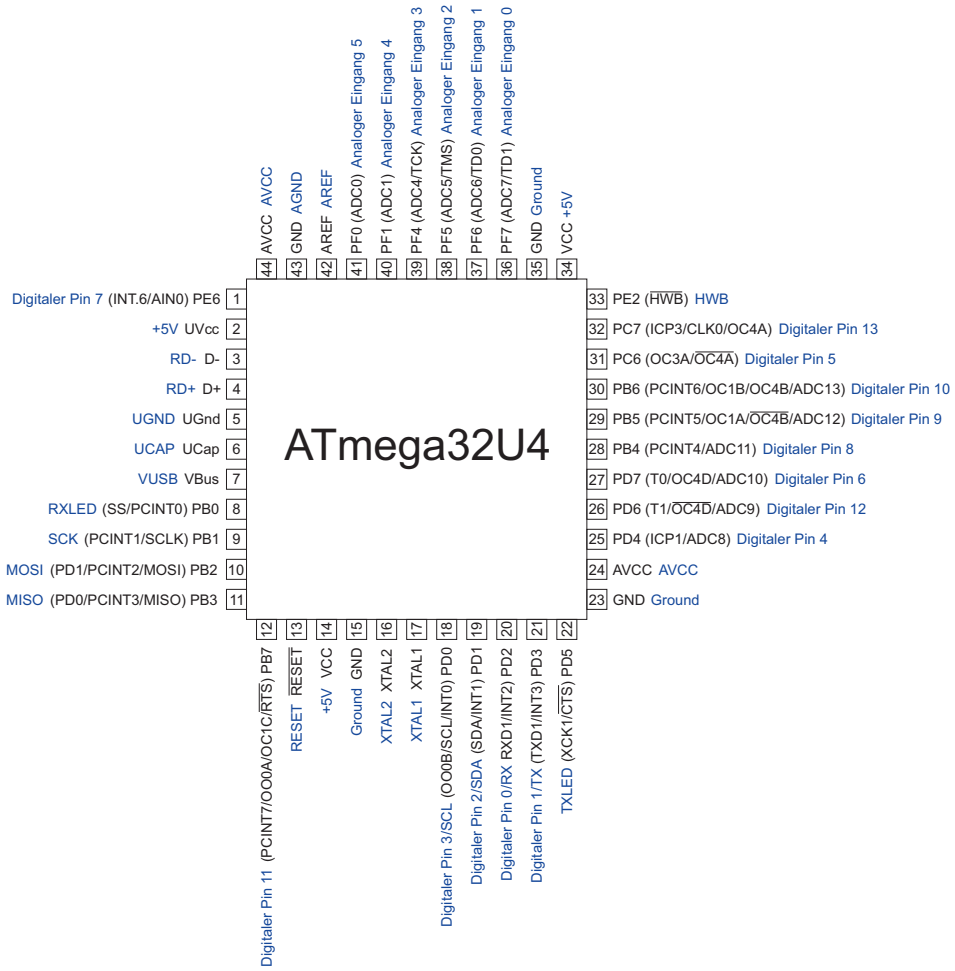


Abbildung 5.2: Pinzuordnung zwischen ATmega32u4 und Arduino Leonardo (Belegung von [14] übernommen).

DC Power	3,3 V	1	2	5 V	DC Power
SDA1, I2C	GPIO02	3	4	5 V	DC Power
SCL1, I2C	GPIO03	5	6	GND	
GPIO_GCLK	GPIO04	7	8	GPIO14	TXD0
	GND	9	10	GPIO15	RXD0
GPIO_GEN0	GPIO17	11	12	GPIO18	GPIO_GEN1
GPIO_GEN2	GPIO27	13	14	GND	
GPIO_GEN3	GPIO22	15	16	GPIO23	GPIO_GEN4
DC Power	3,3 V	17	18	GPIO24	GPIO_GEN5
SPI_MOSI	GPIO10	19	20	GND	
SPI_MISO	GPIO09	21	22	GPIO25	GPIO_GEN6
SPI_CLK	GPIO11	23	24	GPIO08	SPI_CE0_N
	GND	25	26	GPIO07	SPI_CE1_N
I2C ID EEPROM	ID_SD	27	28	ID_SC	I2C ID EEPROM
	GPIO05	29	30	GND	
	GPIO06	31	32	GPIO12	PWM0
PWM1	GPIO13	33	34	GND	
SPI1_MISO, PCM_FS	GPIO19	35	36	GPIO16	SPI1_CE3
	GPIO26	37	38	GPIO20	SPI1_MOSI, PCM DIN
	GND	39	40	GPIO21	SPI1_CLK, PCM DOUT

Abbildung 5.3: Pinbelegung von GPIO-Schnittstelle des Raspberry Pi 2 (Belegung von [3] übernommen).

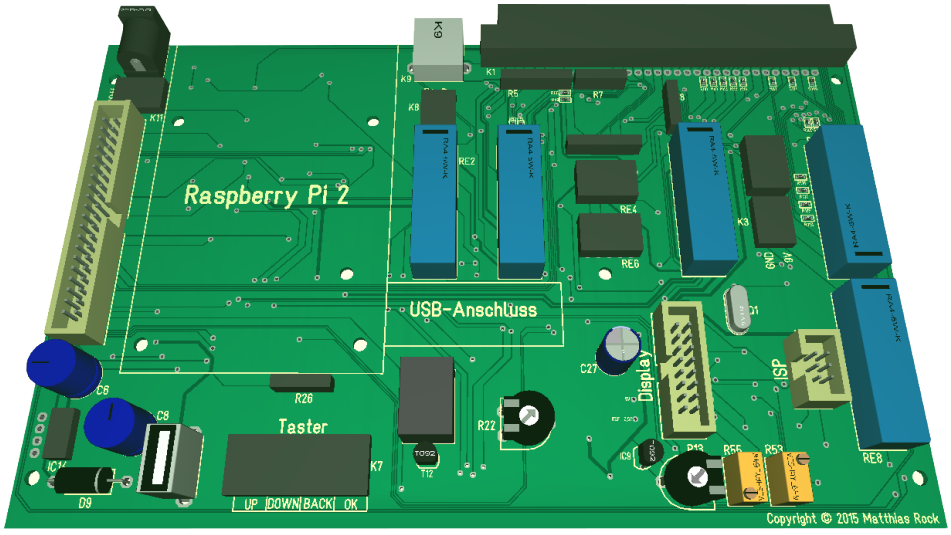


Abbildung 5.4: 3D-Ansicht von entwickelter Platine (oben).

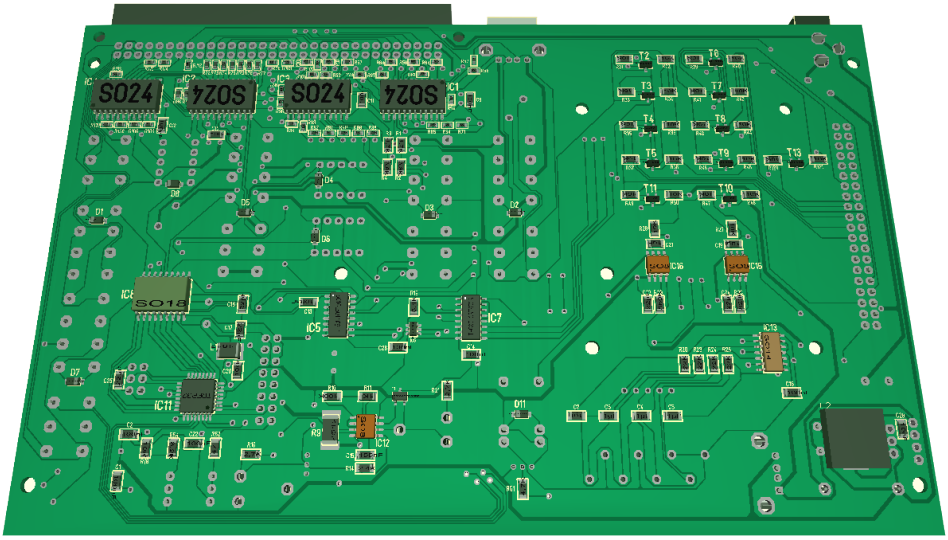


Abbildung 5.5: 3D-Ansicht von entwickelter Platine (unten).

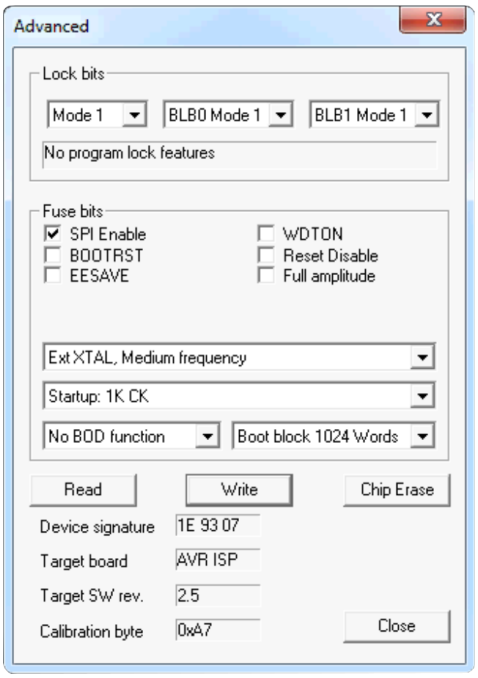


Abbildung 5.6: Notwendige Einstellungen (Fuses) von ATmega8.

5.2 Tabellen

Funktion	Anzahl Pins	Pins
Digitale I/0-Pins	20	A0 - A5, 0 - 13
Analoge Eingänge	12	A0 - A5, 4, 6, 8 - 10, 12
PWM-Ausgänge	7	3, 5, 6, 9 - 11, 13
Serielle Datenübertragung	2	0 (Rx), 1 (Tx)
TWI/I^2C-Schnittstelle	2 (+2)	2 (SDA), 3 (SCL), SDA, SCL
Externe Interrupts	5	0 - 3, 7

Tabelle 5.1: Mögliche Funktionen für I/O-Pins von Arduino Leonardo [4].

Board	Besonderheit
Leonardo	-
Uno (R3)	1 Digitalpin als SS für SPI nutzbar 3 Digitalpins sind mit 6-pol. SPI verbunden 1 zusätzlicher 6-pol. SPI für USB-Controller
Mini (R05)	3 Digitalpins als SPI nutzbar Normale Programmierung erfolgt über UART
Pro Mini	4 Digitalpins als SPI nutzbar Zusätzlich 1 DTR-Input (über Kondensator mit Reset verbunden)
Micro	1 Digitalpin als SS für SPI nutzbar 3 zusätzliche Pins existieren für SPI (keine Digitalpins)
Nano	1 Digitalpin als SS für SPI nutzbar 3 Digitalpins sind mit 6-pol. SPI verbunden

Tabelle 5.2: Besonderheiten von mehreren Arduino-Boards [4], [15]–[19].

Eigenschaft	**Leonar.**	**Uno (R3)**	**Mini (R05)**	**Pro Mini**	**Micro**	**Nano**
Mikrocontroller	ATmega 32U4	ATmega 328P	ATmega 328	ATmega 328	ATmega 32U4	ATmega 168 / 328
Betriebsspannung	5 V	5 V	5 V	3,3 V / 5 V	5 V	5 V
DC-In (Hohlbuchse)	7 - 12 V	7 - 12 V	-	-	-	-
VIN-Pin	7 - 12 V	7 - 12 V	7 - 9 V	3,35 - 12 V / 5 - 12 V	7 - 12 V	7 - 12 V
#USB	1	1	0	0	1	1
#SPI (6-pol.)	1	1	0	0	1	1
#Belegte Pins (ohne 6-pol. SPI)	31	31	35	34	32	30
#5 V-Pins	1	1	3	0 / 2	1	1
#3,3 V-Pins	1	1	0	2 / 0	1	1
#GND-Pins	3	3	4	4	2	2
#IOREF-Pins	1 (5 V)	1 (5 V)	0	0	0	0
#AREF-Pins	1	1	0	0	1	1
#RESET-Pins	1	1	1 (+1)	1 (+1)	1	1 (+1)
#Datenpins (ohne 6-pol. SPI)	20 (+2)	20 (+2)	22 (+3)	22 (+2)	20	22
#Digitale I/O-Pins	20 (+2)	14	14	14	20	14
#Separate analoge Eingänge	0	6 (+2)	8	6	0	8
#Analoge Eingänge von dig. I/O	12	0	0	0	12	0
#PWM-Ausgänge von dig. I/O	7	6	6	6	7	6
#UART-Pins von dig. I/O	2	2	2 (+2)	2 (+2)	2	2
#I^2C-Pins	2 (+2) dig. IO	2 (+2) analog	0	2 analog	2 dig. IO	2 analog

Tabelle 5.3: Spezifikationen von mehreren Arduino-Boards [4], [15]–[19].
Zum Verständnis: der Ausdruck **20 (+2)** bei **#Digitale I/O-Pins** bedeutet beispielsweise, dass 20 digitale I/O-Pins existieren und davon zwei Pins jeweils einen zusätzlichen Pin-Anschluss besitzen.

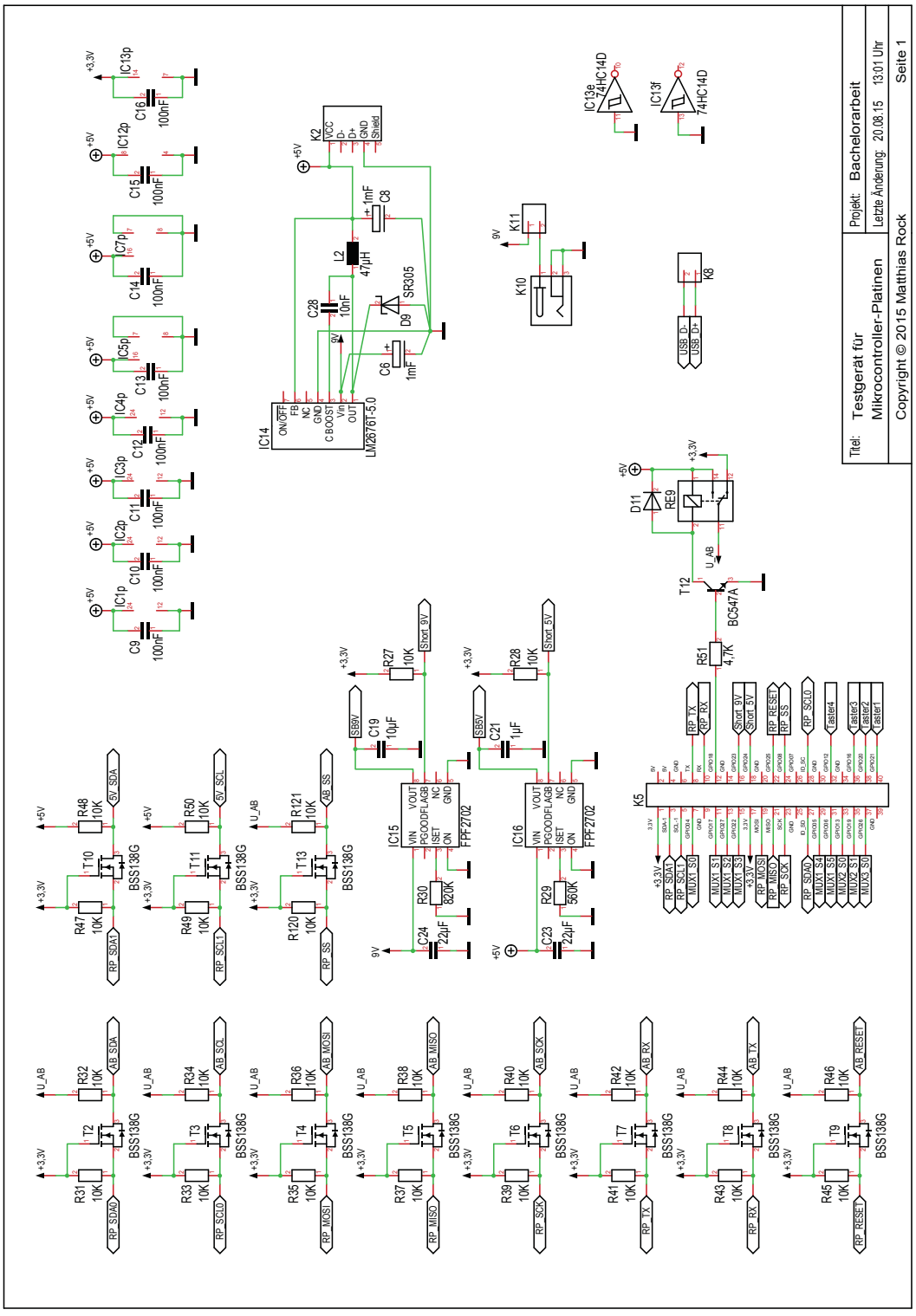
Pin ... verbunden mit...	**... GND**	**... 5 V**
RX←0	Report_2	Report_24
TX→1	Report_3	Report_25
2	Report_4	Report_26
∼3	Report_5	Report_27
4	Report_6	Report_28
∼5	Report_7	Report_29
∼6	Report_8	Report_30
7	Report_9	Report_31
8	Report_10	Report_32
∼9	Report_11	Report_33
∼10	Report_12	Report_34
∼11	Report_13	Report_35
12	Report_14	Report_36
∼13	Report_15	Report_37
SDA	Report_16	Report_38
SCL	Report_17	Report_39
A0	Report_18	Report_40
A1	Report_19	Report_41
A2	Report_20	Report_42
A3	Report_21	Report_43
A4	Report_22	Report_44
A5	Report_23	Report_45

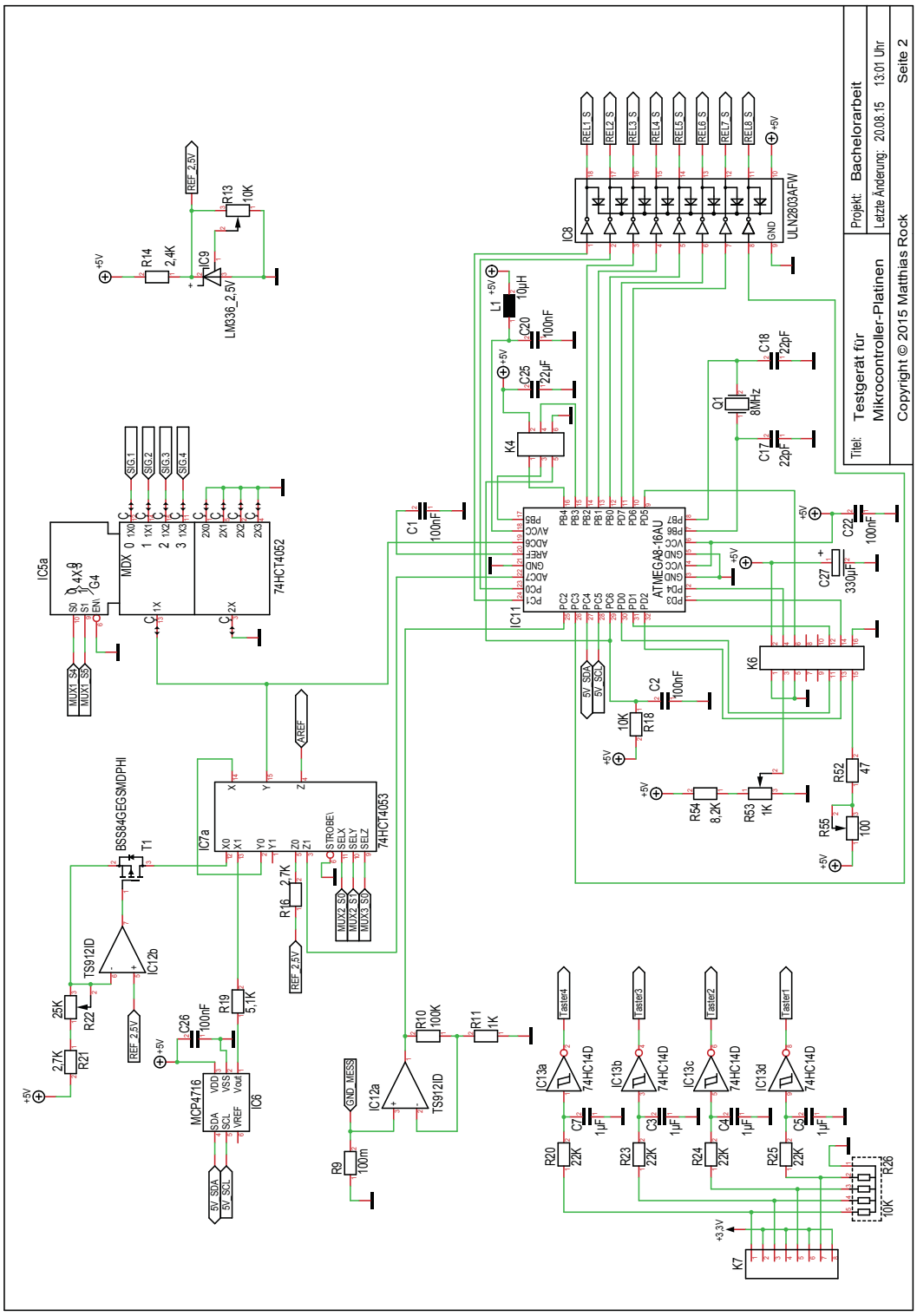
Tabelle 5.4: Funktionstest von fertigem Gerät: Verbindung der Pins mit GND bzw. 5 V.

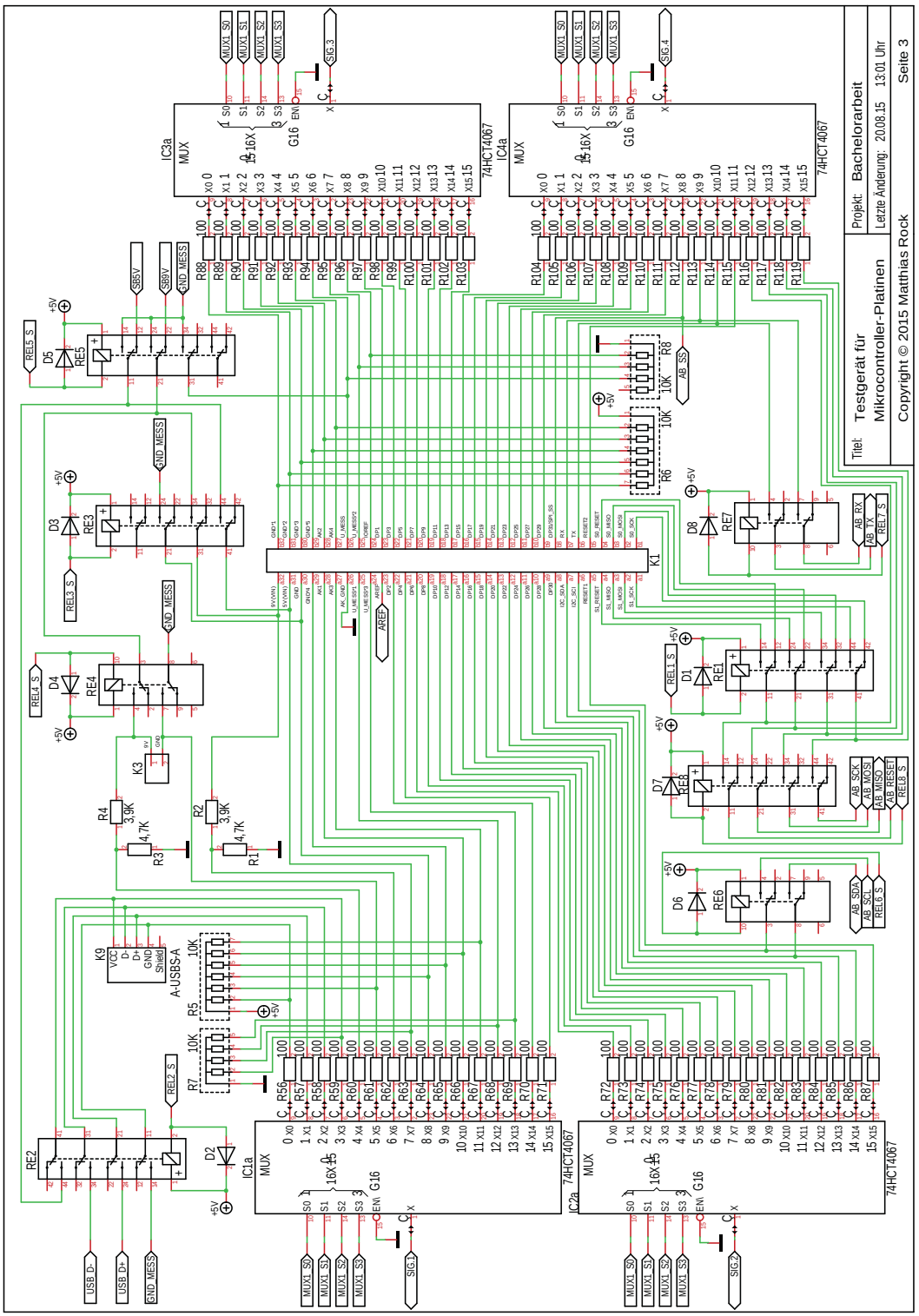
Report	**Art der Manipulation**
Report_1:	Voll intaktes Board
Report_46:	USB-Kabel nicht angeschlossen
Report_47:	PWR (DC-In) nicht angeschlossen
Report_48:	Kurzschluss USB_GND↔USB_5V
Report_49:	Kurzschluss PWR_GND↔PWR_9V
Report_50:	Kein Board angeschlossen
Report_51:	Board nur per USB-Kabel angeschlossen (nicht auf Adapter aufgesteckt)
Report_52:	Board nur per PWR angeschlossen (nicht auf Adapter aufgesteckt)
Report_53:	SPI_SS nicht verbunden
Report_54:	Kurzschluss 5↔7
Report_55:	Kurzschluss 5↔6↔7
Report_56:	Kurzschluss 10↔A1↔A2

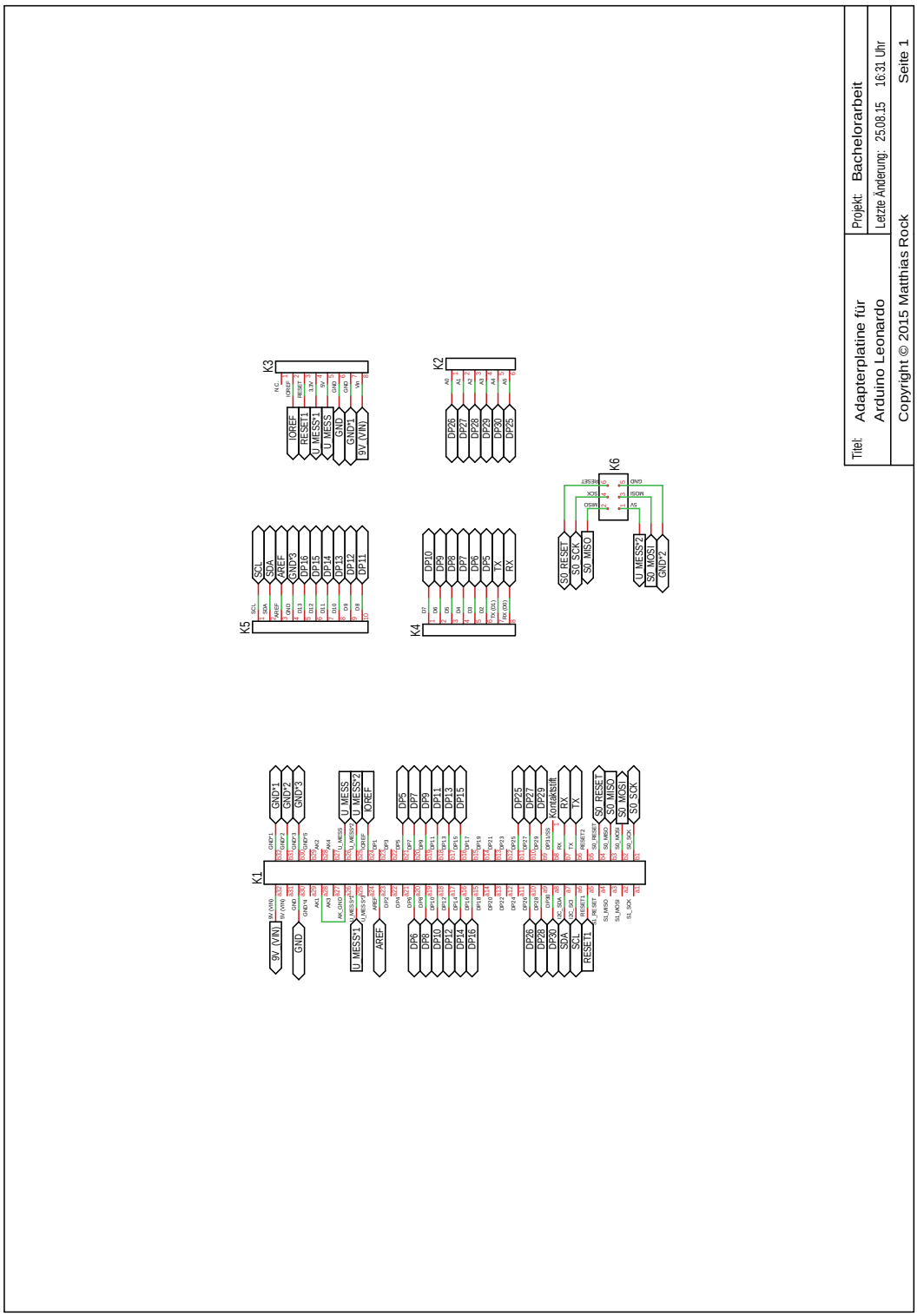
Tabelle 5.5: Funktionstest von fertigem Gerät: Diverse Tests.

5.3 Schaltpläne




Projekt: Bachelorarbeit
Titel: Testgerät für Mikrocontroller-Platinen
Letzte Änderung: 20.08.15 13.01 Uhr
Seite 2
Copyright © 2015 Matthias Rock
ATMEGA8-16AU
ULN2803AFW
74HCT4052
74HCT4053
74HC14D
TS912ID
LM336_2,5V
MCP4716
BSS84GEGSMDPHI
8MHz
10µH
+5V
+3,3V
REF_2,5V
AREF
GND_MESS
5V_SDA
5V_SCL
MUX1_SA
MUX1_SB
MUX2_S0
MUX2_S1
MUX3_S0
STROBE\
SELX
SELY
SELZ
SIG.1
SIG.2
SIG.3
SIG.4
REL1_S
REL2_S
REL3_S
REL4_S
REL5_S
REL6_S
REL7_S
REL8_S
Taster1
Taster2
Taster3
Taster4


Titel: Testgerät für Mikrocontroller-Platinen
Projekt: Bachelorarbeit
Letzte Änderung: 20.08.15 13:01 Uhr
Seite 3
Copyright © 2015 Matthias Rock
74HCT4067
MUX
IC1a
IC2a
IC3a
IC4a
A-USBS-A
K1
K3
K9
RE1
RE2
RE3
RE4
RE5
RE6
RE7
RE8
D1
D2
D3
D4
D5
D6
D7
D8
R1
R2
R3
R4
R5
R6
R7
R8
3,9K
4,7K
10K
+5V
GND_MESS
USB D-
USB D+
AB_SCK
AB_MOSI
AB_MISO
AB_RESET
AB_RX
AB_TX
AB_SDA
AB_SCL
AB_SS


Titel: Adapterplatine für Arduino Leonardo
Projekt: Bachelorarbeit
Letzte Änderung: 25.08.15 16:31 Uhr
Seite 1
Copyright © 2015 Matthias Rock
K1
K2
K3
K4
K5
K6
9V (VIN)
GND
U_MESS*1
AREF
DP6
DP8
DP10
DP12
DP14
DP16
RESET1
SDA
SCL
DP26
DP28
DP30
GND*1
GND*2
GND*3
U_MESS
U_MESS2
IOREF
DP5
DP7
DP9
DP11
DP13
DP15
DP25
DP27
DP29
Kontaktstift
RX
TX
S0_RESET
S0_MISO
S0_SCK
U_MESS2
S0_MOSI
GND*2

5.4 Konfiguration von Raspberry Pi 2

Hier werden alle notwendigen Konfigurationsschritte beschrieben, die am Raspberry Pi 2 vorgenommen werden müssen. Die Konfiguration erfolgt via SSH:

```
1 ssh sysop@IP
```

Unter Windows kann dazu beispielsweise das Programm *Putty*[1] verwendet werden.

Zunächst wird ein neuer Benutzer **config** angelegt:

```
1 sudo adduser config
2 sudo usermod config -aG sudo
```

mit dem Passwort, das ebenfalls **config** lautet (Full Name: **Configuration**). Anschließend wird der Hostname **pipaos** mit

```
1 sudo nano /etc/hostname
2 sudo nano /etc/hosts
```

in beiden Dateien durch **ArduinoTester** ersetzt und der Raspberry Pi neu gestartet:

```
1 sudo reboot
```

Nun erfolgt die SSH-Verbindung via:

```
1 ssh config@IP
```

und der Benutzer *sysop* kann mit

```
1 sudo deluser --remove-home sysop
```

gelöscht werden. Damit das Testgerät zukünftig über den Hostnamen angesprochen werden kann, wird *Samba*[2] installiert und ein Benutzer angelegt (Passwort: **config**):

```
1 sudo apt-get update
2 sudo apt-get install samba
3 sudo smbpasswd -a config
```

Dies funktioniert jedoch nicht mit allen Betriebssystemen, sondern in der Regel nur unter Windows. Hier kann zukünftig eine SSH-Verbindung wie folgt aufgebaut werden:

```
1 ssh config@ArduinoTester
```

Nun wird noch die Nachricht, die beim SSH-Login angezeigt wird, in **Welcome to ArduinoTester!** geändert:

```
1 sudo nano /etc/motd
```

[1] http://www.putty.org/ (zuletzt aufgerufen am 25.08.2015).
[2] https://www.samba.org/ (zuletzt aufgerufen am 25.08.2015).

Um einen einfachen Dateiaustausch zu ermöglichen, wird sowohl ein FTP- als auch ein File-Server eingerichtet. Dazu wird zunächst ein neues Verzeichnis erstellt und *ProFTPD*[3] installiert:

```
1 sudo mkdir /home/Testprogramm
2 sudo chmod 777 /home/Testprogramm
3 sudo apt-get install proftpd
```

In der Installationsroutine wird **standalone** ausgewählt. Nun kann mit einem beliebigen FTP-Programm, wie beispielsweise WinSCP[4] oder Filezilla[5] ein Datenaustausch stattfinden. Zum Einrichten des Fileservers wird die bestehende Konfigurationsdatei von *Samba* gelöscht und eine neue erstellt:

```
1 sudo rm /etc/samba/smb.conf
2 sudo nano /etc/samba/smb.conf
```

In diese wird folgender Inhalt hineingeschrieben:

```
1  [global]
2  workgroup = WORKGROUP
3  server string = %h server
4  log file = /var/log/samba/log.%m
5  security = user
6  map to guest = bad user
7  [Testprogramm]
8  path = /home/Testprogramm
9  public = no
10 guest ok = no
11 write list = config
12 browseable = yes
13 force create mode = 0777
14 force directory mode = 0777
```

Danach muss *Samba* neu gestartet werden:

```
1 sudo /etc/init.d/samba restart
```

Unter Windows kann jetzt beispielsweise im Arbeitsplatz die Netzwerkadresse **ArduinoTester****Testprogramm** hinzugefügt werden. Bei Linux-basierten Betriebssystemen kann z. B. mittels

```
1 smbclient //IP/Testprogramm -U config
```

[3] http://www.proftpd.org/ (zuletzt aufgerufen am 25.08.2015).
[4] https://winscp.net (zuletzt aufgerufen am 25.08.2015).
[5] https://filezilla-project.org/ (zuletzt aufgerufen am 25.08.2015).

eine Verbindung hergestellt werden. Hierfür muss auf dem jeweiligen Rechner ebenfalls *Samba* installiert sein.

Damit das zu testende Mikrocontroller-Board überprüft werden kann, ist es notwendig, ein Testprogramm auf den Mikrocontroller aufzuspielen. Dies erfolgt über die SPI-Schnittstelle. Als Software dient dazu **Avrdude**. Die Vorgehensweise zur Installation wurde von [20] übernommen. Zunächst müssen erst einige andere Programme installiert werden:

```
1 sudo apt-get install gcc git-core make bison autoconf flex
```

Danach wird *Avrdude* heruntergeladen und installiert:

```
1 git clone https://github.com/kcuzner/avrdude
2 cd avrdude/avrdude
3 ./bootstrap && ./configure
4 sudo make install
```

Des Weiteren müssen sowohl die SPI- als auch die zwei I^2C-Schnittstellen aktiviert werden. Hierfür wird die *config.txt*

```
1 sudo nano /boot/config.txt
```

um

```
1 dtparam=spi=on
2 dtparam=i2c_arm=on
3 dtparam=i2c1=on
4 dtparam=i2c0=on
5 dtparam=i2c1_baudrate=10000
6 dtparam=i2c0_baudrate=10000
```

ergänzt und anschließend muss der Raspberry Pi neu gestartet werden [21]–[23]:

```
1 sudo reboot
```

Da das Testprogramm für den Raspberry Pi in C++ 11 programmiert wird, die entsprechende g++-Version jedoch zum Zeitpunkt der Erstellung dieser Arbeit noch nicht in den offiziellen Paketlisten des Betriebssystems verfügbar ist, muss zur Installation wie folgt vorgegangen werden: Zunächst wird in folgender Datei der Name **wheezy** durch **jessie** ausgetauscht:

```
1 sudo nano /etc/apt/sources.list
```

Anschließend werden die Paketlisten aktualisiert und *g++-4.9* installiert:

```
1 sudo apt-get update
2 sudo apt-get install g++-4.9
```

Die während der Installation auftauchende Meldung wird mit *Yes* bestätigt. Danach wird in der vorhin bearbeiteten Datei **jessie** durch **wheezy** ersetzt und die Paketlisten aktualisiert [24]:

```
1 sudo nano /etc/apt/sources.list
2 sudo apt-get update
```

Für den unkomplizierten Zugriff durch das in C++ geschriebene Programm auf die GPIO-Pins wird WiringPi[6] verwendet. Die Installation erfolgt durch

```
1 git clone git://git.drogon.net/wiringPi
2 cd wiringPi
3 ./build
```

und ermöglicht unter anderem auch die Kommunikation via I^2C und SPI [25].

Damit die I^2C-Schnittstelle direkt nach jedem Bootvorgang verfügbar ist, muss in

```
1 sudo nano /etc/modules
```

der Eintrag **i2c_dev** hinzugefügt werden. Nach einem anschließenden Neustart ist nun die Kommunikation via I^2C möglich [26].

In dem Testprogramm soll die Möglichkeit bestehen, die erstellen Reports auf einen USB-Stick kopieren zu können. Um eine möglichst einfache Implementierung zu gewährleisten, muss dafür gesorgt werden, dass der USB-Stick automatisch gemountet wird:

```
1 sudo apt-get install usbmount
2 sudo nano /etc/usbmount/usbmount.conf
```

In der obigen Datei wird der entsprechende Eintrag wie folgt geändert [27]:

```
1 FS_MOUNTOPTIONS="-fstype=vfat,gid=sudo,dmask=0007,fmask=0117"
```

Als letzter Schritt muss dafür gesorgt werden, dass das später erstellte Programm nach jedem Bootvorgang automatisch gestartet wird. Dies kann erreicht werden, indem in folgender Datei

```
1 sudo nano /etc/rc.local
```

der Eintrag /**home/Testprogramm/program** vor **exit 0** hinzugefügt wird [28]. Anschließend muss der Raspberry Pi neu gestartet werden:

```
1 sudo reboot
```

[6] http://wiringpi.com (zuletzt aufgerufen am 25.08.2015).

5.5 Programmablaufplan

- Testgerät wird eingeschaltet
- Ein-/Ausgänge initialisieren

MARKE 1:

- Display:
  - #Hauptmenü:
  - →Board testen
  - -Reports auf USB-
  - Stick speichern

Falls "Board testen" ausgewählt:

- Falls noch kein Adapter angeschlossen worden ist:
  - Display:
    - #Board testen:
    - Bitte Adapter
    - anschließen!
    -

MARKE 2:

- Falls Zurück-Taster gedrückt wird: **GOTO MARKE 1**
- Solange noch kein Adapter angeschlossen wurde: **GOTO MARKE 2**
- Display:
  - #Board testen:
  - Zum Testen von
  - Arduino Leonardo
  - mit OK bestätigen!

MARKE 3:

- Falls Zurück-Taster gedrückt wird: **GOTO MARKE 1**
- Solange noch nicht OK gedrückt wurde: **GOTO MARKE 3**
- Display:
  - #Board testen:
  - Arduino Leonardo
  - wird getestet...
  - (Power)
- RE9 schalten (Mikrocontrollerspannung = 5V)
- ***Alle Anschlüsse/Pins prüfen, die keine Datenpins sind***
  - MUX2 auf "Frei" schalten
  - USB-Spannung anlegen (RE2 = ON)
  - Spannung an USB-Pin (5V) messen

- Strom messen
- Short_5V abfragen
- RE2 = OFF
- PWR-Spannung anlegen (RE4 = ON)
- Spannung an PWR-Pin (9V) messen
- Strom messen
- Short_9V abfragen
- RE4 = OFF
- Spannung anlegen (RE3 = ON)
- Spannung an VIN-Pin (9V) messen
- Strom messen
- Short_5V abfragen
- Bisher gemessene Daten auswerten
- Prüfen, ob GND*, GND(USB) und GND(PWR) mit GND(VIN) verbunden sind
- U_MESS, U_MESS* messen
- IOREF messen

❖ Display:
- #Board testen:
- Arduino Leonardo
- wird getestet...
- (Durchgangspruefung)

❖ ***Durchgangsprüfung von allen Datenpins:***
- RE5 = ON
- MUX2 auf "Konstantstromquelle" einstellen

MARKE 4:

- Mit MUX1 Pin auswählen
- Spannung messen
- Solange noch nicht alle Datenpins gemessen wurden: **GOTO MARKE 4**

❖ MUX2 auf "frei" schalten
❖ RE5 = OFF
❖ Display:
- #Board testen:
- Arduino Leonardo
- wird getestet...
- (uC programmieren)

❖ SPI verbinden (RE8 = ON)
❖ Testprogramm auf µC aufspielen
❖ Display:
- #Board testen:
- Arduino Leonardo
- wird getestet...
- (Digitale Ausgaenge)

❖ ***Digitale Ausgänge prüfen:***

MARKE 5:

- Signal an µC per SPI senden: Alle I/O-Pins als Ausgänge und anschließend fünf Bytes für Pin-Zustände empfangen

MARKE 6:

- Mit MUX1 Pin auswählen
- Spannung an SIG messen
- Solange noch nicht alle relevanten Pins gemessen wurden: **GOTO MARKE 6**
- Solange an µC noch nicht alle Testmuster gesendet wurden: **GOTO MARKE 5**

❖ Display:
- #Board testen:
- Arduino Leonardo
- wird getestet...
- (Digitale Eingänge)

❖ ***Digitale Eingänge prüfen:***
- Signal an µC senden: alle I/O-Pins als Eingänge, Pullups deaktivieren
- MUX1 auf beliebigen Eingang schalten
- MUX2 einstellen
- Hohe LOW-Spannung an DAU einstellen

MAKRE 7:

- Mit MUX1 Pin auswählen
- Spannung an SIG messen, um evtl. Defekt am Eingang festzustellen
- An µC LOW/HIGH-Pegel bestimmen
- Solange noch nicht alle Eingänge gemessen wurden: **GOTO MARKE 7**
- Niedrige HIGH-Spannung an DAU einstellen

MARKE 8:

- Mit MUX1 Pin auswählen
- An µC LOW/HIGH-Pegel bestimmen
- Solange noch nicht alle Eingänge gemessen wurden: **GOTO MARKE 8**

❖ Display:
- #Board testen:
- Arduino Leonardo
- wird getestet...
- (Analoge Eingänge)

- ***Analoge Eingänge prüfen:***
  - Signal an µC senden: ADU aktivieren/initialisieren

  MARKE 9:

  - Gewünschte Spannung an DAU einstellen

  MARKE 10:

  - Mit MUX1 Pin auswählen
  - An µC Spannung messen
  - Solange noch nicht alle analogen Eingänge gemessen wurden: **GOTO MARKE 10**
  - Solange noch andere Spannungen getestet werden sollen: **GOTO MARKE 9**
- Display:
  - #Board testen:
  - Arduino Leonardo
  - wird getestet...
  - (I2C-Schnittstelle)
- ***I2C testen:***
  - Signal an µC senden: I2C aktivieren
  - Test-Byte an µC via I2C senden
  - Test-Byte von µC via I2C empfangen
- Display:
  - #Board testen:
  - Test beendet!
  - Standardprogramm
  - wird aufgespielt...
- Standardprogramm (mit Bootloader) auf µC brennen
- Display:
  - #Board testen:
  - Test beendet!
  - Fuses werden
  - gesetzt...
- Fuses setzen
- Display:
  - #Board testen:
  - Fertig!
  - Daten werden
  - ausgewertet...

Literaturverzeichnis

[1] M. Rost und S. Wefel, *Elektronik für Informatiker - Von den Grundlagen bis zur Mikrocontroller-Applikation*. München: Oldenbourg Wissenschaftsverlag, 2013, S. 404–405.

[2] H.-W. Huang, *The Atmel AVR Microcontroller: Mega and XMega in Assembly and C*. Clifton Park, NY, USA: Cengage Learning, 2013, S. 22–85.

[3] K. Dembowski, *Raspberry Pi - Das technische Handbuch: Konfiguration, Hardware, Applikationserstellung*, 2. Aufl., Wiesbaden: Springer Vieweg, 2015.

[7] H. Bernstein, *Mikrocontroller: Grundlagen der Hard- und Software der Mikrocontroller ATtiny2313, ATtiny26 und ATmega32*. Wiesbaden: Springer Vieweg, 2015, S. 106–108.

[8] G. Roberts, F. Taenzler und M. Burns, *An Introduction to Mixed-Signal IC Test and Measurement*, 2. Aufl., Oxford, NY, USA: Oxford University Press, 2011.

Internetquellen

[4] Arduino LLC, *Arduino - ArduinoBoardLeonardo*, Zuletzt aufgerufen am 29.04.2015. Adresse: http://www.arduino.cc/en/Main/ArduinoBoardLeonardo.

[5] Raspberry Pi Foundation, *Raspberry Pi 2 Model B*, Zuletzt aufgerufen am 05.05.2015. Adresse: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

[6] ——, *Raspberry Pi FAQs - Frequently Asked Questions*, Zuletzt aufgerufen am 05.05.2015. Adresse: https://www.raspberrypi.org/help/faqs/.

[9] P. Alberts, *Logic Level Bidirectional*, Zuletzt aufgerufen am 01.07.2015. Adresse: http://cdn.sparkfun.com/datasheets/BreakoutBoards/Logic_Level_Bidirectional.pdf.

[10] *RPi Distributions - eLinux.org*, Zuletzt aufgerufen am 12.05.2015. Adresse: http://elinux.org/RPi_Distributions.

[11] A. Casals, *pipaOS - Homepage*, Zuletzt aufgerufen am 12.05.2015. Adresse: http://pipaos.mitako.eu/.

[12] *AVR-GCC-Tutorial/LCD-Ansteuerung*, Zuletzt aufgerufen am 22.08.2015. Adresse: http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/LCD-Ansteuerung.

[13] Atmel Corporation, *AVR182: Zero Cross Detector*, Zuletzt aufgerufen am 02.09.2015. Adresse: http://www.atmel.com/Images/doc2508.pdf.

[14] Arduino LLC, *Arduino - PinMapping32u4*, Zuletzt aufgerufen am 29.04.2015. Adresse: http://www.arduino.cc/en/Hacking/PinMapping32u4.

[15] ——, *Arduino - ArduinoBoardUno*, Zuletzt aufgerufen am 13.05.2015. Adresse: http://www.arduino.cc/en/Main/ArduinoBoardUno.

[16] ——, *Arduino - ArduinoBoardMini*, Zuletzt aufgerufen am 13.05.2015. Adresse: http://www.arduino.cc/en/Main/ArduinoBoardMini.

[17] ——, *Arduino - ArduinoBoardProMini*, Zuletzt aufgerufen am 13.05.2015. Adresse: http://www.arduino.cc/en/Main/ArduinoBoardProMini.

[18] ——, *Arduino - ArduinoBoardMicro*, Zuletzt aufgerufen am 13.05.2015. Adresse: http://www.arduino.cc/en/Main/ArduinoBoardMicro.

[19] ——, *Arduino - ArduinoBoardNano*, Zuletzt aufgerufen am 13.05.2015. Adresse: http://www.arduino.cc/en/Main/ArduinoBoardNano.

[20] V. Urban, *Raspberry Pi als Universalprogrammer*, Zuletzt aufgerufen am 04.07.2015. Adresse: http://www.mikrocontroller.net/articles/Raspberry_Pi_als_Universalprogrammer.

[21] *I2C, SPI, I2S, LIRC, PPS, stopped working?*, Zuletzt aufgerufen am 04.07.2015. Adresse: https://www.raspberrypi.org/forums/viewtopic.php?t=97314.

[22] *Raspberry Pi - View topic - Change I2C speed*, Zuletzt aufgerufen am 03.08.2015. Adresse: https://www.raspberrypi.org/forums/viewtopic.php?f=44&t=34734.

[23] *Adding I2C0 port to Raspberry Pi B Rev 2.0*, Zuletzt aufgerufen am 20.08.2015. Adresse: https://xdevs.com/article/adding-i2c0-port-raspberry-pi-b-rev-20/.

[24] *Raspberry Pi - Install GCC 4.9 and compile C++14 programs | Solarian Programmer*, Zuletzt aufgerufen am 03.08.2015. Adresse: https://solarianprogrammer.com/2015/01/13/raspberry-pi-raspbian-install-gcc-compile-cpp-14-programs/.

[25] *Raspberry Pi | Wiring | Download and Install | Wiring Pi*, Zuletzt aufgerufen am 03.08.2015. Adresse: http://wiringpi.com/download-and-install/.

[26] *Raspberry Pi - View topic - Getting i2c to work on Raspberry Pi2*, Zuletzt aufgerufen am 03.08.2015. Adresse: https://www.raspberrypi.org/forums/viewtopic.php?f=44&t=104133.

[27] *USB-Stick automatisch einbinden (Raspberry Pi)*, Zuletzt aufgerufen am 20.08.2015. Adresse: http://www.elektronik-kompendium.de/sites/raspberry-pi/1911271.htm.

[28] *Raspberry Pi - Autostart von Skripten und Programmen einrichten*, Zuletzt aufgerufen am 20.08.2015. Adresse: http://raspberry.tips/raspberrypi-einsteiger/raspberry-pi-autostart-von-skripten-und-programmen-einrichten/.