Fachbereich 2
Informatik und Ingenieurwissenschaften
Studiengang Informatik

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science in Informatik

Ansteuerung eines LCD-Moduls über die GPIO-Anschlüsse des Raspberry Pi Einplatinencomputers

Vorgelegt von: Baran Altundal
Matrikelnummer: 952129

Referent: Prof. Dr. Christian Baun
Korreferent: Prof. Dr. Thomas Gabel

Eingereicht am: 17.08.2015

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit selbstständig und nur unter Zuhilfenahme der ausgewiesenen Hilfsmittel angefertigt habe. Alle Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter der Angabe der Quellen kenntlich gemacht.

Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Frankfurt am Main, den 19.10.15

Baran Altundal

Danksagung

Ich möchte mich hiermit bei allen Menschen bedanken, die mich bei der Anfertigung meiner Bachelorarbeit unterstützt haben.

Zunächst möchte ich mich ganz besonders bei Herrn Prof. Dr. Christian Baun für die Betreuung, Motivation und Bereitstellung der technischen Mittel bedanken. Ohne seine Unterstützung wäre die Bachelorarbeit in dieser Form nicht zustande gekommen.

Herrn Prof. Dr. Thomas Gabel danke ich vor allem für die Bereitschaft und Übernahme der Zweitkorrektur.

Ein ganz besonderer Dank gilt sowohl meiner Familie für die Unterstützung und den Rückhalt während meiner Studienzeit als auch meinen Freunden und meiner Freundin, die mich jederzeit stets motiviert und moralisch unterstützt haben.

Für die Korrekturlesung danke ich Frau Susanne Hofsäss-Kusche.

Kurzfassung

Die vorliegende Bachelorarbeit beschäftigt sich mit der Ansteuerung eines 16x2 Zeichen-LCD-Moduls über die GPIO-Anschlüsse des Einplatinenrechners Raspberry Pi. Ziel ist es, mithilfe einer Ansteuerungsbibliothek eine Textausgabe auf dem LCD-Modul zu realisieren. Es wird die Funktionsweise zwischen den beiden Komponenten ausgearbeitet und verdeutlicht. Das Ergebnis wird zeigen, dass neben einer einfachen Textausgabe weitaus mehr möglich ist, wie zum Beispiel eine selbstprogrammierte Umrechnung von Dezimalzahlen auf das LCD-Modul darzustellen.

Abstract

This bachelor thesis deals with the control of a 16x2 character LCD display over the GPIO-pins from the single-board-computer Raspberry Pi. The objective is, to realize a text output on the display with using a control library. The functionality between the two components will be clarified in this thesis. At the end you will see that there are much more possibilities beyond a simple text output, such as presenting a self-programmed conversion of decimal numbers on the LCD-Module.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

GPIO:	Generell Purpose Input/Output
LED:	Light-Emitting Diode
LCD:	Liquid-Crystal Display
HDMI:	High Definition Multimedia Interface
SD:	Secure Digital
SSH:	Secure Shell
SHA:	Secure Hash Algorythm
MHz:	Megahertz
LAN:	Local Area Network
GB:	Gigabyte
MB:	Megabyte
Mbit/s:	Megabit pro Sekunde
ms:	Millisekunden
USB:	Universal Serial Bus
SPI:	Serial Peripheral Interface
I2C:	Inter-Integrated Circuit
UART:	Universal Asynchronous Receiver Transmitter
IP:	Internetprotokoll
DHCP:	Dynamic Host Configuration Protocol
ASCII:	American Standard Code for Information Interchange
RS:	Register Select
R/W:	Read/Write
E:	Enable
DB:	Data Bus
DD-RAM:	Display Data – Random Access Memory
CG-ROM:	Character Generator – Read Only Memory
CG-RAM:	Character Generator – Random Access Memory

1 Einleitung

Heutzutage gibt es eine große Auswahl an Einplatinencomputern auf dem Markt. Besonders beliebt ist unter anderem das Raspberry Pi, entwickelt von der britischen Raspberry Pi Foundation. Durch seine Verwendungsmöglichkeiten, dazu den niedrigen Preis und geringen Stromverbrauch ist er besonders für Entwickler und Programmieranfänger interessant. Das Raspberry Pi Modell B+ ist bereits ab 25 € erhältlich[1]. Mit einem Stromverbrauch von 0,072 Kilowattstunden pro Tag entstehen Kosten bei einer einfachen Inbetriebnahme (siehe Kapitel 3.3) für das ganze Jahr in Höhe von ca. 7,36 €, mit dem Durchschnittspreis von ca. 0,28 € pro Kilowattstunde. [1] Auch ausschlaggebend für die große Beliebtheit (siehe Kapitel 2.1) sind die vielfältigen Einsatzmöglichkeiten. So kann das Raspberry Pi als Spielkonsole, Steuereinheit oder auch als Mini-Rechner verwendet werden. [3] [25] Ein Beispiel für die Verwendung als Steuereinheit stellt der Einsatz in der Heimautomation dar. Darunter wird die Steuerung von Haushaltgeräten über einen zentralen Punkt verstanden. Ein Beispiel wäre das einfache Ein- und Ausschalten der Lichter über ein Smartphone oder Tablet.

Eine weitere Verwendungsmöglichkeit und auch das Thema dieser Arbeit betrifft die Ansteuerung eines LCD-Moduls über die GPIO-Pins des Raspberry Pi. Die GPIO-Pins bieten eine Schnittstelle für die Ansteuerung diverser Hardware und ermöglichen die Realisierung interessanter Projekte. So lassen sich beispielsweise Wetterstationen erstellen oder kleine Motoren als auch LED-Leuchten ansteuern. [25] [31]

[1] Quelle: http://www.conrad.de/ce/de/product/1227449/Raspberry-Pi-Model-B-512-MB-ohne-Betriebssystem?ref=searchDetail

Mit Hilfe der Letzteren kann das von Philips patentierte Ambilight[2] nachgebaut werden. Darunter versteht sich die optische Vergrößerung des Fernsehbildschirms mittels LED-Leuchten, welche an die hinteren Ränder des Fernsehgeräts angebracht sind. Diese leuchten in den Farben des momentanen Bildes und erzeugen für den Betrachter somit eine optische Vergrößerung des Bildschirms. Gleichzeitig wird der Übergang zwischen Gerät und Hintergrund flüssiger. Der sich daraus ergebende Effekt besteht darin, dass das Bild für die Augen ruhiger und angenehmer wirkt. LCD-Module finden ihren Platz in Peripheriegeräten wie Druckern, Scannern, Kopierern oder diversen Netzwerkgeräten wie zum Beispiel in Servern. Sie dienen dazu, den Status von Prozessen oder Systeminformationen wie Temperatur oder Rechenleistung anzuzeigen. Oft findet die Verwendung auch im eigenen Computer statt. So hat ein Privatbenutzer ebenfalls einen Überblick hinsichtlich der Auslastung.

Grundlage für eine solche Ansteuerung ist die Kenntnis über die Funktionalität zwischen einem Einplatinencomputer und LCD-Modul. Aus diesem Grund beschäftigt sich diese wissenschaftliche Arbeit mit der Ansteuerung eines LCD-Displays.

[2] Quelle: http://www.philips.de/c-m-so/fernseher/p/ambilight

1.1 Aufbau der Arbeit

Diese Bachelorarbeit ist in vier Kapitel untergliedert. Nachdem das **erste Kapitel** die Einleitung beinhaltet und eine Einführung in die Thematik vermittelt hat, werden im folgenden **zweiten Kapitel** die Grundlagen erläutert. Es werden der Einplatinenrechner Raspberry Pi als auch das LCD-Modul vorgestellt.

Der Fokus richtet sich auf das **dritte Kapitel** dieser Arbeit. In diesem Kapitel wird die Ansteuerung des LCD-Moduls mit den GPIO Pins des Raspberry Pi genauer beschrieben. Hierbei wird zunächst die verwendete Hardware aufgezeigt und auf die Implementierung eingegangen. Anschließend wird die Funktionalität der GPIO-Pins und des LCD-Moduls näher erläutert. Über Anwendungsbeispiele in diesem Kapitel werden die unterschiedlichen Verwendungsmöglichkeiten aufgezeigt.

Das **vierte Kapitel** bildet den inhaltlichen Abschluss dieser Arbeit und enthält eine Zusammenfassung als auch einen Ausblick dieser Thematik.

2 Grundlagen

In diesem Kapitel werden die Grundlagen zum Thema dieser Arbeit beschrieben. Es werden der Einplatinencomputer Raspberry Pi als auch das LCD-Modul vorgestellt und zwei unterschiedliche Ansteuerungsmöglichkeiten untersucht. Für das Verständnis der im Rahmen dieser Arbeit behandelten Thematik werden außerdem grundlegende Begrifflichkeiten definiert.

2.1 Der Raspberry Pi Einplatinencomputer

Der Raspberry Pi wurde von der Raspberry Pi Foundation entwickelt und war erstmals Anfang 2012 in den Märkten erhältlich. Das Ziel der Foundation bestand darin, einen kostengünstigen Rechner zu produzieren, welcher besonders jungen Menschen das Erlernen von Programmiersprachen erleichtern und das Aneignen von Hardwarekenntnissen vereinfachen sollte. [2]

In der vergleichbaren Größe einer Bankkarte besteht die Möglichkeit, durch die zahlreichen Anschlüsse nach Belieben einen „kleinen“ Rechner zusammenzubauen oder mehrere Einplatinencomputer als eine Einheit zusammenzuschließen. Die Flexibilität, die dadurch entsteht, sollte vor allem neue Ideen fördern, aber auch dazu anregen, sich entsprechend dem Konzept „Learning by doing“ durch Ausprobieren neues Wissen anzueignen. So ist es auch möglich, das Raspberry Pi mit verschiedenen Linux-Distributionen als Betriebssystem zu bespielen. Die Distributionen werden in der Regel von SD-Karten gebootet (siehe Kapitel 3.3).

Die Vielfalt der Systeme ist in den letzten Jahren stark angestiegen, wobei diese von zahlreichen verschiedenen Entwicklern stammen. Die Beliebtheit des Raspberry Pi zeigt sich besonders an den Verkaufszahlen. So wurden bis Anfang 2015 mehr als fünf Millionen Geräte verkauft [26] und ebenso das Zubehör und die Erweiterungsmöglichkeiten für diese haben in dieser kurzen Zeit rasant zugenommen. Auch die Anzahl der Bücher, Hilfsprogramme und Communities rund um dieses sind mit der ansteigenden Beliebtheit enorm gewachsen. [2]

Bei dem für diese Arbeit verwendeten Pi handelt es sich um das Raspberry Pi Modell B+ (siehe Abbildung 2.1). Dieses kam Mitte 2014 auf den Markt und erweiterte das bisherige Modell B. So wurde für dieses Modell die Anzahl er GPIO-Pins und der USB-Ports vergrößert. Neben weiteren Leistungsverbesserungen wurde auch der SD-Karten-Slot durch einen Micro-SD-Karten-Slot ausgetauscht.

Das Datenblatt des Raspberry Pi B+ Modell sieht wie folgt aus:

- Prozessor: Broadcom BCM2835
- Arbeitsspeicher: 512 MB
- CPU: ARM1176JZF-S (700 MHz)
- GPU: Broadcom Dual Core VideoCore IV
- LAN: 10/100 Mbit/s
- Gewicht: ca. 45 g
- Maße: 85,6 mm x 56 mm x 20 mm (Länge x Breite x Höhe)
- USB-Anschlüsse: 4 x USB 2.0
- Schnittstellen: 40 GPIO-Pins, I2C, SPI und UART, CSI, DSI, microSD Slot
- Videoausgabe: HDMI, Composite Video
- Audioausgabe: Klinkenstecker 3,5 mm, HDMI
- Stromversorgung: 5 V via Micro-USB Anschluss
- Stromverbrauch: ca. 500 – 600 mA (2,5 – 3 W) [27]

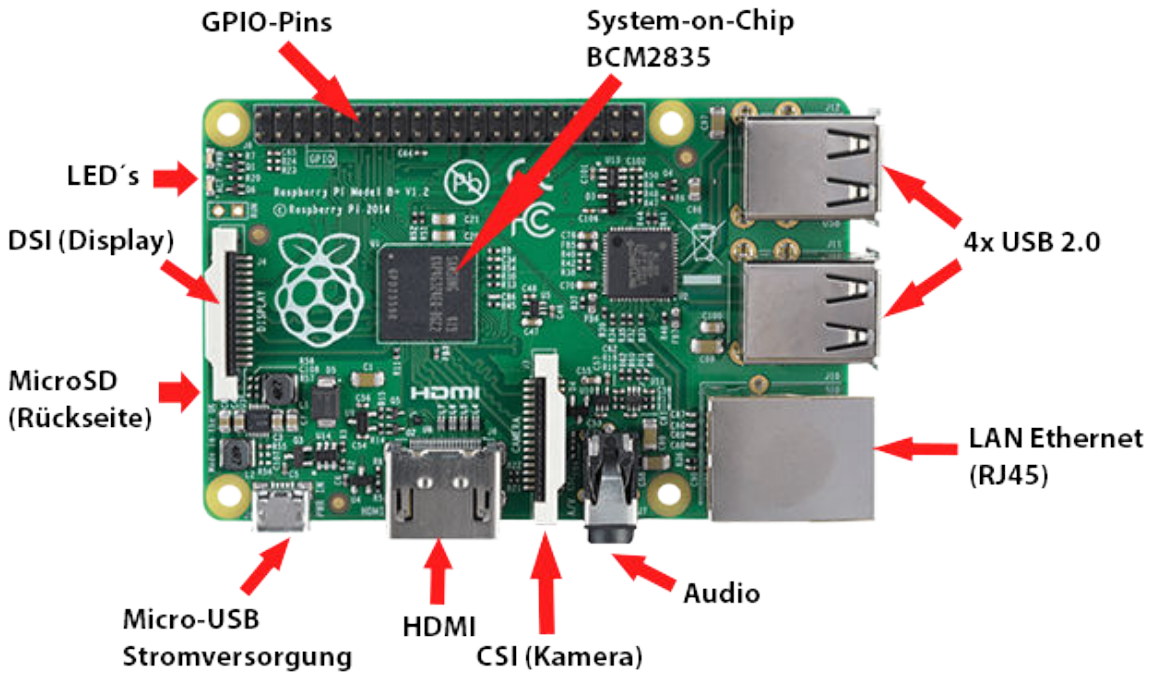


Abbildung 2.1: Raspberry Pi Modell B+ [3]

2.1.1 GPIO – General Purpose Input/Output

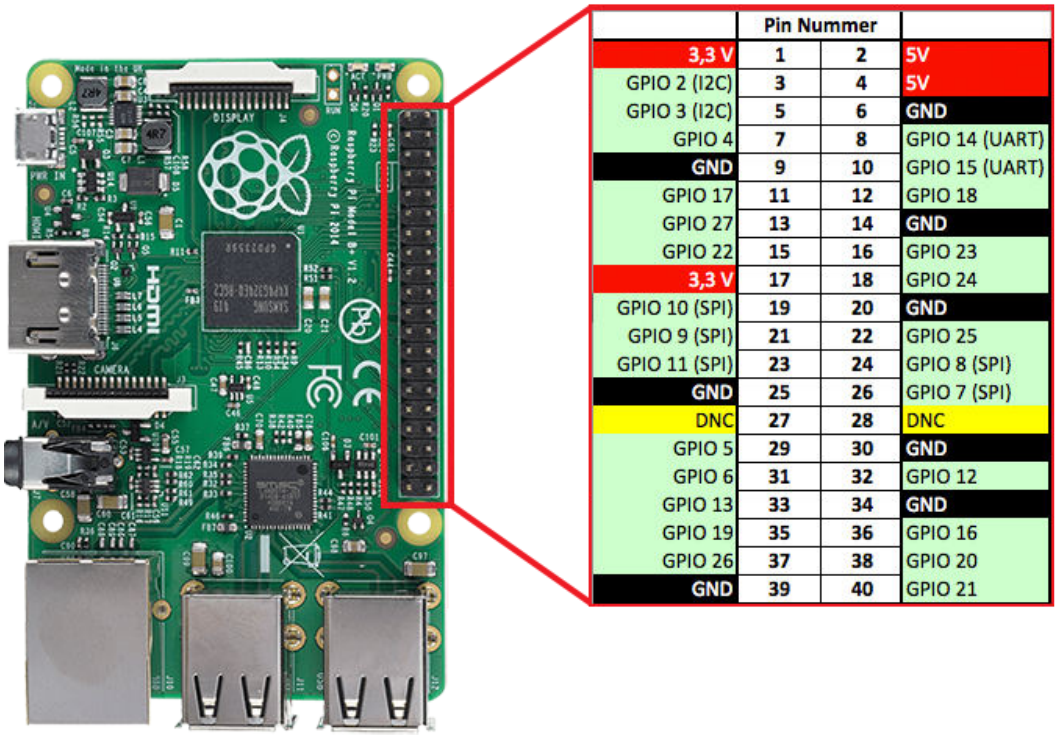
Das Anschließen und Ansteuern von diversen Hardware-Komponenten, wie zum Beispiel Sensoren oder kleinen Motoren, erfolgt hauptsächlich über die Eingabe- und Ausgabe-Pins (*engl.* GPIO – General Purpose Input/Output) des Raspberry Pi. Als frei programmierbare Pins dienen sie als Schnittstelle, um Daten über elektrische Signale zu übertragen. Über den integrierten ARM-Mikroprozessor BCM2835 werden die Signale der Pins zur Verfügung gestellt (siehe Kapitel 3.4). Das Raspberry Pi B+ Modell hat im Vergleich zu seinen Vorgängermodellen einen erweiterten Anschluss von 40 Pins (siehe Abbildung 2.2).

[3] Bildquelle: https://cdn-reichelt.de/bilder/web/xxl_ws/A300/RASPBERRY_PI_B_PLUS_06.png

Die Vorgängermodelle sind mit einer Leiste von 26 Pins ausgestattet. Je nach Modell existieren unterschiedliche Revisionen der Pinbelegung. Die ersten 26 Pins sind bei allen Revisionen identisch belegt. Die physikalische Nummerierung der Pins ist in der Reihenfolge von links nach rechts eingeordnet.

Die 40 GPIO-Pins bestehen aus zwei 3,3-Volt- und 5-Volt-Spannungspins, acht Massepins (0 Volt) und den 26 GPIO-Ein- und -Ausgabepins. Die zwei DNC (Do not Connect) Pins haben keine Funktion, da diese für spätere freischaltbare Funktionen von den Entwicklern freigelassen wurde. Neun der GPIO-Pins verfügen über erweiterte Schnittstellenfunktionen von I2C, SPI und UART (siehe Abbildung 2.2). [31] Äußerste Vorsicht ist bei dem Anschluss höherwertiger Spannungen an die GPIO-Pins geboten, da es zu einer Beschädigung des Einplatinenrechners führen kann. [3] [4]

Die folgende Abbildung veranschaulicht die Belegung der GPIO-Pins:



	Pin Nummer		
3,3 V	1	2	5V
GPIO 2 (I2C)	3	4	5V
GPIO 3 (I2C)	5	6	GND
GPIO 4	7	8	GPIO 14 (UART)
GND	9	10	GPIO 15 (UART)
GPIO 17	11	12	GPIO 18
GPIO 27	13	14	GND
GPIO 22	15	16	GPIO 23
3,3 V	17	18	GPIO 24
GPIO 10 (SPI)	19	20	GND
GPIO 9 (SPI)	21	22	GPIO 25
GPIO 11 (SPI)	23	24	GPIO 8 (SPI)
GND	25	26	GPIO 7 (SPI)
DNC	27	28	DNC
GPIO 5	29	30	GND
GPIO 6	31	32	GPIO 12
GPIO 13	33	34	GND
GPIO 19	35	36	GPIO 16
GPIO 26	37	38	GPIO 20
GND	39	40	GPIO 21

Abbildung 2.2: GPIO-Pinbelegung des Raspberry Pi B+

2.2 LCD-Modul TC1602A-09

Ein LCD-Modul ist eine Flüssigkeitsanzeige und findet überwiegend ihren Einsatz in Peripheriegeräten, wo sich Systeminformationen als ein einfacher Text oder eine Grafik ausgeben lassen, wie zum Beispiel in Scannern, Druckern oder Kopierern. Über einen integrierten Controller-Chip werden die Daten über Signale verarbeitet und ausgegeben.

Der sogenannte HD44780 Controller-Chip von Hitachi hat sich mittlerweile als Standard durchgesetzt und findet sich als solcher oder in Nachbauten mit ähnlicher Funktionsweise in den unterschiedlichsten LCD-Modulen. Der Vorteil hierbei ist, dass sich trotz der vielen unterschiedlichen Modelle die Anschlüsse der Pinbelegung kaum voneinander unterscheiden. Die Datenübertragung erfolgt parallel im 4-Bit- oder 8-Bit-Modus an Mikrocontrollern. Das LCD-Modul TC1602A-09 von Pollin verwendet den SPLC780D1-Controller, welcher dem HD44780-Controller gleichgestellt und in der Größe von 16 Zeichen x 2 Zeilen für 5,95 € zu erwerben ist[4] (siehe Abbildung 2.3). [5]

2.2.1 Pinbelegung

Ein wichtiger Bestandteil für den Schaltungsaufbau ist die Funktion der Pinbelegung. [8] Das LCD-Modul verfügt über einen Anschluss von 16-Pins, der den Industriestandard bei den meisten LCD-Modulen des Hitachi HD44780 bietet (siehe Abbildung 2.3). Dieser besteht aus drei Steuerleitungen, RS (Register Select), R/W (Read/Write) und E (Enable), aus acht Datenleitungen DB0 bis DB7, den Leitungen für die Stromversorgung des Moduls VSS und VDD sowie den Kontakten für die Hintergrundbeleuchtung LED+ und LED- (siehe Tabelle 2.1). Für den Kontakt wurden Steckverbindungen an die Pins gelötet.

[4] Quelle: http://www.pollin.de/shop/dt/Nzc1OTc4OTk-/Bauelemente_Bauteile/Aktive_Bauelemente/Displays/LCD_Modul_TC1602A_09.html

Über die Datenleitungen kann bestimmt werden, ob das Display im 4-Bit- oder 8-Bit-Modus angesteuert wird. Besonders zu beachten sind die Anschlussbelegungen der Stromversorgung VSS und VDD. Diese können je nach Hersteller in umgekehrter Reihenfolge belegt sein und bei falschem Anschluss zu einem Defekt des LCD-Moduls führen. [6] [7]

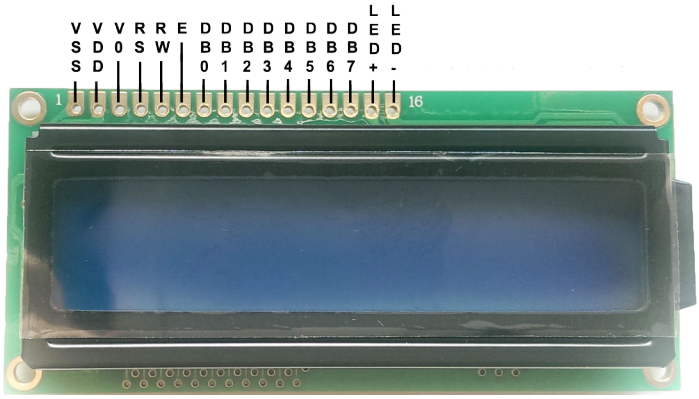


Abbildung 2.3: 16x2 Zeichen LCD-Modul

Tabelle 2.1: LCD-Modul Pinbelegung

Pin	Name	Bedeutung
1	VSS	Masse Pin (GND)
2	VDD	Stromversorgung (+5 Volt)
3	V0	Kontrastspannung
4	RS	Register Select
5	RW	Read/Write
6	E	Enable
7	DB0	Datenleitung für 8-Bit-Ansteuerung
8	DB1	Datenleitung für 8-Bit-Ansteuerung
9	DB2	Datenleitung für 8-Bit-Ansteuerung
10	DB3	Datenleitung für 8-Bit-Ansteuerung
11	DB4	Datenleitung für 4-Bit- und 8-Bit-Ansteuerung
12	DB5	Datenleitung für 4-Bit- und 8-Bit-Ansteuerung
13	DB6	Datenleitung für 4-Bit- und 8-Bit-Ansteuerung
14	DB7	Datenleitung für 4-Bit- und 8-Bit-Ansteuerung
15	LED +	Hintergrundbeleuchtung Pluspol
16	LED -	Hintergrundbeleuchtung Minuspol

Im nachfolgenden Beispiel werden die Funktionalitäten der Pins erläutert:

- **Pin 1** (VSS) und **Pin 2** (VDD): regelt die Stromversorgung des LCD-Moduls.
- **Pin 3** (V0): bestimmt die Kontrastregelung der Zeichen auf dem Display.
- **Pin 4** (Register Select): übermittelt einen Befehl oder ein Zeichen an das Zielregister des LCD-Moduls. Ist RS = 0 (low), werden die Daten als ein Befehl interpretiert und in das Befehlsregister geschrieben. Ist RS = 1 (high), werden die Daten als Zeichen verarbeitet und in den Datenregister geschrieben.
- **Pin 5** (Read/Write): legt den Lese- oder Schreibzugriff fest. Ist R/W = 0 (low), erfolgt ein Schreibzugriff und es können nur Daten an das Display gesendet werden. Bei R/W = 1 (high) wird der Lesezugriff bestimmt, was das Lesen von Informationen aus dem Displaychip erlaubt.
- **Pin 6** (Enable): verarbeitet über steigende und fallende Pegel die Datenbits der Datenleitungen.
- **Pin 7** (DB0) bis **Pin 14** (DB7): sendet die Datenbits im 4-Bit- oder 8-Bit-Modus. Im 4-Bit-Modus werden nur die Leitungen DB4 – DB7 benötigt, der 8-Bit-Modus benutzt alle Leitungen.
- **Pin 15** (LED +) und **Pin 16** (LED -): regelt die Stromversorgung für die Hintergrundbeleuchtung. [6]

2.3 Ansteuerungsmöglichkeiten

Unterschiedliche Programme und Bibliotheken bieten die Möglichkeit, eine Ansteuerung auf dem LCD-Modul zu realisieren. Getestet wurden das kostenlose Programm LCD4Linux und die Bibliothek wiringPi, deren Eigenschaften und Ansteuerungsergebnis im folgenden Abschnitt veranschaulicht werden.

2.3.1 LCD4Linux

LCD4Linux ist ein Programm, welches Informationen aus einem Kernel ausliest und auf einem LCD-Modul ausgibt. Es beinhaltet 47 Treiber für das Ansteuern unterschiedlichster LCD-Module und ist Linux-basiert. Über das Einbinden von Widgets und Plugins in die Konfigurationsdatei können unterschiedliche Anzeigemöglichkeiten als auch Funktionen auf das LCD-Modul ausgegeben werden. Die Konfigurationsdatei lcd4linux.conf befindet sich nach der Installation im /etc Verzeichnis des Systems und bildet das Layout von LCD4Linux. [9]

Der Tabelle 2.2 zeigt eine Beispielkonfiguration von der lcd4linux.conf Datei [10] für das Anzeigen der CPU-Last auf einem HD44780 Standard-basierten LCD-Modul:

Tabelle 2.2: Beispielkonfiguration von lcd4Linux

```
1  #generic HD44780 display (LCD4Linux wiring)
2  Display HD44780-generic {
3     Driver 'HD44780'
4     Model 'generic'
5     Port '/dev/parports/0'
6     Size '16x2'
7     Wire {
8      RW      'GND'
9      RS      'AUTOFD'
10     ENABLE  'STROBE'
11     ENABLE2 'GND'
12     GPO     'INIT'
13      }
14 }
15
16
```

```
17  Widget CPU {
18      class      'Text'
19      expression  uname('machine')
20      prefix     'CPU'
21      width       9
22      align      'L'
23      update      tick
24  }
25
26
27  Layout Default {
28      Row1 {
29          Col1 'CPU'
30      }
31  }
32
33  Variables {
34     tick 500
35  }
36
37  Display 'HD44780-generic
38  Layout  'Default'
```

Zum Zeitpunkt dieser Abschlussarbeit ist eine Ansteuerung mit LCD4Linux als eine parallele Schnittstelle über die GPIO-Pins des Raspberry Pi nicht realisierbar. Es ist kein Treiber vorhanden, der die Zustände der GPIO-Pins über einen parallelen Port als Eingabe oder Ausgabe bestimmen kann.

In Zeile 5 der Konfigurationsdatei ist eine Definition des Ports notwendig:

```
Port '/dev/parports/0'    // Zugriff auf Parallele Schnittstelle
```

Da die GPIO-Pins nicht über einen Port bestimmt sind und dadurch keine parallele Schnittstelle bieten, ist eine Ansteuerung in der Art nicht umsetzbar. Jedoch wäre es denkbar, einen Treiber zu entwickeln, der eine Ansteuerung möglich macht. [9] Die Alternative, ein LCD-Modul mit LCD4Linux über die GPIO-Pins des Raspberry Pi anzusteuern, ist mit einem I2C-Bus möglich. Ein I2C-Bus wird an die Pins des LCD-Moduls angebracht und dient als eine serielle Schnittstelle für die Chip-zu-Chip-Kommunikation.

2.3.2 wiringPi

wiringPi ist eine Bibliothek, die eine Schnittstellte für die Ansteuerung der GPIO-Pins bildet. Sie wurde in der Programmiersprache C verwirklicht und dient dem Standard des BCM2835-Prozessors. Viele Projekte lassen sich über wiringPi verwirklichen, so zum Beispiel auch das Schalten von LED-Leuchten über die GPIO-Pins. Für die Ansteuerung eines LCD-Moduls mit dem Raspberry Pi sind die Dateien lcd.h, lcd.c, wiringPi.h und wiringPi.c in der Bibliothek von wiringPi zuständig. Sie ermöglichen das Steuern des LCD-Controllers und bestimmen den Ein- und Ausgangszustand der GPIO-Pins über die Registerzugriffe. [11]

Aufgrund von Änderungen der Hardware-Revisionen und Pin-Bezeichnungen des Raspberry Pi hat der Entwickler von wiringPi ein einheitliches Pin-Belegungsschema entwickelt, sodass Programme durch eine einmalige Deklaration weiterhin nach Änderungen der Bord-Revision funktionieren und nicht ständig geändert werden müssen. [12]

Die Tabelle 2.3 zeigt die LCD-Initialisierung und verwendbare Funktionen [11] im Schema von wiringPi:

Tabelle 2.3: Beispielkonfiguration von wiringPi

```
1   #include <lcd.h>
2   #include <wiringPi.h>
3
4   int main(){
5
6   /* Zugriff auf das wiringPi System, um die Pinbelegung
7   Initialisieren zu können */
8   wiringPiSetupGpio();
9
10
11  // LCD-Initialisierung und Bestimmung der Pinbelegungen
12 int  lcdInit (int rows, int cols, int bits, int rs, int strb,
13 int d0, int d1, int d2, int d3, int d4, int d5, int d6,int d7);
14
15  /* Einige Beispiel Funktionen: */
16
17  //Löschen der Anzeige auf dem 19 Display
18  lcdDisplay (int fd, int state);
19
20  //Ein- oder Ausschalten des 21 Eingabezeigers
21  lcdCursor (int fd, int state);
22
23  //Blinken des 23 Eingabezeigers Ein- oder Ausschalten
24  lcdCursorBlink (int fd, int state);
25
26  //Position des 26 Eingabezeigers festlegen (x=Spalte, y=Zeile)
27  lcdPosition (int handle, int x, int y);
28
29  // Zeichenausgabe 29 auf dem LCD-Modul
30  lcdPutchar (int handle, unsigned char data);
31  }
```

Die Ansteuerung des LCD-Moduls mittels wiringPi war erfolgreich. Die nächsten Kapitel beschreiben die Funktionsweise der Ansteuerung und Datenverarbeitung des LCD-Controllers mit den GPIO-Pins.

3 Ansteuerung und Realisierung

Dieses Kapitel beschreibt den Aufbau und die Ansteuerung des LCD-Displays mit dem Raspberry Pi. Es wird zunächst auf die Anforderung eingegangen und die Konfiguration des Raspberry Pi als auch die Installation der wiringPi Bibliothek werden erläutert. Anschließend wird die Funktionalität der Komponenten miteinander verdeutlicht.

3.1 Anforderungen

Für die Realisierung der Ansteuerung, sind außer dem Display und Raspberry Pi weitere Hard- und Software-Komponenten benötigt. Über Steckverbindungen, auch Jumper-Kabel genannt, ist das LCD-Modul mit den GPIO-Pins des Raspberry Pi angeschlossen. Diese können sich jedoch in der äußeren Endung unterscheiden. Es ist zu beachten, dass der Anschluss der jeweiligen Pins richtig zugeordnet ist, da hier sonst die Funktionalität auszuschließen ist und eine fehlgeleitete elektrische Spannung im schlimmsten Fall zu einer Beschädigung der Hardware führt. [5]

Ein Steckbrett sorgt als elektrischer Leiter zusätzlich für die Übersichtlichkeit der Steckverbindungen. Das Steckbrett dient als eine einfache Überbrückung der Verbindungen zwischen dem Raspberry Pi und dem LCD-Modul. Es bietet den Vorteil, die Steckverbindungen durch einfaches Umstecken jederzeit zu ändern. Zusätzliche Bauelemente, wie zum Beispiel Widerstände, können an das Steckbrett angebracht werden, um dadurch den elektrischen Strom auf einen angemessenen Wert anzupassen.

Über den Anschluss eines LAN- oder Crossoverkabels (RJ45) zwischen dem Notebook und Raspberry Pi ist es mit einer Secure Shell- (SSH) oder Remotedesktopverbindung möglich, die Grafikausgabe direkt auf den Bildschirm zu übertragen als auch den Internetzugriff freizugeben. Weitere Peripheriegeräte wie Maus, Tastatur und HDMI-Kabel sind für eine externe Grafikausgabe auf einem Monitor oder Fernseher nicht mehr notwendig. Somit werden Störungen durch lästige Kabel vermieden und für zusätzliche Ordnung gesorgt. Das Raspberry Pi ist über eine 16-GB-microSD-Speicherkarte gebootet, die vorher mit dem kostenlosen Linux-Betriebssystem Raspbian „Wheezy“ von Debian[5] installiert wurde. Die Stromversorgung der Geräte erfolgt über ein Mikro-USB-Kabel (siehe Abbildung 3.1).

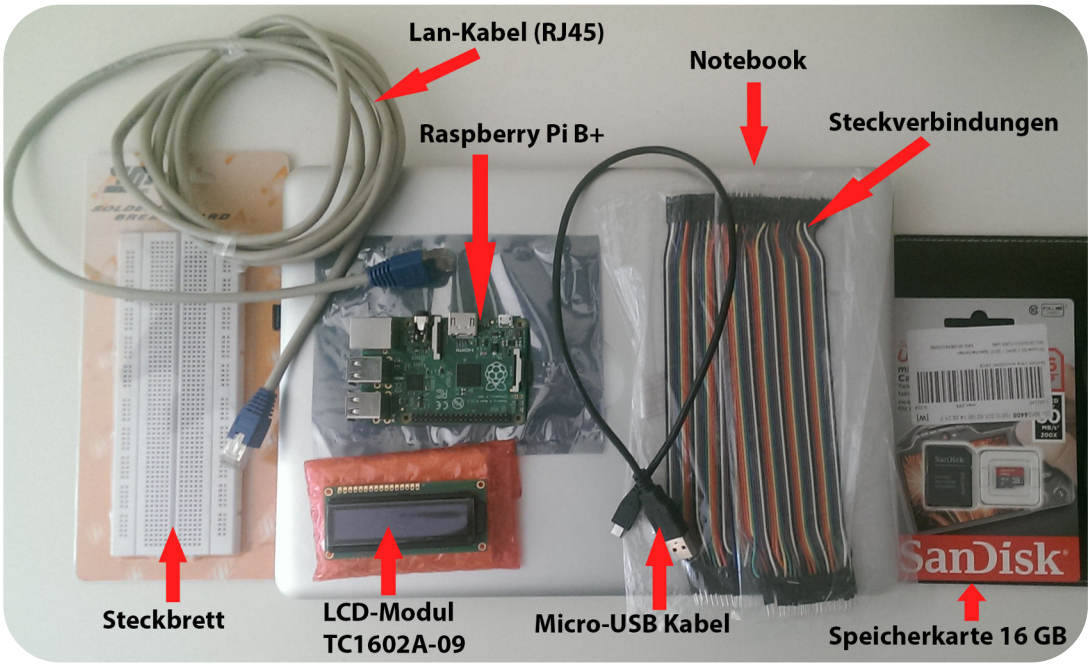


Abbildung 3.1: Verwendete Hardware

[5] Quelle: https://www.raspberrypi.org/downloads/

3.2 Schaltungsaufbau

Da die Ansteuerung im 4-Bit-Modus erfolgt, werden nicht alle Datenleitungen mit dem LCD-Modul verbunden. Es langt der Anschluss von vier Datenleitungen der Pins DB4 bis DB7. Der Vorteil liegt hierbei darin, dass nur ein Anschluss von vier statt acht Datenleitungen benötigt wird und dadurch weitere Anschlüsse erspart bleiben (siehe Abbildung 3.2). Im 8-Bit-Modus werden die Kommandos als ganzes Byte übertragen, wodurch hingegen im 4-Bit-Modus die doppelte Anzahl an Zugriffen erfolgt und somit ein halbes Byte (Nibble) zweimal übertragen wird. Die Daten werden im 8-Bit-Modus einige Millisekunden schneller übertragen, sind jedoch für das menschliche Auge nicht erkennbar. [13]

Tabelle 3.1 stellt die Verbindungsübersicht bei einer Ansteuerung im 4-Bit-Modus zwischen dem LCD-Modul und den GPIO-Pins dar.

Tabelle 3.1: LCD-Modul-/GPIO-Pins-Verbindungsübersicht

LCD Pin:	Verbunden mit:	Funktion:
1 - VSS	GND	Versorgungsspannung 0 Volt
2 - VDD	+5 Volt	LCD Stromversorgung +5 Volt
3 - V0	GND	Kontrastspannung der LCD-Anzeige
4 - RS	GPIO16	Übermittlung eines Befehls oder Zeichens über GPIO-Pin Nummer 36
5 - RW	GND	LCD-Funktion als Schreibzugriff bestimmt, da der LCD-Pin 5 auf (low) Masse gelegt ist
6 - E	GPIO12	Taktleitung über GPIO-Pin Nummer 32
7 - DB0	-	Keine Verwendung für 4-Bit-Modus
8 - DB1	-	Keine Verwendung für 4-Bit-Modus
9 - DB2	-	Keine Verwendung für 4-Bit-Modus
10 - DB3	-	Keine Verwendung für 4-Bit-Modus

11 - DB4	GPIO25	Datenübertragung zwischen LCD-Pin 11 und GPIO-Pin Nummer 22
12 - DB5	GPIO24	Datenübertragung zwischen LCD-Pin 12 und GPIO-Pin Nummer 18
13 - DB6	GPIO23	Datenübertragung zwischen LCD-Pin 12 und GPIO-Pin Nummer 16
14 - DB7	GPIO18	Datenübertragung zwischen LCD-Pin 14 und GPIO-Pin Nummer 12
15 - LED +	+5 Volt	LCD Hintergrundbeleuchtung
16 - LED -	GND	Masse

Die folgenden Abbildungen 3.2 und 3.3 stellen den Schaltungsaufbau und die Verbindungsübersicht der Anschlüsse von Tabelle 3.1 grafisch dar:

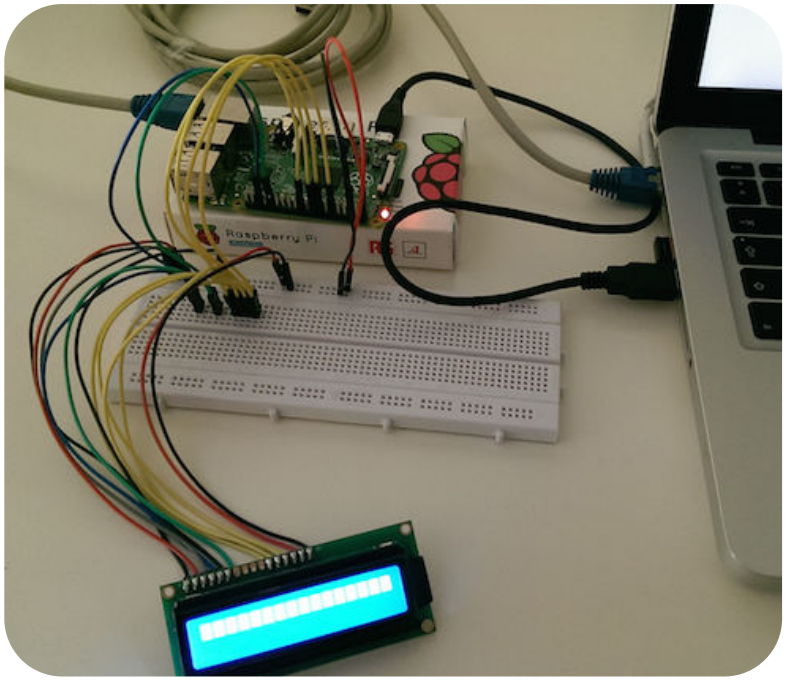


Abbildung 3.2: Schaltungsaufbau

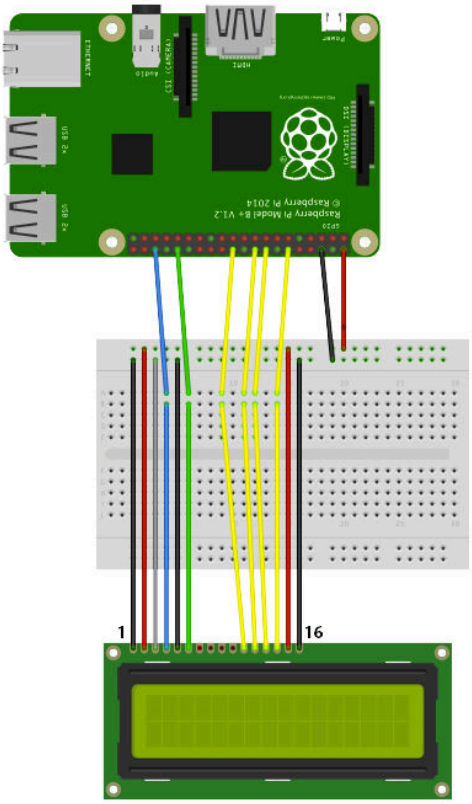


Abbildung 3.3: Übersicht der Schaltung als Fritzing-Diagramm[6]

3.3 Raspberry Pi Konfiguration

Bevor das Raspberry Pi in Betrieb genommen werden kann, muss das Betriebssystem auf die Speicherkarte installiert werden. Bevorzugt wird das kostenlose Linux-Betriebssystem Raspbian „Wheezy“, da dieses vom ARM-Prozessor BCM2835 unterstützt wird. [4]

Raspbian zählt derzeit aufgrund der schnellen und stabilen Performance zu den beliebtesten Linux-Distributionen von Debian. Zudem wird es aufgrund täglicher Softwareupdates durchgehend auf dem aktuellsten Stand gehalten.

[6] Quelle: http://fritzing.org/download/

Die heruntergeladene Debian-Abbild Datei hat eine Größe von 990 MB und muss vorerst aus dem Archiv entpackt werden. Es ist zu empfehlen, die Datei zusätzlich auf Sicherheit gegen Manipulation zu prüfen. Hierzu wird das Terminalfenster unter Linux geöffnet und in das Verzeichnis gewechselt, indem sich die Datei befindet. Das Kommando `$ openssl sha1 2015-05-05-wheezy-raspbian.zip` zeigt die SHA-1-Prüfsumme an, welche mit der angegebenen Prüfsumme auf der Downloadseite abgeglichen werden kann. [14]

Als Nächstes steht die Formatierung der 16-GB-Speicherkarte im FAT32-Format unter dem Apple-Betriebssystem Mac OS X an. Hierzu gilt es, zunächst die Bezeichnung der Speicherkarte herauszufinden. Alle Informationen über das System können unter Mac OS X unter „Systembericht“ entnommen werden, darunter auch die Bezeichnung der angeschlossenen Speicherkarte. Es ist zu erkennen, dass unter dem Reiter „Kartenleser” im Eintrag „BSD-Name: /dev/disk1s2“ die Bezeichnung der Speicherkarte zu finden ist. Die Partition wird vorerst über das Terminal durch `$ sudo umount /dev/disk1s2` deaktiviert, jedoch nicht ausgeworfen.

Nun kann das Abbild von Raspbian über das Kommando `$ sudo dd bs=1m if=2015-05-05-wheezy-raspbian.img of=/dev/disk1s2` auf die microSD-Karte kopiert werden.
Sudo ist ein Befehl, der mehr Rechte als der Standardbenutzer bezogen auf das System ermöglicht. Das „dd“[7] steht für dump device und kopiert die Daten blockweise auf das Speichermedium. Das bs = 1 m bestimmt die Blockgröße beim Kopiervorgang. 1 M entspricht 1048576 Byte (1024*1024). Die Bezeichnung „if“ und „of“ steht für input file und output file.

Der erste Startvorgang kann mit dem Einführen der Speicherkarte in das Raspberry Pi und dem Anschluss der Stromversorgung mit dem mikro-USB-Kabel beginnen.

[7] Quelle: https://wiki.ubuntuusers.de/dd

Um eine Verbindung mit dem Raspberry Pi über SSH herzustellen, ist es notwendig, die IP-Adresse herauszufinden. Damit der Raspberry Pi eine IP-Adresse zugewiesen bekommt, ist es unter dem Betriebssystem Mac OS X wichtig, die Internetfreigabe für die Ethernet-Verbindung zu aktivieren. Die Einstellung der Netzwerkverbindung erlaubt es, den Internetzugriff über die LAN-Schnittstelle gemeinsam zu nutzen. Standardmäßig ist das Raspberry Pi so konfiguriert, dass eine IP-Adresse automatisch aus dem DHCP-Server des Routers zugwiesen wird. Über das Kommando `$ ifconfig` lassen sich die IP-Adressen im Netzwerk anzeigen (siehe Abbildung 3.3).

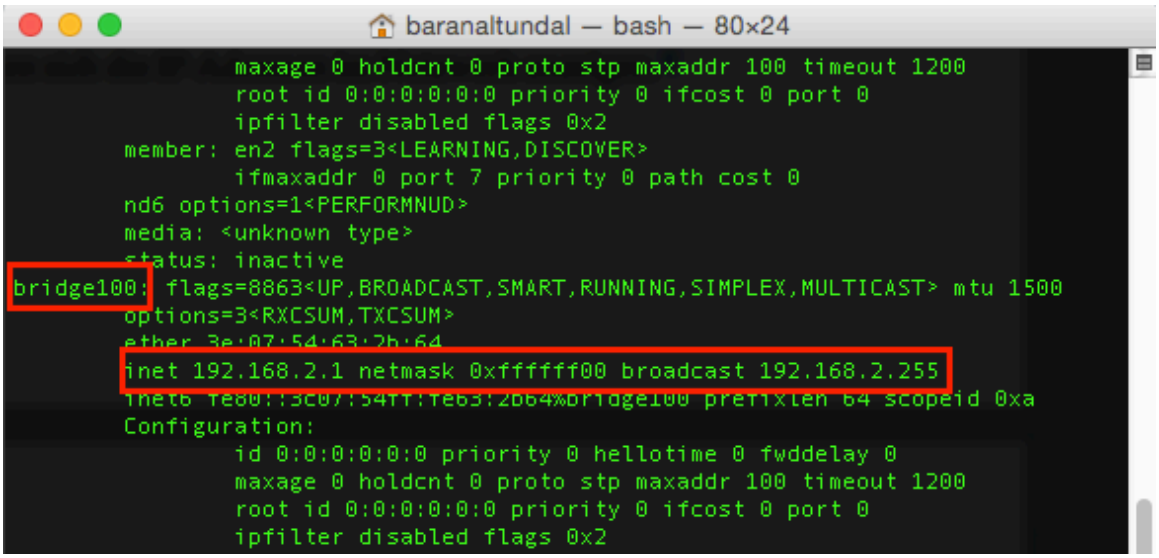


Abbildung 3.4: Anzeigen der IP-Adressen im Netzwerk

Die LAN-Bezeichnung bridge100 zeigt die im Netzwerk über Ethernet freigegebenen IP-Adressen an. Die IP-Adresse des Raspberry Pi befindet sich im Bereich 192.168.2.1 bis 192.168.2.255. Um die genaue IP-Adresse zu ermitteln, kann dieser Bereich durch das Kommando `$ nmap 192.168.2.0/24` durchsucht werden. Nmap steht für network mapper und ist ein Scanner, wodurch sich IP-Adressen und offene Ports in einem Netzwerk ermitteln lassen (siehe Abbildung 3.4).

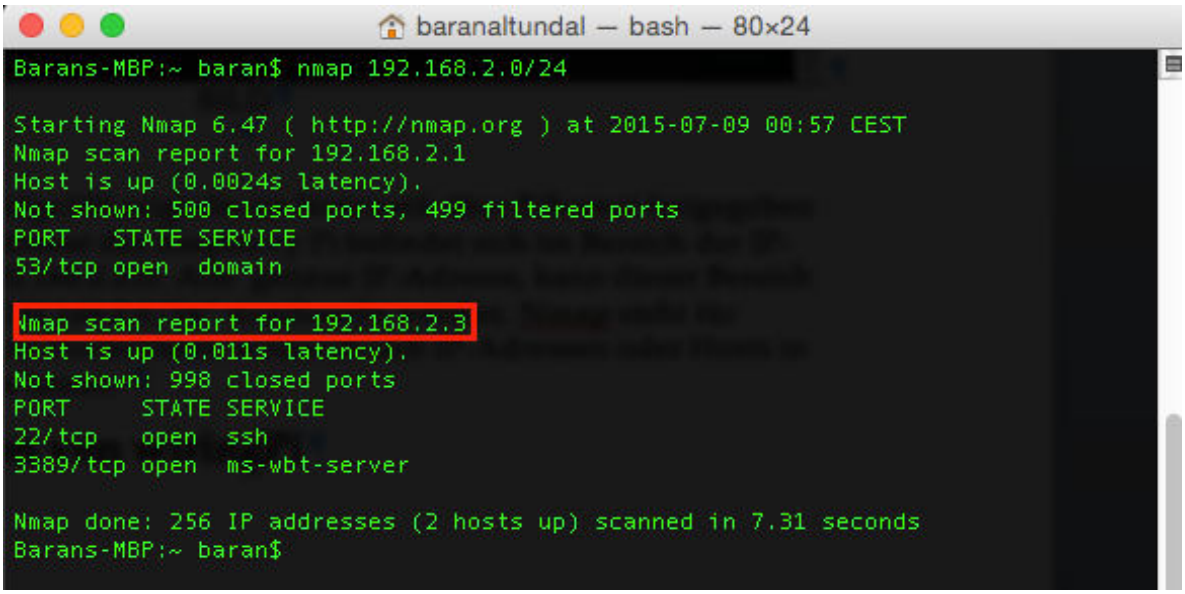


Abbildung 3.5: Ermitteln der IP-Adresse des Raspberry Pi über Nmap

Die IP-Adresse ist ermittelt und der Verbindungsaufbau über SSH kann mit dem Kommando `$ ssh pi@192.168.2.3` erfolgen. Es schließt sich die Passwortabfrage an, die standardgemäß „raspberry“ lautet.

Vor der Inbetriebnahme von Raspbian sind über die SSH-Verbindung einige Konfigurationen des Systems vorzunehmen. Die Übersicht des Tools lässt sich mit `$ sudo raspi-config` öffnen. Ein blaues Fenster erscheint mit den vorzunehmenden Einstellungen (siehe Abbildung 3.6).

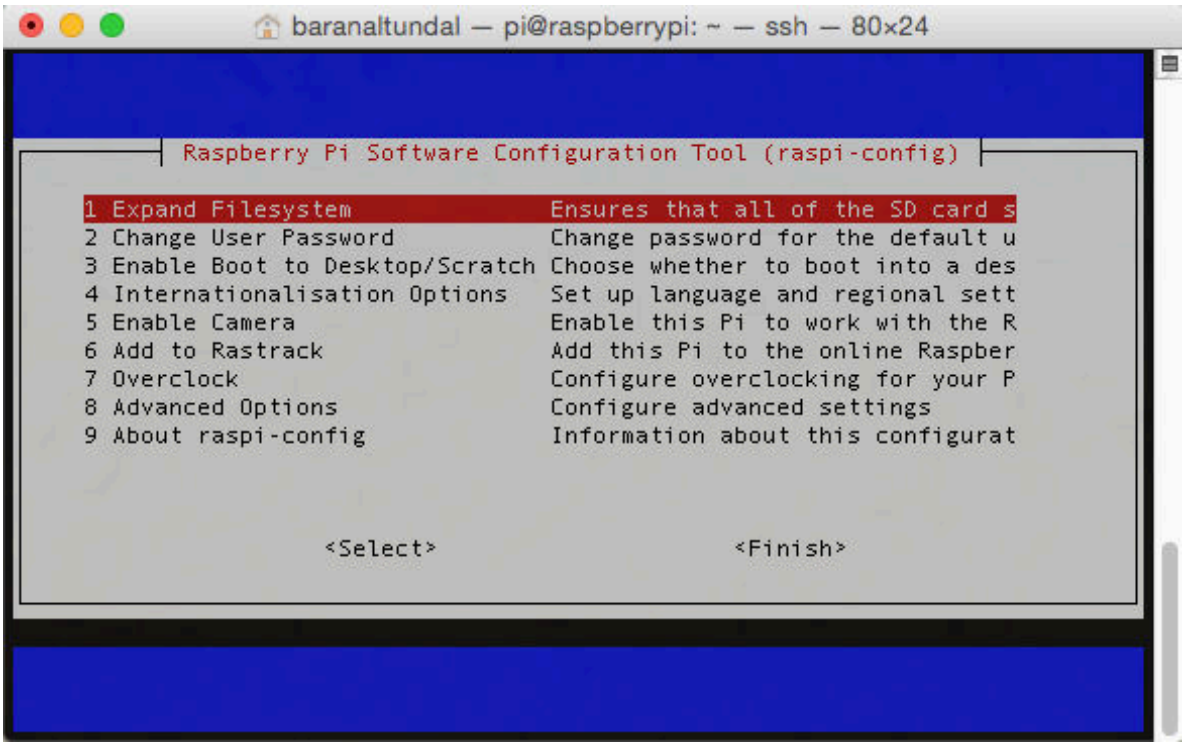


Abbildung 3.6: Raspberry Pi Konfiguration raspi-config

1. Expand Filesystem: das Betriebssystem Raspbian auf dem restlichen Speicher der Speicherkarte erweitern.

2. Change User Password: das Standardpasswort „raspberry“ umändern.

4. Internationalisation Options: Einstellen der Europäischen Zeitzone und deutschen Tastaturbelegung.

8. Advanced Options: Änderung des Hostnamen, alternativ SSH Ein- oder Ausschalten und Aktualisierung des Tools „raspi-config“.

Nach Abschluss der Konfiguration, ist ein Neustart des Systems notwendig und anschließend eine Aktualisierung des Betriebssystems vorzunehmen. Die Aktualisierung erfolgt über das Kommando `$ sudo apt-get update` und `sudo apt-get upgrade`.

Die grafische Oberfläche lässt sich mit einem beliebigen Remote-Desktop-Programm anzeigen. Es gilt, vorerst das Paket „xrdp“ auf dem Raspberry Pi zu installieren. Dieses Paket stellt die Nutzung eines Servers für eine Remotedesktopverbindung her und ist über das Kommando `$ sudo apt-get install xrdp` auf Raspbian zu installieren. Die Installation von Paketen verdeutlicht schrittweise durch die Füllung von Inhalt den Aufbau des Systems. Nach einem Eintrag der IP-Adresse sowie Benutzername und Passwort in das Remote-Desktop-Programm kann die Anzeige der grafischen Oberfläche starten (siehe Abbildung 3.7).

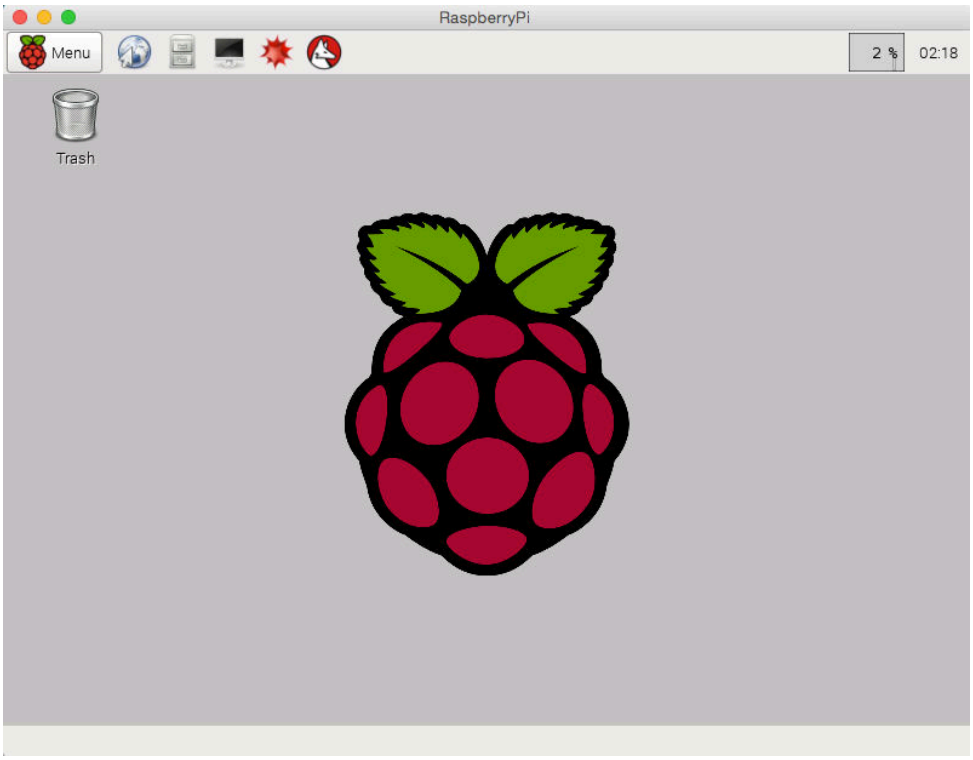


Abbildung 3.7: Remotedesktopverbindung

3.3.1 Installation von wiringPi

Die Installation von wiringPi ist über die Kommandozeile in Raspbian durchzuführen. Die Bibliotheken sind im Archiv des Linux-Versionsverwaltungssystem GitHub[8] von Git zum Herunterladen verfügbar. Git ist ein Management-System in Linux für Entwickler, die gemeinsam an kostenlosen Software-Projekten arbeiten und dadurch eine Übersicht zu Änderungen und Bearbeitungsstand erhalten. Der Befehl in der Kommandozeile `$ sudo apt-get install git-core` installiert Git auf Raspbian. Nach der Installation ist das Kommando `$ git clone git://git.drogon.net/wiringPi` für das Herunterladen von wiringPi zuständig. Anschließend ist in das Verzeichnis von wiringPi zu wechseln und in der Kommandozeile mit `$ git pull origin` die Bibliothek zu aktualisieren und über `$ ./build` zu installieren. [15]

3.3.1.1 Zeichenausgabe

Der folgende, in C geschriebene Quellcode zeigt, wie eine Textausgabe mittels wiringPi umgesetzt werden kann (siehe Tabelle 3.2). Hier erscheint als Textausgabe in der ersten Zeile des 16x2-Zeichen-LCD-Moduls das Wort „**Hallo**“ und in der zweiten Zeile das Wort „**Welt!**“ (siehe Abbildung 3.9).

[8] Quelle: https://github.com/

Tabelle 3.2: Beispielprogramm für die Zeichenausgabe

```
1  #include <lcd.h>
2  #include <wiringPi.h>
3
4   int main() {
5
6   int lcd;
7   wiringPiSetupGpio();
8
9   lcd = lcdInit(2, 16, 4, 16, 12, 25, 24, 23, 18, 0, 0, 0, 0);
10
11  lcdPosition(lcd,0,0);
12  lcdPuts(lcd, "Hallo");
13
14  lcdPosition(lcd,0,1);
15  lcdPuts(lcd, "Welt!");
16
17 }
```

Zeile 1: Einbinden der Headerdatei lcd.h, welche die Funktionen der Datei lcd.c deklariert.

Zeile 2: Einbinden der Headerdatei wiringPi.h, welche die Funktionen der Datei wiringPi.c deklariert.

Zeile 7: Die Funktion wiringPiSetupGpio() initialisiert die Pinbelegung und bestimmt über den Registerzugriff des BCM2835-Prozessors den Zustand eines GPIO-Pins.

Zeile 9: *lcdInit* deklariert die Pinbelegung im Schema von wiringPi. Es beinhaltet die Initialisierung des 16x2-Zeichen-LCD-Moduls, die Ansteuerung im 4-Bit-Modus und die Belegung der GPIO-Pins über die Daten- und Steuerleitungen des LCD-Moduls.

Eine Übersichtstabelle der Pinbelegung lässt sich mit dem Kommando `$ gpio readall` anzeigen (siehe Abbildung 3.8).

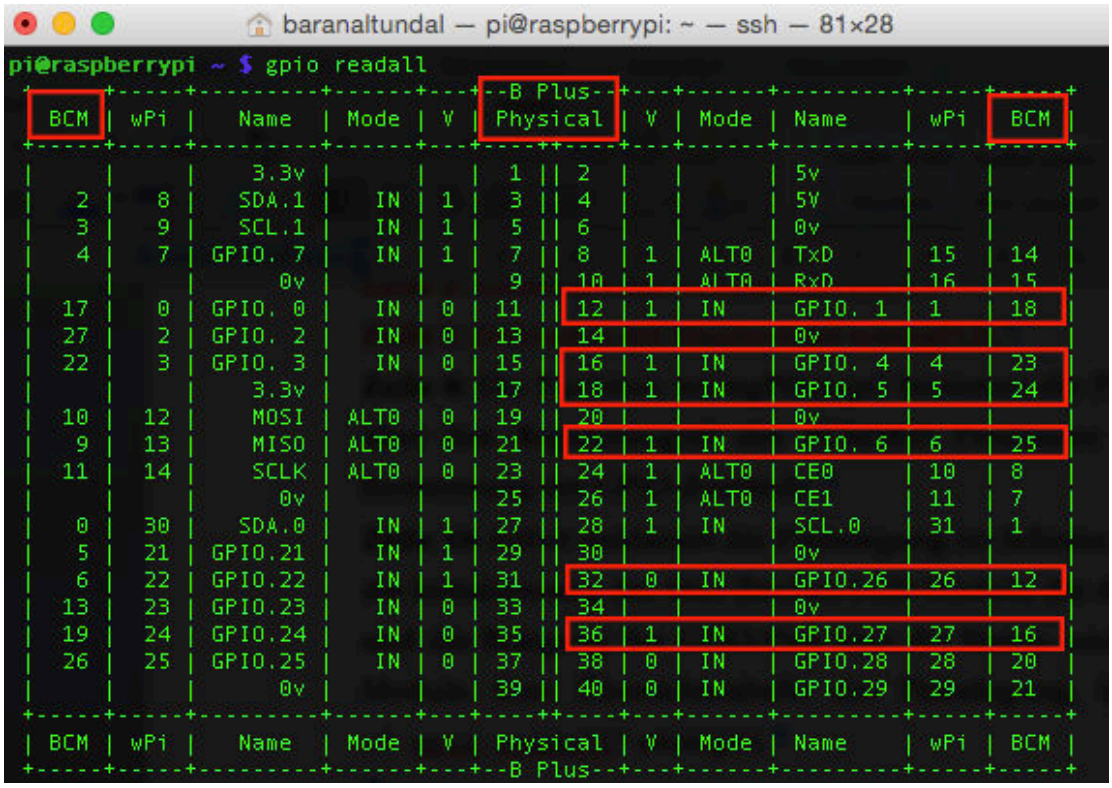


Abbildung 3.8: wiringPi Pinbelegung gpio readall

Wie in Abbildung 3.8 zu erkennen, sind die Bezeichnungen der GPIO-Pins unter dem BCM-Reiter aufgeführt, was ebenfalls dem BCM-Standard entspricht. Für die Initialisierung ist somit der physikalische Pin 12 als GPIO18 im BCM-Standard deklariert (siehe Abbildung 2.2).

Zeile 11 und 14: Die Funktion *lcdPosition()*, setzt die Position der Textausgabe im DD-RAM-Speicher auf dem Display fest. Der erste Zahlenwert in der Funktion bestimmt die Zeilenposition, der zweite Zahlenwert die Position der Spalte.
Zeile 12 und 15: Über die Funktion *lcdPuts()*, lassen sich Zeichenketten oder ein einzelnes ASCII-Zeichen auf dem Display anzeigen.

Für die Textausgabe ist es notwendig, das Programm mit dem Kommando `$ gcc -o text text.c -lwiringPi -lwiringPiDev` in eine ausführbare Datei umzuwandeln und über das Kommando `$ sudo ./text` zu starten.

Abbildung 3.9: Ausgabe von "Hallo Welt!"

3.4 ARM-Mikroprozessor BCM2835

Die Realisierung einer Zeichenausgabe auf dem LCD-Modul ist über einen ganzen Prozessablauf bestimmt. Um Befehle über die Steuer- und Datenleitungen an den LCD-Controller zu senden, ist vorerst der Ausgangszustand der GPIO-Pins zu bestimmen. Diese Aufgabe übernimmt der ARM-Mikroprozessor[9] BCM2835 der Firma Broadcom, der ausschließlich für den Raspberry Pi entworfen wurde. Seine besondere Eigenschaft ist die Integration der Komponenten von CPU, RAM und GPU gleichzeitig.

ARM-Mikroprozessoren finden heutzutage ihren Platz in eingebetteten Systemen wie zum Beispiel in Smartphones, Tablets, Routern oder diversen Netzwerkgeräten. Sie sind klein, leistungsfähig und haben einen geringen Energiebedarf. [4]

[9] Quelle: http://www.arm.com/products/processors/

Die weltweit größten Soft- und Hardware-Hersteller-Unternehmen wie Apple und Samsung verwenden die Chips ebenfalls in ihren Produkten, wobei Samsung auch für die Anfertigung der Chips zuständig ist. [16]

3.4.1 Zugriff und Ansteuerung der GPIO-Pins

Die Ansteuerung eines GPIO-Pins wird über den Zugriff auf Registern bestimmt.
Der BCM2835 hat 54 GPIO-Pins und 41 Register. Diese beinhalten sechs 32 Bit große Funktionsauswahlregister, das sogenannte GPIO Alternate function select register (GPFSELn), in dem die 54 GPIO-Pins verteilt sind. Die ersten fünf Register decken jeweils zehn Pins ab sowie das letzte Register vier Pins. Die Register werden mit dem Namen GPIO Alternate function select register 0 bis GPIO Alternate function select register 5 bezeichnet. [17]

Eine Datenübertragung an das LCD-Modul geschieht über den Ausgangszustand eines GPIO-Pins. Die Bestimmung des Ausgangszustands erfolgt über den Zugriff und die Beschriftung des physikalischen Speichers. Dieser Prozess entsteht durch die Übersetzung der 32-Bit-Bus-Adresse in eine physikalische Adresse.
In Linux verschafft das Verzeichnis „/dev/mem“ den Zugriff auf die physikalische Adresse, welches das Abbild des Hauptspeichers im Kernel bildet. [17] [18] [19] [31]

Folgende Informationen sind im BCM2835-Benutzerhandbuch über die Adressen auf Seite 6 zu lesen:

„Physical addresses range from 0x20000000 to 0x20FFFFFF for peripherals. The busaddresses for peripherals are set up to map onto the peripheral bus address range starting at 0x7E000000. Thus a peripheral advertised here at bus address 0x7Ennnnnn is available at physical address 0x20nnnnnn.“ [17]

Das folgende Zitat besagt, dass die physikalische Adresse für Peripheriegeräte im Adressbereich von 0x20000000 bis 0x20FFFFFF liegt. Die Bus-Adressen für Peripheriegeräte beginnen im Adressbereich 0x7E000000. Somit ist die Bus-Adresse 0x7Ennnnnn für Peripheriegeräte über die physikalische Adresse 0x20nnnnnn erreichbar. Das n steht für den Bereich einer bestimmten Adresse.

Bit(s)	Field Name	Description	Type	Reset
31-30	---	Reserved	R	0
29-27	FSEL19	FSEL19 - Function Select 19 000 = GPIO Pin 19 is an input 001 = GPIO Pin 19 is an output 100 = GPIO Pin 19 takes alternate function 0 101 = GPIO Pin 19 takes alternate function 1 110 = GPIO Pin 19 takes alternate function 2 111 = GPIO Pin 19 takes alternate function 3 011 = GPIO Pin 19 takes alternate function 4 010 = GPIO Pin 19 takes alternate function 5	R/W	0
26-24	FSEL18	FSEL18 - Function Select 18	R/W	0
23-21	FSEL17	FSEL17 - Function Select 17	R/W	0
20-18	FSEL16	FSEL16 - Function Select 16	R/W	0
17-15	FSEL15	FSEL15 - Function Select 15	R/W	0
14-12	FSEL14	FSEL14 - Function Select 14	R/W	0
11-9	FSEL13	FSEL13 - Function Select 13	R/W	0
8-6	FSEL12	FSEL12 - Function Select 12	R/W	0
5-3	FSEL11	FSEL11 - Function Select 11	R/W	0
2-0	FSEL10	FSEL10 - Function Select 10	R/W	0

Abbildung 3.10: BCM2835 GPIO Alternate function select register 1[10]

Wie in Abbildung 3.10 zu erkennen, steht der Name FSEL10-19 für die Bezeichnung der GPIO-Pins. Jeder dieser GPIO-Pins wird über 3 Bit für acht mögliche Funktionen bestimmt, wovon zwei den Ein- oder Ausgangszustand definieren.

[10] Bildquelle: http://www.farnell.com/datasheets/1521578.pdf

Die restlichen sechs Alternativen sind potenziell für die GPIO-Pins bestimmt, welche über weitere Funktionen wie SPI, UART, I2C verfügen oder Signale über Pulsweitenmodulationen erzeugen. [31] Das Bitmuster 000 bestimmt den Eingangszustand eines GPIO-Pins als auch das Bitmuster 001 den Ausgangszustand. [17]

Die Pins GPIO12 und GPIO16 sind über die Steuerleitungen RS und E am LCD-Modul angeschlossen (siehe Tabelle 3.1). Um diese in den Ausgangszustand zu setzen, muss das Bitmuster 001 an die entsprechenden Stellen des Registers geschoben werden.

GPIO12 (FSEL12) belegt im 32 Bit Alternate function select register 1 die Bits 6 – 8. GPIO16 (FSEL16) belegt im 32 Bit Alternate function select register 1 die Bits 18 – 20 (siehe Abbildung 3.10).

Somit ergibt sich im Alternate function select register 1 folgendes Muster:

32 - 00000000000**001**000000000000**001**00000 - 0

Die Abbildung 3.11 zeigt, dass das Register GPFSEL1 die Bus-Adresse 0x7E200004 vorweist. Somit wird die physikalische Adresse 0x20200004 beschrieben und die Pins GPIO12 und GPIO16 werden als Ausgangszustand gesetzt. [20]

Address	Field Name	Description	Size	Read/ Write
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0004	GPFSEL1	GPIO Function Select 1	32	R/W
0x 7E20 0008	GPFSEL2	GPIO Function Select 2	32	R/W
0x 7E20 000C	GPFSEL3	GPIO Function Select 3	32	R/W
0x 7E20 0010	GPFSEL4	GPIO Function Select 4	32	R/W
0x 7E20 0014	GPFSEL5	GPIO Function Select 5	32	R/W

Abbildung 3.11: GPIO-Registerübersicht[11]

3.5 LCD-Controller

Der integrierte Controllerchip am LCD-Modul interpretiert die gesendeten Befehle der GPIO-Pins und gibt sie auf dem Display aus. Von der Datenverarbeitung bis hin zur Adressierung gehören diese zu seinen Aufgaben. In den folgenden Abschnitten wird die Funktionsweise einer Zeichenverarbeitung im 4-Bit-Modus auf dem LCD-Modul beschrieben.

3.5.1 Übertragungsprozess

Die Funktionen der wiringPi-Dateien lcd.c und lcd.h ermöglichen die Datenübertragung über die GPIO-Pins an das LCD-Modul. Durch steigende und fallende Flanken signalisieren die Steuer- und Datenleitungen Daten, die der LCD-Controller verarbeitet und am Display ausgibt. Hierbei muss der zeitliche Ablauf der Signalübertragung zwischen den Leitungen korrekt abgestimmt sein. [21] Die Abbildung 3.12 zeigt den Prozess einer Zeichenübertragung im Schreibmodus.

[11] Bildquelle: http://www.farnell.com/datasheets/1521578.pdf

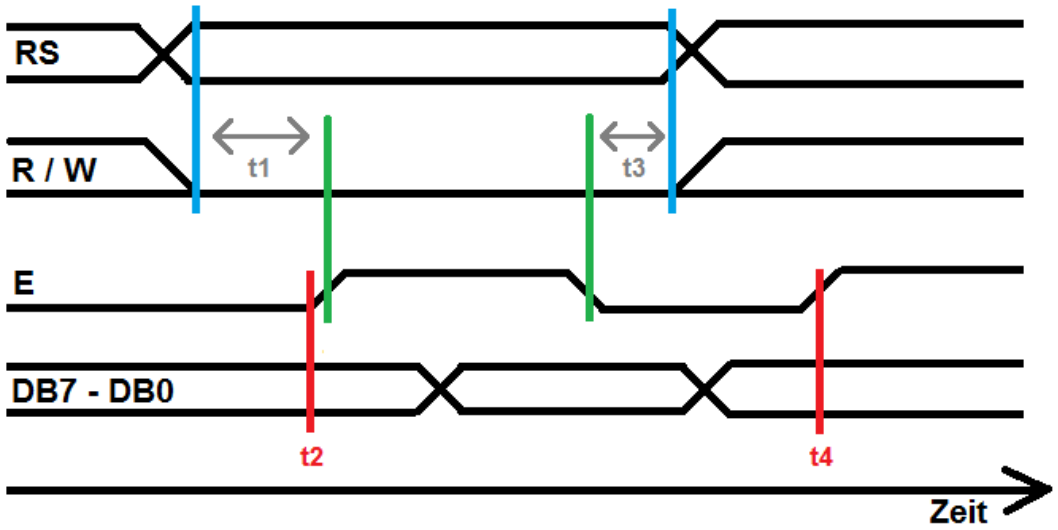


Abbildung 3.12: LCD-Controller-Datenübertragungsprozess

Vor Beginn des Schreibprozesses werden die Zustände von RS und R/W bestimmt. Im Moment des high Pegels von Enable zwischen t1 und t3 liest der LCD-Controller den aktuellen Status der beiden Steuerleitungen ein. Die Steuerleitung RS signalisiert im Zustand 0, ob die ankommenden Daten als Befehl in das Befehlsregister oder im Zustand 1 als Zeichen in das Datenregister geschrieben werden. Da ein Schreibzugriff erfolgt, befindet sich der Status von R/W dauerhaft auf 0. Während des steigenden und fallenden Pegels von Enable zwischen t2 und t4 gelingt es dem Display, die Daten von der Datenleitung DB0 – DB7 einzulesen und auszugeben.

Die Informationen über den zeitlichen Ablauf befinden sich in den Produktdatenblättern [21], welche jedoch vom Programmierer unterschiedlich definiert werden kann.

3.5.2 LCD-Register

Die beiden 8-Bit-Befehls- und -Datenregister im LCD-Controller sind für Anweisungen und Zeichenausgaben auf dem Display zuständig. Das Befehlsregister steuert Befehle, wie das Display zu löschen oder die Cursorposition zu bestimmen, und speichert Adressinformationen vom DD-RAM und CG-RAM. Das Datenregister ist für das Speichern und Anzeigen der darzustellenden Zeichen im DD-RAM und CG-RAM auf dem Display zuständig. Der DD-RAM hat einen Datenspeicher von 80 Bytes und somit Platz für 80 Zeichen. Die Länge des LCD-Moduls ist meist kürzer als der Adressbereich, dadurch ist eine vollständige Anzeige nicht möglich. So liegt bei einem 16x2-Zeichen-LCD-Modul der Adressbereich in der ersten Zeile bei 00 bis 0F und in der zweiten Zeile bei 40 bis 4F. Jedes Zeichen ist in Form einer 5x8-Punktematrix dargestellt und wird automatisch vom integrierten Adresszähler in die nachfolgenden Zeilen geschrieben (siehe Abbildung 3.13). [22]

	Zeile 1															
1	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
	Zeile 2															
2	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

16 16

Abbildung 3.13: DD-RAM-Adresse in Hexadezimal

Der CG-ROM im LCD-Controller beinhaltet einen festen ASCII-Zeichensatz von 192 Zeichen. Enthalten sind 160 Zeichen in einer 5x8-Punktematrix und 32 Zeichen in einer 5x10-Punktematrix. Zusätzlich dient ein CG-RAM als Zeichengenerator, worüber acht benutzerdefinierte Zeichen in einer 5x8-Punktematrix und vier Zeichen in einer 5x10-Punktematrix konfigurierbar sind. [21]

Der Zeichensatz besteht aus den arabischen Ziffern, dem lateinischen Alphabet mit Klein- und Großbuchstaben, Umlauten, Satzzeichen und Symbolen sowie einigen japanischen Katakana-Symbolen. Jedes Zeichen hat eine Größe von 8 Bit und wird aus einem oberen und unteren Nibble zusammengesetzt (siehe Abbildung 3.14). [23]

Abbildung 3.14: CG-ROM-Zeichensatz[12]

[12] Bildquelle: http://www.mikrocontroller.net/attachment/166031/LCD-charset.gif

3.5.3 Zeichenausgabe

Der LCD-Controller empfängt jedes Zeichen über Signale der Datenleitungen als 8 Bit großen Zeichencode und übersetzt sie für die Ausgabe auf dem Display. Vor dem Schreibzugriff findet ein Initialisierungsprozess statt. [21] Dieser Prozess bereitet das Display auf den gewünschten Betriebsmodus vor. Währenddessen ist es nicht möglich, das Display anzusteuern. Nach dem Einschalten benötigt das Display ca. 15 ms, um die Betriebsspannung von 4,5 Volt zu erreichen.

Im folgenden Beispiel ist der Prozess der Initialisierung des Displays ausführlich dargestellt:

R/W	**RS**	**DB7 - DB0**	**Beschreibung**
0	0	0 0 1 1 – * * * *	Function Set: 8-Bit-Betrieb, wird 3x wiederholt.
0	0	0 0 1 0 – * * * *	Function Set: Einstellen auf 4-Bit-Betrieb.
0	0	0 0 1 0 – 0 0 0 0	Funktion Set: 4-Bit, 2 Zeilen, 5*8-Rastergröße.
0	0	1 0 0 0 – 0 0 0 0	Zweiter Nibble
0	0	0 0 0 0 – 0 0 0 0	Display On/Off: Anzeige einschalten.
0	0	1 0 0 0 – 0 0 0 0	Zweiter Nibble
0	0	0 0 0 0 – 0 0 0 0	Display clear: Datenspeicher löschen und Cursorposition auf die erste Adresse setzen.
0	0	0 0 0 1 – 0 0 0 0	Zweiter Nibble
0	0	0 0 0 0 – 0 0 0 0	Entry Mode Set: Cursor zählt die Adresse automatisch beim Schreibvorgang hoch.
0	0	0 1 1 0 – 0 0 0 0	Zweiter Nibble [21] [24]

Im folgenden Beispiel ist der Prozess der Textausgabe „Hallo Welt!“ auf dem Display ausführlich dargestellt:

R/W	**RS**	**DB7 - DB0**	**Beschreibung**
0	1	0 1 0 0 – 0 0 0 0	H
0	1	1 0 0 0 – 0 0 0 0	Zweiter Nibble
0	1	0 1 1 0 – 0 0 0 0	a
0	1	0 0 0 1 – 0 0 0 0	Zweiter Nibble
0	1	0 1 1 0 – 0 0 0 0	l
0	1	1 1 0 0 – 0 0 0 0	Zweiter Nibble
0	1	0 1 1 0 – 0 0 0 0	l
0	1	1 1 0 0 – 0 0 0 0	Zweiter Nibble
0	1	0 1 1 0 – 0 0 0 0	o
0	1	1 1 1 1 – 0 0 0 0	Zweiter Nibble
0	1	0 1 0 1 – 0 0 0 0	W
0	1	0 1 1 1 – 0 0 0 0	Zweiter Nibble
0	1	0 1 0 1 – 0 0 0 0	e
0	1	1 1 1 1 – 0 0 0 0	Zweiter Nibble
0	1	0 1 1 0 – 0 0 0 0	l
0	1	1 1 0 0 – 0 0 0 0	Zweiter Nibble
0	1	0 1 1 1 – 0 0 0 0	t
0	1	0 1 0 0 – 0 0 0 0	Zweiter Nibble
0	1	0 0 1 0 – 0 0 0 0	!
0	1	0 0 0 1 – 0 0 0 0	Zweiter Nibble [24]

3.6 Anwendungsbeispiele

Die Anwendungsbeispiele verdeutlichen, dass es außer einer einfachen Textausgabe weitaus mehr Möglichkeiten gibt, etwas auf dem Display auszugeben. Das erste Beispiel zeigt einen Umrechner, der eine Dezimalzahl in eine Binärzahl umwandelt und ausgibt. Das zweite Beispiel gibt aufeinanderfolgend alle 256 Zeichen der ASCII-Tabelle aus.

3.6.1 Dezimal-binär-Umrechner

Ein Computer rechnet und verarbeitet Informationen im Binärsystem. Der folgende Quellcode ermöglicht die Umrechnung einer Zahl aus dem Dezimalsystem in das Binärsystem (siehe Tabelle 3.3). Dabei wird ein Algorithmus verwendet, welcher mittels einer Division über das Restergebnis der Dezimalzahlen in eine Binärzahl umwandelt. Dividiert wird bei der Umwandlung in eine Binärzahl immer durch zwei. So sieht zum Beispiel der Algorithmus für die Umrechnung der Dezimalzahl 10 wie folgt aus:

$10 \div 2 = 5$ Rest 0
$5 \div 2 = 2$ Rest 1
$2 \div 2 = 1$ Rest 0
$1 \div 2 = 0$ Rest 1

Aus der Folge der Reste ergibt sich die Binärzahl für die 10. Nach diesem Algorithmus würde die Binärzahl 0101 die Dezimalzahl 10 darstellen. Dies entspricht jedoch nicht der korrekten Norm, da der Rest in der Little-Endian-Darstellung aufgeführt ist. Üblicherweise werden Binärzahlen als Big-Endian im Computerspeicher verarbeitet, bei dem das höchstwertige Bit an erster Stelle steht.

Für die Umwandlung aus der Little-Endian-Darstellung in die Big-Endian ist noch eine Umkehrung der Zeichenfolge nötig, sodass die Dezimalzahl 10 auch binär 1010 entspricht (siehe Abbildung 3.15).

Die Funktion für die Umrechnung der Datei umrechner.c ist in der Programmiersprache C wie folgt umgesetzt: [28] [29]

Tabelle 3.3: Anwendungsbeispiel 1, Dezimal-binär-Umrechner

```
1 #include <lcd.h>
2 #include <wiringPi.h>
3 #include <stdio.h>
4
5 char stringBinaryNumber[16];
6 char invertedstringBinaryNumber[16];
7 int integerRest;
8 char charRest;
9
10 void convertDecimalIntoBinary (int DecimalNumber) {
11   int i=0;
12   int j=0;
13   int k=0
14   do{
15        integerRest=DecimalNumber % 2;
16
17        if (integerRest==1){charRest='1';}
18        else {charRest='0';}
19        stringBinaryNumber[i]=charRest;
20        DecimalNumber=DecimalNumber/2;
21        i++;
22    }while (DecimalNumber>0);
23
24    for(j=i-1; j>=0; j--){
```

```
25        invertedstringBinaryNumber[k]=stringBinaryNumber[j];
26        k++;
27    }
28 }
29
30 int main()
31 {
32     convertDecimalIntoBinary (10);
33
34  int lcd;
35  wiringPiSetupGpio();
36  lcd = lcdInit(2, 16, 4, 16, 12, 25, 24, 23, 18, 0, 0, 0, 0);
37
38    lcdPosition(lcd,0,0);
39    lcdPuts(lcd,"Dec:10");
40
41    lcdPosition(lcd,0,1);
42    lcdPuts(lcd,"Bin=");
43
44    lcdPosition(lcd,4,1);
45    lcdPuts(lcd,("%s",invertedstringBinaryNumber));
46 }
```

Zeile 3: #include <stdio.h> bindet die Standardbibliothek der Programmiersprache C ein.

Zeile 5: Deklaration eines Arrays für die umgerechnete Binärzahl in der Größe von 16 Zeichen.

Zeile 6: Deklaration eines Arrays für die Umkehrung der Zeichenfolge in der Größe von 16 Zeichen.

Zeile 14 bis 22: Mittels der fußgesteuerten do-while-Schleife lässt sich die Binärzahl ermitteln. Dabei wird für jeden Durchlauf eine Modulo-2-Operation auf die Dezimalzahl, dargestellt durch die Variable „DecimalNumber“, durchgeführt.

Die Variable „integerRest“ speichert den ermittelten Rest. Im oben gezeigten Beispiel entspricht das dem Rest der Division durch zwei. Um die Binärzahl in der richtigen Darstellung anzuzeigen, ist es erforderlich, diese als String auszugeben. Hierfür ist die if-Abfrage zuständig, welche für 0 bzw. 1 das entsprechende character der Variablen „charRest“ zuweist. Dieses character wird dann an die Stelle i in das Array „stringBinaryNumber“ geschrieben.

Daraufhin folgen die Teilung und Überschreibung der Dezimalzahl „DecimalNumber“, sodass für den nächsten Schleifendurchlauf die entsprechende Division durchgeführt werden kann. Abbruchbedingung für die Schleife ist, dass „DecimalNumber“ kleiner oder gleich 0 ist. Nach Beendigung des Schleifendurchlaufs ist die Dezimalzahl als Little-Endian-Darstellung in eine Binärzahl umgewandelt.

Zeile 24 bis 27: Die for-Schleife ist für die Umwandlung in die Big-Endian-Darstellung zuständig, welche das Array „stringBinaryNumber“, in dem die Little-Endian-Darstellung enthalten ist, und das Array „stringinvertedBinaryNumber“, in welchem die Big-Endian enthalten sein soll, gegenläufig durchläuft. Dabei wird die Laufvariable j von der maximalen Arraygröße aus dekrementiert und k von 0 aus inkrementiert. Somit positioniert sich der letzte Eintrag aus „stringBinaryNumber“ an die erste Stelle von „stringinvertedBinaryNumber“ und die vorletzte an die zweite Stelle, solange bis in „stringBinaryNumber“ noch ein Zeichen enthalten ist. Das Ergebnis entspricht nun der Dezimalzahl.

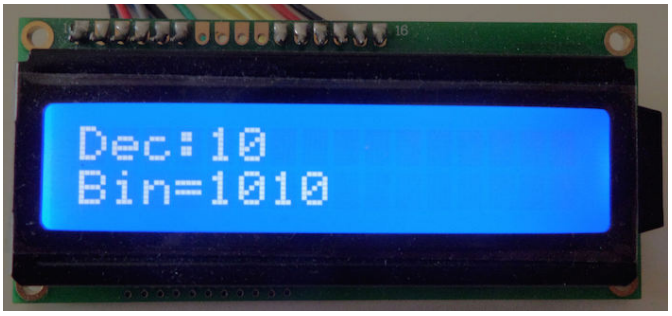


Abbildung 3.15: Ausgabe der Dezimal-binär-Umrechnung

3.6.2 ASCII-Zeichenausgabe

Das folgende Beispiel in Tabelle 3.4 stellt alle ASCII-Zeichen vom CG-ROM auf dem Display dar (siehe Abbildung 3.14 und Abbildung 3.16). [28] [30]

Tabelle 3.4: Anwendungsbeispiel 2, ASCII-Zeichenausgabe

```
1 #include <lcd.h>
2 #include <wiringPi.h>
3
4 int main()
5 {
6    int lcd;
7    int c;
8
9    wiringPiSetupGpio();
10   lcd = lcdInit(2, 16, 4, 16, 12, 25, 24, 23, 18, 0, 0, 0, 0);
11
12   for(c=' '; c<256; c++){
13   lcdPosition(lcd,c,0);
14   lcdPutchar(lcd,c);
15   delay(100);
16   }
17
18 }
```

Zeile 12 bis 16: Für die Zeichenausgabe der ASCII-Tabelle ist eine for-Schleife notwendig. Sie dient dazu, die Anzahl der Wiederholungen einer Anweisung kontrolliert zu bestimmen. Der Schleifenkopf besteht aus einer Initialisierung, Bedingung und einem Inkrement/Dekrement Operator.

Die Zuweisung der Variablen c deutet darauf hin, mit dem Wert des Leerzeichens aus der ASCII-Tabelle zu beginnen. Um alle Zeichen der ASCII-Tabelle auszugeben, wird der Wert der Variablen c bis zum letzten Dezimalwert inkrementiert. Das heißt, dass die Schleife bei dem letzten ASCII-Zeichen mit dem Dezimalwert 256 aufhört. Gleichzeitig erfolgt über eine Variablenzuweisung in der Funktion lcdPosition(lcd, c, 0) die Zeichenausgabe im Durchlauf von links nach rechts über die Zeilen des LCD-Moduls. Die Darstellung der Zeichen ermöglicht die Funktion lcdPutchar(lcd, c). Die Funktion delay(100) bestimmt die Schnelligkeit der Programmausführung in Millisekunden, in diesem Fall beträgt die Ausführungszeit 100 Millisekunden.

Abbildung 3.16: ASCII-Zeichenausgabe

4 Zusammenfassung

Die vorliegende Arbeit befasste sich mit der Ansteuerung eines LCD-Moduls mit den GPIO-Pins des Einplatinencomputers Raspberry Pi. Für die Realisierung einer Ansteuerung wurden zwei unterschiedliche Programme getestet.

Im ersten Ansteuerungsversuch über LCD4Linux wurde deutlich, dass aufgrund des nicht vorhandenen Treibers umfangreiche Kenntnisse im Bereich der Hardware-Entwicklung vorausgesetzt werden. Für die Treiberprogrammierung ist es erforderlich, die Standardtechniken des Kernels zu kennen. Das Gegenteil wird im zweiten Ansteuerungsversuch mit wiringPi bewiesen. Über Grundkenntnisse in der Programmiersprache C lässt sich eine Textausgabe auf dem LCD-Modul realisieren. Den Grundbaustein bilden die Funktionen der wiringPi-Bibliotheken. Diese ermöglichen es, auf das Speichersystem zuzugreifen, dadurch den Ausgangszustand eines GPIO-Pins zu bestimmen und Daten über Signale an den LCD-Controller zu senden.

Zuvor mussten die Steckverbindungen an die Pins des LCD-Moduls gelötet werden, damit diese über ein Steckbrett mit den GPIO-Pins verbunden werden können. Der Vorteil eines Steckbretts besteht darin, dass die Schaltungen nachträglich verändert werden können und eine Übersicht der Anschlüsse geboten wird. Außerdem ist es weitaus umfangreicher möglich, Projekte unterschiedlichster Art einer Ansteuerung durchzuführen, wie zum Beispiel das Steuern von LED-Leuchten.

Anhand der Anwendungsbeispiele wurden neben einer einfachen Textausgabe unterschiedliche Einsatzmöglichkeiten aufgezeigt. Die Erweiterungen haben es ermöglicht, Dezimalzahlen in Binärzahlen umzurechnen und auf dem Display darzustellen als auch alle Zeichen der ASCII-Tabelle auszugeben. Das Raspberry Pi bietet sowohl für Entwickler als auch Einsteiger einen Einblick in die Funktionalität und Kommunikation der Komponenten eines Mikrocontrollers. So lassen sich durch geringe Anschaffungskosten große Projekte verwirklichen. Die Gesamtkosten des Raspberry Pi inklusive Zubehör für die Ansteuerung eines LCD-Moduls liegen bei rund 50 €.

Abschließend lässt sich sagen, dass die Ansteuerung eines LCD-Moduls über die GPIO-Pins des Raspberry Pi Einplatinencomputers erfolgreich war und eine Anleitung über den Aufbau als auch die Funktionsweise der Datenverarbeitung vermittelt wurde.

4.1 Ausblick

Da es mit der LCD4Linux-Software nicht möglich ist, ein LCD-Modul über die GPIO-Pins des Raspberry Pi anzusteuern, könnte dies in Zukunft ein entwickelter Treiber verwirklichen. So würden sich Informationen aus dem Kernel greifen und dadurch die Prozessorauslastung, Netzwerkauslastung oder IP-Adresse auf dem LCD-Modul darstellen lassen. Zudem ist es weitaus stärker möglich, spannende Projekte auf Basis der Ansteuerung mit dem Raspberry Pi durchzuführen.

5 Literaturverzeichnis

[1] Raspberry.tips. *Raspberry Pi Stromverbrauch - Modellvergleich,*
http://raspberry.tips/faq/raspberry-pi-stromverbrauch-modellvergleich/,
letzter Abruf am 10.08.2015.

[2] Maik Schmidt. *Raspberry Pi, Einstieg-Optimierung-Projekte,*
1. Auflage 2013, dpunkt.Verlag GmbH, Heidelberg, 2013.

[3] Christian Immler. *Linux mit Raspberry Pi,*
2. Auflage, Franzis Verlag GmbH, Haar bei München, 2014.

[4] Klaus Dembowski. *Raspberry Pi - Das technische Handbuch*,
2. erweiterte und überarbeitete Auflage, Springer Vieweg, Wiesbaden, 2015.

[5] Brendan Horan. *Practical Raspberry Pi,*
Springer Verlag, New York, 2013.

[6] Microcontroller.net. *AVR-GCC-Tutorial/LCD-Ansteuerung,*
http://www.mikrocontroller.net/articles/AVR-GCC-Tutorial/LCD-Ansteuerung,
letzter Abruf am 10.08.2015.

[7] Sebastián Barschkis. *Ansteuerung eines LCD*,
http://www6.in.tum.de/pub/Main/TeachingWs2014ProseminarMicrocontrollerEmbedded/Ansteuerung_eines_LCD.pdf,
letzter Abruf am 10.08.2015.

[8] Sprut.de. *Dot-Matrix LCD-Displays,*
http://www.sprut.de/electronic/lcd/index.htm#stecker,
letzter Abruf am 10.08.2015.

[9] lcd4Linux. *lcd4Linux-Tutorial*,
https://lcd4linux.bulix.org/wiki/Howto,
letzter Abruf am 11.08.2015.

[10] lcd4Linux. *How to write new display drivers,*
https://lcd4linux.bulix.org/wiki/driver_howto,
letzter Abruf am 11.08.15.

[11] wiringPi. *LCD Library*,
http://wiringpi.com/dev-lib/lcd-library/,
letzter Abruf am 11.08.2015.

[12] wiringPi. *Pins*,
http://wiringpi.com/pins/,
letzter Abruf am 11.08.2015.

[13] Institute for Program Structures and Data Organization. *Anschluss eines LCD-Displays,* http://www.ipd.uka.de/~buchmann/microcontroller/lcd.htm,
letzter Abruf am 11.08.2015.

[14] Erik Bartmann. *Durchstarten mit Raspberry Pi,*
1 Auflage 2012, O'Reilly Verlag, Köln, 2012.

[15] wiringPi. *Download and Install*,
http://wiringpi.com/download-and-install/,
letzter Abruf am 11.08.2015.

[16] Wikipedia. *ARM-Architektur,*
https://de.wikipedia.org/wiki/ARM-Architektur,
letzter Abruf am 12.08.2015.

[17] Broadcom. *BCM2835 ARM Peripherals,*
http://www.farnell.com/datasheets/1521578.pdf,
letzter Abruf am 12.08.2015.

[18] Pieter-Jan.com. *Low Level Programming of the Raspberry Pi in C,*
http://pieter-jan.com/node/15,
letzter Abruf am 12.08.2015.

[19] Raspberrypi.org. *Simple GPIO access in C,*
https://www.raspberrypi.org/forums/viewtopic.php?t=8476,
letzter Abruf am 12.08.2015.

[20] Kampis Elektroecke. *Einfacher I/O Zugriff,*
http://kampis-elektroecke.de/?page_id=5333,
letzter Abruf am 12.08.2015.

[21] Produktdatenblatt LCD-Modul TC1602A-09,
http://www.pollin.de/shop/downloads/D120422D.PDF.

[22] Alfred State. *LCD Addressing,*
http://web.alfredstate.edu/weimandn/lcd/lcd_addressing/lcd_addressing_index.html,
letzter Abruf am 12.08.2015.

[23] Wikipedia. *HD44780,*
https://de.wikipedia.org/wiki/HD44780,
letzter Abruf am 02.08.2015.

[24] Protostack. *HD44780 Character LCD Displays -Part 1,*
http://www.protostack.com/blog/2010/03/character-lcd-displays-part-1/,
letzter Abruf am 12.08.2015.

[25] Dr. Andrew Robinson, Mike Cook. *Raspberry Pi Projects,*
1. Auflage, John Wiley & Sons Ltd, United Kingdom, 2014.

[26] heise online. *Raspberry Pi: Mehr als fünf Millionen verkaufte Kleinstcomputer,*
http://www.heise.de/newsticker/meldung/Raspberry-Pi-Mehr-als-fuenf-Millionen-verkaufte-Kleinstcomputer-2553159.html,
letzter Abruf am 12.08.2015.

[27] Raspberry Pi B+, *Datenblatt,*
http://www.farnell.com/datasheets/1883763.pdf,
letzter Abruf am 12.08.2015

[28] Jürgen Wolf. *C von A bis Z - Das umfassende Handbuch <openbook>,*
3. aktualisierte und erweiterte Auflage, Rheinwerk Computing, 2009.

[29] cquestions.com. *Program to convert decimal to binary in c,*
http://www.cquestions.com/2011/07/program-to-convert-decimal-to-binary-in.html,
letzter Abruf am 12.08.2015

[30] cquestions.com. *Printing ASCII value using c program,*
http://www.cquestions.com/2008/01/write-c-program-for-printing-ascii.html,
letzter Abruf am 12.08.2015

[31] David Money Harris & Sarah L. *Harris. Digital Design and Computer Architecture,*
Second Edition, Morgan Kaufmann/Elsevier, Waltham, USA, 2013.