Beuth Hochschule für Technik
Fachbereich VI - Informatik und Medien
Luxemburger Straße 10
13353 Berlin

MASTERARBEIT
zur Erlangung des akademischen Grades
Master of Science

im Rahmen des Studiums
Medieninformatik

Entwicklung einer Softwarelösung zum Energiemonitoring

vorgelegt von

Dogan A,
Matrikelnummer: s123456

E-Mail: do.al@gmx.net

Berlin, den 1. November 2018

Betreuer: Prof Dr. Joachim
Gutachter: Prof Dr. Alfred

Inhaltsverzeichnis

1. Einleitung

Energie ist für vieles nötig. Ohne Energie wären die bisherige industrielle Entwicklung und das Leben wie es bekannt ist, nicht möglich. Erst Energieträger wie Öl, Erdgas und vor allem elektrische Energie sollen den wirtschaftlichen und technischen Fortschritt erst ermöglichen haben [**Bpb2013**] and [**Bpb2013**]. 2014 wurden 684 TWh Strom erzeugt. Die Quelle der erzeugten elektrischen Energien wurden vor allem durch fossile Brennstoffe erzeugt [**EDG**].

Aufgrund des Energieverbrauchs finden auch Klimaänderungen durch Umweltverschmutzung, verursacht durch Treibhausgase wie Kohlendioxid, statt. Dies hat nicht nur eine steigende globale Durchschnittstemperatur, sondern auch extreme Wetterereignisse, wie Stürme und Überflutungen zur Folge [**Ven2013**]. Die globale Durchschnittstemperatur soll um bis zu vier Grad Celsius steigen, wenn nichts dagegen unternommen wird.

Die Reduzierung des Stromverbrauchs hat nicht nur klimaschützende, sondern auch Kosten reduzierende Wirkung, da verminderte Nachfrage und Verbrauch auch entsprechend niedrigere Ausgaben für Strom bedeuten. Die Kontrolle des eigenen Energieverbrauchs wird daher immer wichtiger. Damit ist nicht nur die Überwachung der eigenen Energiekosten möglich, sondern auch die Steigerung der Energieeffizienz von Anlagen [**pae**]. Zudem wird durch die Energieverbrauchskontrolle die Möglichkeit eröffnet, die Umwelt zu schonen. Die Reduzierung des eigenen Stromverbrauchs und die Vermeidung von Spitzen sorgen dafür, dass die Stromversorgung und somit auch die Zulieferung und Produktion von Strom eingedämmt wird. Umweltschutz wird also nicht nur dadurch gefördert, wie und mit welchen Mitteln Strom gewonnen wird, z.B. durch erneuerbare Energien, sondern auch durch die Nachfrage danach. Um zum Umweltschutz beizutragen kommt es also nicht nur darauf an, wie der Strom erzeugt wird, sondern auch wie viel Energie verbraucht wird. Es hängt somit auch vom eigenen Verbrauch ab, weil die Menge des produzierten Stroms vermindert wird. Da die Stromerzeugung auch von der Nachfrage abhängig ist.

Hier kommt die in dieser Arbeit entwickelte Softwarelösung zum Einsatz. Mit der S0-Schnittstelle, nach DIN EN 62053-31, kann nicht nur der aktuelle Verbrauchstand von den Zählern direkt abgelesen, sondern auch über ganze Zeiträume hinweg erfasst werden, die dann aufbereitet und Visualisiert werden. So werden die Verbrauchsdaten nicht nur für einen bestimmten Zeitpunkt dargestellt, sondern für ganze Zeiträume.

Bei der Visualisierung der Energieverbrauchsdaten über die Zeit können die Verbrauchsdaten analysiert werden. Durch die Analyse können unnötige Verbräuche erfasst, dargestellt, eventuelle Zusammenhängen erkannt und so der Energieverbrauch optimiert werden.

2. Aufgabenstellung

Die Optimierung des Stromverbrauchs wird immer wichtiger. Dadurch kann die Umwelt geschützt und der eigene Energieverbrauch gesenkt werden. Ersteres wird aufgrund der immer weiter steigenden Umweltverschmutzung bei der Energiegewinnung wichtiger.

Zur Reduzierung der Umweltverschmutzung bei der Energiegewinnung und damit auch beim Energieverbrauch kann beigetragen werden, indem der eigene Verbrauch analysiert, unnötige Verbräuche ermittelt und optimiert werden können. Die Reduzierung des Energieverbrauchs hat auch einen Ausgaben reduzierenden Charakter.

Dazu soll in dieser Bachelorarbeit eine Monitorringlösung entwickelt werden, die über vier S0-Schnittstellen, den Energieverbrauch in Gebäuden automatisch am Zähler misst, verarbeitet und abspeichert. In dieser Arbeit soll mit dem Aufsatz SD0, direkt über seinen vier S0-Kanälen, von den am Stromzähler angebrachten S0-Schnittstellen der Energieverbrauch gemessen werden. Die Abspeicherung der Verbrauchswerte soll sowohl lokal, als auch auf einem Server erfolgen können. Dazu gibt der Aufsatz die Verbrauchswerte an den Rasperry Pi weiter, wo die Verbrauchswerte abgespeichert werden. Vom Raspberry Pi aus können, wenn nötig, die Verbrauchsdaten an einen Server weitergeleitet und dort abgelegt zu werden. Die Visualisierung der Verbrauchswerte soll über eine Webdarstellung vorgenommen werden.

Die Verbrauchsmessung und deren Darstellung ermöglicht, dass die Verbrauchsdaten analysiert und der Stromverbrauch des Nutzers bzw. des Systems optimiert werden können.

Die aufgenommenen S0-Impulse können Werte von Elektro-, Wasser- oder Gaszählern repräsentieren. Für die Möglichkeit der Konfiguration der Impulskonstante und der Messleitung ist eine Einstellmöglichkeit herauszusuchen. Die entwickelte Lösung ist prototypisch zu implementieren und in einem Testaufbau zu überprüfen.

3. Fachliches Umfeld

3.1. Beschreibung des Raspberry Pi

In diesem Kapitel werden die für diese Arbeit eingesetzte Hardware und die eingesetzten Schnittstellen UART und I^2C erläutert. Zusätzlich werden die Datenhalungs bzw. -darstellungsform beschrieben. Bei der Abschlussarbeit wird Raspberry Pi und die Erweiterung SD0 von Busware eingesetzt.

Da für die Entwicklung der Softwarelösung der Aufsatz SD0 von Busware verwendet werden muss, trifft die Wahl des Einplatinencomputers auf den Raspberry Pi, weil der Aufsatz für den Raspberry Pi konzipiert worden ist [**bsd0**].

Als Platine für die Erweiterung wird der Raspberry Pi eingesetzt, da die Erweiterung SD0 dafür ausgerichtet ist. In diesem Kapitel wird entschieden welches Modell des Raspberry Pi ausgewählt wird und dieses beschrieben.

Aufgrund der Anzahl der Pins und der Anordnung der Komponenten auf dem Board kann zwischen den Raspberry Pi Modellen A und B ausgewählt werden, da der Aufsatz SD0 an sich 26 Pins hat. Bei den anderen Raspberry Pi Modellen, also beim Raspberry Pi A+, B+ und Raspberry Pi 2 Modell B würde es beim Aufsetzen auf die Raspberry Pi Modellen mit mehr als 26 Pins die anderen Pins umbiegen bzw. zerstören. Zudem würde der *Display Connection Port* (DSI), das Aufstecken des Aufsatzes SD0 behindern. Der Aufsatz hat eine entsprechende Einbuchtung für den DSI Port bei Raspberry Pi A und B.

Da der Raspberry Pi Modell B, im Gegensatz zum Modell A einen höheren Arbeitsspeicher und einen Netzwerkanschluss hat, wird Modell B eingesetzt. Der Netzwerkanschluss ist für die Darstellung des Energieverbrauchs wichtig, weil die Daten entweder für die Darstellung an einen Server gesendet oder auf dem Raspberry Pi präsentiert werden.

In diesem Kapitel wird nicht nur der SoC *BCM2835*, welcher den Kern des Raspberry Pis bildet erläutert, sondern auch die Komponenten die für den Einsatz der zu entwickelnden Softwarelösung benötigt werden. Dabei handelt es sich vor al-lem um die Ein- und Ausgänge, insbesondere den GPIO Pins mit den benutzten Funktionen. Der Grafikprozessor, mit seinen Ausgängen wird nur am Rande erwähnt.

3.1.1. System on a Chip (SoC)

Der Raspberry Pi Model B basiert auf dem *BCM2835* von Broadcom, dessen Kern die CPU *ARM1176JZF-S* ist. Die CPU gehört der ARM11 Generation an, welche auf der Architektur ARMv6 mit 700 MHz basiert [**Kof2015**] und [**Upt2014**]. Sie hat eine Adressbreite von 32 Bit [**Kof2015**]. Beim BCM2835 handelt es sich um einen SoC, also um einen System on a Chip. Dies bedeutet, dass es nicht nur die CPU sondern auch andere Teilsysteme beinhaltet, so dass diese nicht einzeln auf dem Raspberry Pi verbaut sind. Neben dem CPU beinhaltet es den Grafikprozessor (GPU) und den Arbeitsspeicher (SDRAM) in der Größe von 512 MB [[**Kof2015**] und [**Dem2013**]] sowie die Verbindungen zu den GPIO Pins. Die CPU besitzt zudem separate Cache-Speicher für Daten und Befehle [**Dem2013**]. Beim Arbeitsspeicher handelt es sich um einen Internen Speicher,sodass die CPU keine externen Datenverbindungen besitzt, da keine externen Arbeitsspeicher angeschlossen werden [**Dem2013**].

Tabelle 3.1.: Eigenschaften des Raspberry Pi

Eigenschaften		Modell A	Modell A+	Modell B	Modell B+
Gesamtgrösse (in mm)	Länge	93	70,4	9	
	Breite	63,5	57,2	63	
	Höhe	17	12	2	
SoC		Broadcom BCM2835			
CPU	Typ	ARM1176JZF-S			
	Kern	1			
	Takt	700 MHz			
	Architektur	ARMv6			
Arbeitsspeicher		256 MB	512 MB		
Speicher			Kartenleser für		
Full SD	Micro SD	Kartenleser für Full SD	Micro SD	Micro SD	
Anzahl der USB 2.0 Anschlüsse		1		2	4
Ethernet		-		10 und 1	
Pin		26	40	26	
GPIO-Pins		17	26		
Weitere Schnittstellen		1 x CSI, 1 x DSI, 1 x I^2C, 1 x I^2S			
Stromversorgung		5,0 V; über einen Micro-USB-Anschluss (Micro-U			

Beim Grafikprozessor handelt es sich um einen *VideoCore IV* welches den H.264 Standard mit einer 40 MBits/s unterstützt, die Videos mit einer Datenkompression von H.263 mit einer Rate von bis zu 40 MBits/s darstellen kann. Der Grafikspeicher soll automatisch ein Teil des Arbeitsspeichers mitbenutzt werden [**Kof2015**]. Die Darstellung kann über drei verschiedene Ausgänge erfolgen. Bei den Ausgängen handelt es sich um HDMI, Composite Video und DSI [**Dem2013**].

Die mit dem Grafikprozessor verbundenen Comosite-Video und HDMI Anschlüsse können direkt eingesetzt werden. Der Display Serial Interface benötigt spezielle Hardware [**Upt2014**] wie einem Folienkabel [**Dem2013**].

Das DSI ist eine serielle Steckerleiste mit 15 Pins, an dem ein Display angeschlossen und der RPi autonom benutzt werden kann, ohne dass es an einen Monitor oder Display, wie sie von normalen Computern her bekannt ist, angeschlossen werden muss [[**Upt2014**] und [S.72]Dem2013]. Die Klemmhalterung dient der Aufnahme eines Folienkabels.

In dieser Arbeit wird weder der Grafikprozessor noch die Anschlüsse des Grafikprozessors verwendet. Lediglich der Jumper des DSI muss abmontiert werden, da sonst der Aufsatz SD0 von Busware nicht richtig auf die GPIO-Leiste des Raspberry Pis aufgesteckt werden kann.

3.1.2. Stromversorgung

Der Raspberry Pi Modell B sollte mit einem Strom von mindestens 700 mA (Milliampere) und einer Spannung von 5 Volt versorgt werden. Eine Polyfuse verhindert eine Überversorgung von über einem Ampere, also 1000 Milliampere [**Dem2013**].

3.1.3. Datenspeicher des Raspberry Pis

In diesem Unterkapitel wird nicht der Arbeitsspeicher behandelt, sondern der Daten- und Programmspeicher. Da der Raspberry Pi keinen Speicher bzw. keine Festplatte besitzt, benötigt er eine SD-Karte [**Upt2014**]. Auf der SD-Karte werden nicht nur Programme und Daten abgespeichert; darauf läuft auch das Betriebs-system.

Es können drei verschiedene SD-Karten-Standards am Raspberry Pi Model B an-geschlossen werden, da diese die gleichen Abmessungen besitzen. Anzumerken ist, dass der Schreibschutz an den Karten vom Raspberry Pi nicht ausgewertet wird. An den SD-Karten Slot können SD-Karten mit den Abmessungen 32 mm x 24mm x 2,4mm angeschlossen werden. Bei den einsetzbaren SD-Karten handelt es sich um folgende Karten [**Dem2013**]:

Es können drei verschiedene SD-Karten-Standards am Raspberry Pi Model B an-geschlossen werden, da diese die gleichen Abmessungen besitzen. Anzumerken ist, dass der Schreibschutz an den Karten vom Raspberry Pi nicht ausgewertet wird. An den SD-Karten Slot können SD-Karten mit den Abmessungen 32 mm x 24mm x 2,4mm angeschlossen werden. Bei den einsetzbaren SD-Karten handelt es sich um folgende Karten [**Dem2013**]:

SD-Karte mit einer Kapazität von 8 MB bis 2 GB

SDHC (SD High Capacity) Card mit einer Kapazität von 4 GB bis 32 GB und

SDXC (SD eXtended Capacity) Card mit einer Kapazität von 64 GB bis 2 TB

Die SDHC-Karte lässt sich wiederum in verschiedene Klassen unterteilen. Diese Klassen sind Abstufungen der Mindestübertragungsrate der Karte. Diese lassen sich folgendermaßen aufteilen [**Dem2013**]:

Class 2 mit 2 MByte/s Mindestübertragungsrate

Class 4 mit 4 MByte/s Mindestübertragungsrate

Class 6 mit 6 MByte/s Mindestübertragungsrate

Class 8 mit 8 MByte/s Mindestübertragungsrate

Mit den unteren Übertragungsraten, also mit Class 2 und 4 soll es kaum Probleme geben. Mit den höherklassigen SD-Karten, also mit Class 6 und 8, kann es jedoch Kompatibilitätsprobleme geben. Die höheren Klassen werden bei der Wiedergabe von ruckelfreien Videowiedergaben bevorzugt [**Dem2013**]. Für das Betriebssystem des Raspberry Pi werden mindestens 4 GB gebraucht [**Upt2014**]. Es wird jedoch 8 GB empfohlen [**Upt2014**].

Das SD-Karten Slot, das sich unterhalb des Raspberry Pi befindet, ist direkt mit dem SoC BMC2835 verbunden. Der eigentliche Card-Controller befindet sich auf der SD-Karte. Neben der Masseleitung des Blechslots (MTG1 und MTG2) und den Masseleitung für die Karte selbst (Vss1 und Vss2) wird der Slot bzw. die Karte über die Leitung Vdd mit Spannung versorgt. Die Datenübertragung erfolgt taktgesteuert. Die Karte wird über die Leitung CLK mit dem Takt vom BCM2835 versorgt. Die Datenübertragung an sich findet über die Datenleitungen DAT0, DAT1, DAT2, DAT3 statt. Die Leitung DAT3 kann gleichzeitig zur Kartenerkennung (CD: Card Detection) benutzt werden. Ob auf die Karte geschrieben oder von der Karte gelesen werden soll, wird von der Leitung CMD bestimmt [**Dem2013**].

3.1.4. Betriebssystem

Für den Raspberry Pi existieren unterschiedliche Betriebssysteme, die für verschiedene Einsatzzwecke konzipiert wurden. Bei den meisten handelt es sich um Betriebssysteme von Drittanbietern [**RpiD**].

Neben Betriebssystemen, wie OpenELEC (Open Embedded Linux Entertainment Centre) oder OSMC (Open Source Media Centre) [**RpiD**]. die als Mediacenter, also für das Streaming von Medien, wie Musik und Filmen genutzt werden können, [**Kof2015**] gibt es Betriebssysteme, die auf verschiedenen Linux-Distributionen basieren. Dabei handelt es sich beispielsweise um Raspbian, RISC OS, Pidora oder ArchLinux ARM. Das Betriebssystem wird vor allem im Hinblick auf die Unterstützung des Raspberry Pi und evtl. späterer Änderungen und Erweiterungen. des im Rahmen dieser Arbeit erstellten Softwarelösung ausgewählt.

Pidora, ist ein auf der Fedora-Distribution basierendes Betriebssystem [**Kof2015**]. Pidora soll langsam arbeiten und im Betrieb sollen merkliche Fehler auftauchen[**Dem2013**].

Beim **ArchLinux** handelt es sich um einen an Einplatinencomputer, wie BeagelBone und Raspberry Pi optimierte Arch-Distribution [**Kof2015**]. Es soll jedoch eine geringe Anzahl an Softwarepaketen anbieten, was nicht optimal für spätere Erweiterungen der entwickelten Software sein kann.

RiscOS ist ein nicht auf Linux basierendes, im Jahre 1987 von Acorn entwickeltes Betriebssystem. Für dieses Betriebssystem soll es wenig Open-Source-Software geben [**Dem2013**].

Beim **Raspbian** handelt es sich um ein vom Raspberry Foundation empfohlenes Betriebssystem [**Upt2014**].Bei dieser Arbeit wird Raspbian eingesetzt. Das hat verschiedene Gründe. Raspian soll einerseits ein stabiles, auf Debian basierendes und andererseits speziell auf den Broadcom BCM2835 angepasstes Betriebssystem sein, [**Dem2013**] das auf ein direktes Zusammenspiel mit der Hardware des Raspberry Pi ausgerichtet ist. [**EPB1**] Es unterstützt beispielsweise den Floating Point Unit [**Dem2013**]. Auch in Hinblick auf den oben genannten Punkt der Erweiterung der zu entwickelnden Software, erfüllt es das Kriterium. Mit Raspbian werden Softwarepakete mitgeliefert, die entweder nur ihm zur Verfügung stehen oder in anderen Distributionen kompiliert werden müssen, falls diese dort zum Einsatz kommen sollen [**Kof2015**].

3.1.5. I/O - Anschlüsse

Der Raspberry Pi benutzt den Micro USB-Anschluss und den SD-Karten-Slot zum Betrieb. Auf der SD-Karte läuft, wie im Kapitel „Datenspeicher“ erwähnt, das Be-triebssystem. Über den Micro USB-Anschluss wird der Raspberry Pi mit Strom ver-sorgt. Darüber hinaus kann es auch mit einem Computer verbunden werden. In die-sem Fall wird es über den Computer mit Strom versorgt. Über den SD-Karten Slot wird eine SD-Karte angeschlossen, die das Betriebssystem für den Raspberry Pi be-inhaltet. Die SD-Karte wird nicht nur für das Betriebssystem, sondern auch als Spei-cher benutzt.

Neben den Anschlüssen, die für den Betrieb gebraucht werden, besitzt der Raspberry Pi Model B jeweils einen Audio- und Video Composit-Ausgang und zwei USB-Anschlüsse, über die beispielsweise Tastatur oder Maus angeschlossen wer-den können. Daneben besitzt er einen HDMI-Anschluss für eine hochauflösende, verlustfreie Video- und Audiosignalübertragung [vgl. ITW]. Wie in der Tabelle 3.2.1 im Kapitel 3.2 dargestellt, besitzt der Raspberry Pi Model B auch einen Ethernet -Anschluss, für einen Anschluss an ein Netzwerk.

Der Raspberry Pi Model B hat neben den fünf Status LEDs und dem Ethernet An-schluss noch die Verbindungen, die von S1 bis S7 und P1 bis P6 bezeichnet werden. In den folgenden beiden Tabellen werden die Schnittstellen erläutert und mit konkre-ten Anschlüssen dargestellt.

Tabelle 3.2.: Schnittstellen des Raspberry

Anschluss	Beschreibung
S1	Micro USB-Anschluss (für die Stromversorgung)
S2	DSI-Anschluss
S3	HDMI Anschluss, Typ A
S4	Composite Video Anschluss (RCA)
S5	CSI Anschluss
S6	Audio-Ausgang; 3,5 Klinker Anschluss
S7	SD Karten-Slot (befindet sich unterhalb des RPi)

Die Folgende Tabelle Listet die verfügbaren GPIO Pins des Raspberry Pis auf:

Tabelle 3.3.: GPIO Anschlüsse des Raspberry

Anschluss	Beschreibung
P1	GPIO Steckerleiste; 2x 13 Pins
P2	JTAG für GPU; Für 8 Pins (keine angeschlossenen Pins)
P3	JTAG für LAN9512; Für 7 Pins (keine angeschlossenen Pins)
P4	RJ45 Ethernet Anschluss
P5	GPIO Erweiterungspins; 8 Anschlüsse ohne Pins
P6	Reset-Button-Anschluss; (keine angeschlossenen Pins)

In dieser Arbeit werden die Anschlüsse S1, S7, P1 und P4 verwendet.

3.1.6. GPIO Anschlüsse

Bei einer General Purpose Input/Output (kurz GPIO) Schnittstelle handelt es sich, ganz allgemein betrachtet, um eine Schnittstelle die nichts über die einzelnen Pins, noch über deren elektrische Daten oder die Art und Weise der Programmierung aussagt. Erst durch den Einsatz auf einer bestimmten Platine und deren Bestückung gewinnt die GPIO an Bedeutung [vgl. Dem2013 S. 74 bis S.75]. Beim GPIO des Raspberry Pi handelt es sich um eine Schnittstelle, über die der Raspberry Pi mit Schaltungen, Platinen usw. erweitert werden kann. Wie aus der Tabelle 3.2 zu entnehmen ist, existieren neben der eigentlichen GPIO-Schnittstelle mit der Bezeichnung P1 noch weitere Schnittstellen, die in unterschiedlichen Bereichen eingesetzt werden.

Bei den Schnittstellen mit der Bezeichnung P2 und P3 handelt es sich um JTAG Schnittstellen (Joint Test Action Group). Diese werden bei der Produktion des Raspberry Pi benutzt, um die Funktionalität des Raspberry Pi zu testen [vgl. Kof2015 S. 364 bis S.365]. Die Schnittstelle mit der P2-Bezeichnung wird für den GPU also für die Grafikeinheit benutzt. Mit der P3-Schnittstelle

wird die Funktionalität des LAN2512-Chips getestet. Der LAN2512 beinhaltet den USB- Hub, mit 2 USB2.0-Ports und den LAN Controller.

Die GPIO Schnittstelle mit der Bezeichnung P4 ist mit der RJ45 bzw. Ethernet Buchse und dem LAN-Controller des LAN9512 verbunden. Daher wird diese Schnittstelle beim Raspberry Pi Modell B für Ethernet Anschluss verwendet und ist beim Modell B standardmäßig mit dem jeweiligen Port belegt.[**Kof2015**]

Die GPIO Schnittstelle mit der Bezeichnung P5 besitzt acht Kontakte, die nicht mit Pins bestückt sind. Diese GPIO Schnittstelle steht erst ab der 2 Revision zur Verfügung. Über die ersten beiden Kontakte werden Spannungen zur Verfügung gestellt. Beim ersten Kontakt sind 5 Volt und über den zweiten Kontakt 3,3 Volt verfügbar. Die letzten beiden Kontakte bilden die Masse. Die restlichen vier Kontakte der Schnittstelle bieten zwei Funktionen an: die Die Primärfunktion bildet das Bussystem I^2C über die Kontakte 3 und 4. Dieser ist der zweite I^2C Kanal, auf dem Raspberry Pi. Die zweite bzw. alternative Funktion bilden die Kontakte 5 und 6. Über diese ist der I^2S Bus benutzbar. Dieses Bussystem überträgt Audiosignale [vgl. Kof2015 S.365].

Die Schnittstelle P6 bietet die Möglichkeit eines Hardware-Rests an. Darüber wird also die CPU neugestartet [vgl. Kof2015 S.365]. Dazu müssen die beiden Kontakte miteinander kurzgeschlossen werden [vgl. Kof2015 S.1032].

Die GPIO-Schnittstelle mit der Bezeichnung P1 stellt die Basis für weitergehende Projekte dar. Durch die GPIO-Schnittstelle kann der Raspberry Pi mit Schaltungen, Platinen usw. um Funktionalitäten erweitert werden. In diesem Kapitel wird diese GPIO- Schnittstelle mit der Anschlussbezeichnung P1 näher beschrieben, da diese Schnittstelle nicht nur für die eigentlichen Erweiterungen, sondern auch in dieser Arbeit für die Erweiterung SD0 von Busware verwendet wird. Die Schnittstelle selbst arbeitet intern mit einer Spannung von 3,3 Volt. Die Zufuhr von höheren Spannungen als 3,3 Volt oder der Kurzschluss der 5 Volt Pins, mit den anderen GPIO-Pins könnte den Raspberry Pi zerstören [vgl. Upt2014 S. 222].

Beim Raspberry Pi erfüllt jede GPIO Pin eine bestimmte Aufgabe. Einige Pins haben Doppelfunktionen. Alle Signale werden vom BCM2835 bereitgestellt. Die GPIO-Schnittstelle des Raspberry Pi Modell B, welches sic h in der Abbildung 3.1auf der oberen linken Ecke des Raspberry Pis befindet, besteht aus zwei Reihen mit jeweils 13 Pins.

3.1.7. LAN9512

Das in dieser Arbeit eingesetzte Raspberry Pi Model B hat einen LAN9512. Der LAN9512 ist neben den SoC BCM2835 der zweite Chip auf dem Raspberry Pi Model B. Es beinhaltet verschiedene Einheiten, wie einen LAN-Controller und den USB Hub.

Anzumerken ist, dass der SoC BCM2835 als USB-Controller und der LAN9512 nur als USB-Hub mit 2 Ports dient [vgl. LAN9512, S. 6 und Dem2013, S. 81]. Der USB-Anschluss wird in dieser Arbeit nicht verwendet. Aber jedoch der Ethernet-anschluss.

3.1.8. Ethernet

Der im LAN9512 eingesetzte Ethernet-Controller unterstützt die Ethernet-Standards 10Base-T mit 10Megabit/s, die im IEEE802.3 und 100Base-TX mit 100Megabit/s, welches im IEEE 802.3u spezifiziert sind [vgl. LAN9512 S.7 und Dem2013 S.84]. Zusätzlich besteht die Funktion „Wake on LAN“. Dies bedeutet, dass der Raspberry Pi über das angeschlossene Netzwerk aktiviert werden kann, wenn sich dieser im Ruhe Modus befindet. 100Base-TX unterstützt die automatische

Abbildung 3.1.: Schaltung des GPIO Steckerleiste von Raspberry Pi Model B

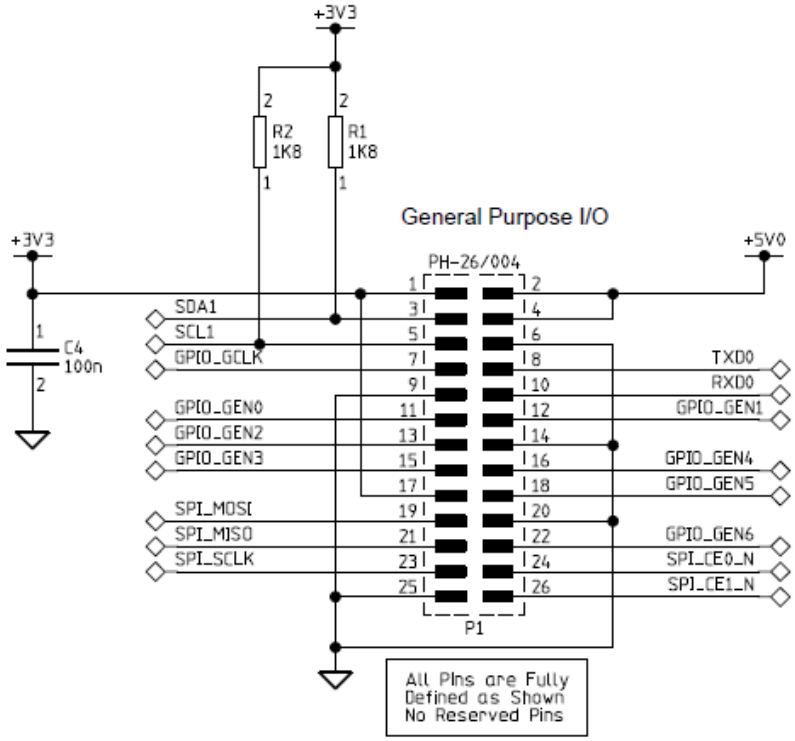
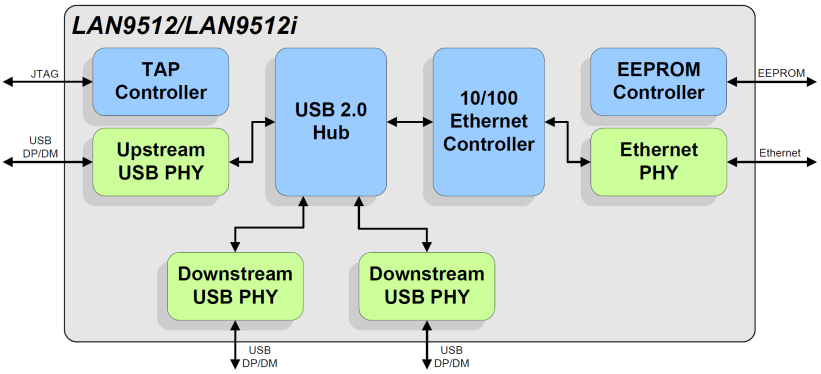
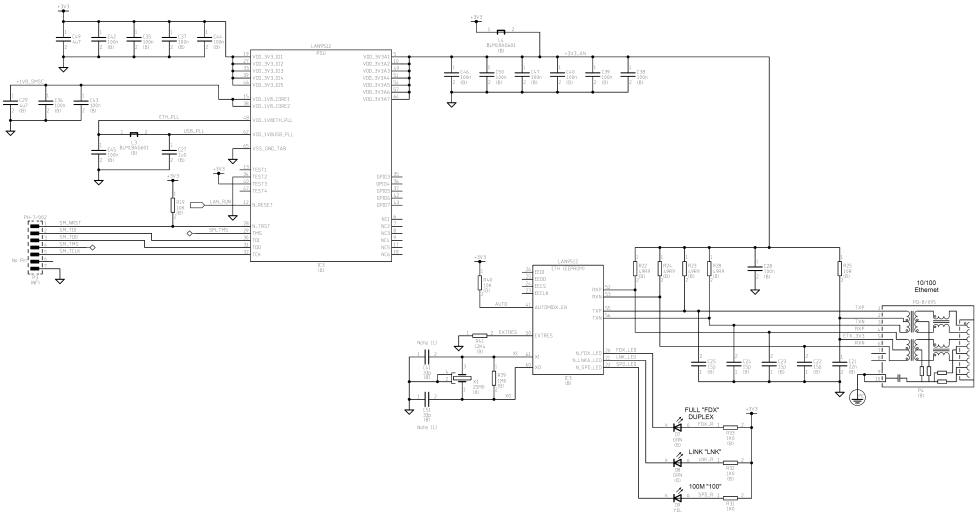


Abbildung 3.2.: Prinzipschaltbild des LAN9512



Detektion (Auto Negotiation) der Betriebsart. Es erkennt also die Verdrahtung des Ethernet-Kabels, das an der RJ45-Buchse angesteckt ist. Es wird auch Full-Dublex unterstützt, es kann somit gleichzeitig ge-sendet und empfangen werden [vgl. Dem2013 S.84].

Abbildung 3.3.: Ethernet-Schaltung des Raspberry Pi Model B



3.2. Datenbusse

Der Raspberry Pi Modell B stellt verschiedene Bussysteme zur Verfügung. Bussysteme, auch als Bus abgekürzt, werden für die Datenübertragung zwischen verschiedenen Teilnehmern benutzt [**Kof2015**]. Je nach verwendetem Bussystem werden die Daten seriell, also hintereinander über dieselben Datenleitungen oder parallel, also gleichzeitig über verschiedenen Datenleitungen übertragen. Auf dem Raspberry Pi Modell B sind folgende fünf Bussysteme verfügbar, bei dem alle Daten seriell übertragen werden [**Kof2015**],

- UART, Abk. für Universal Asynchronous Receiver/Transmitter.
- SPI, Abk. für Serial Peripheral Interface
- I^2C, Abk. für Inter Integrated Circuit
- I^2S, Abk. für Integrated Interchip Sound
- 1Wire, Eindraht-Verbindung.

Da in dieser Arbeit der UART und I^2C verwendet werden, werden diese zwei Bussysteme beschrieben. Die vollständige Erläuterung der Bussysteme würde den Rahmen dieser Arbeit sprengen, daher werden nur die grundlegenden Punkte erläutert. Die Bussysteme unterscheiden sich nicht nur, aber vor allem in der Datenübertragung. Dabei soll es zwei grundsätzliche Formen der Datenübertragung geben, die sich vor allem bei der Synchronisation durch einen Takt unterscheiden [**Bei2011**]:

Bei der synchronen Datenübertragung wird parallel zur Datenübertragung der Takt übertragen. Durch die Taktübertragung können die Kommunikationspartner den Be-ginn und das Ende einer

Datenübertragung erkennen [**Kof2015**; **Bei2011**].

Bei der asynchronen Datenübertragung existiert keine separate Taktleitung. Die Daten werden als einzelne Zeichen übertragen. Dabei werden diese in Start- und Stoppbits verpackt. Die Start- und Stoppbits können variieren [**Bei2011**].

Der Vollständigkeit halber wird im Folgenden der 1Wire und I^2S beschrieben. Beim 1-Wire Bus handelt es sich um einen Eindraht-Bus, deshalb auch der Name des Bussystems [vgl. Kof2015 S.491]. Der, ursprünglich vom Unternehmen Dallas (heute Maxim) [vgl. Gei S. 3] entwickelte 1-Wire-Bus besitzt im Vergleich zu SPI und I^2C keine zusätzliche Taktleitung. Dabei handelt es sich also um eine asynchrone Datenübertragung. 1-Wire-Bus führt lediglich neben der einzelnen Leitung zur Bus-kommunikation, also der Datenleitung mit der Bezeichnung DQ, noch die Masse Lei-tung [vgl. Gei S. 3 und Kof2015 S.491]. Die Datenleitung wird nicht nur dazu benutzt, um Daten zu senden und zu empfangen, sondern auch um die Erweiterungen mit ei-ner Versorgungsspannung von 3,3 Volt zu versorgen [vgl. msx und Kof2015 S.491]. Die Übertragung erfolgt dabei seriell, bidirektional und asynchron, da über die Datenleitung entweder nur gesendet oder empfangen werden kann und dies auch ohne Taktleitung erfolgt [vgl. Gei S. 3].

Der Raspberry Pi bietet ab der Revision 2 den Soundbus I^2S, Abk. für Integrated Interchip Sound. Der I^2S Bus transportiert Audiosignale zwischen integrierten Schalt-kreisen. [vgl. Kof2015 S.489].

Das ursprünglich von Motorola entwickelte Bussystem Serial Peripheral Interface (SPI) arbeitet nach dem Master-Salve-Prinzip, mit einem Master und einer theore-tisch unbegrenzten Anzahl an Slaves. Dabei erfolgt die Kommunikation Synchron [vgl. Bei2011 S. 208].

3.2.1. UART

Der Universal Asynchrounous Receiver/Transmitter (kurz UART) Protokoll wird bei der Kommunikation von RS232- oder RS485-Schnittstellen eingesetzt. Der UART wird auch zur Datenübertragung mit Mikrocontrollern und wie hier mit Einplatinen-Computern benutzt.

Die Daten werden beim UART, im Gegensatz zum USART, der eine synchrone Datenübertragung erlaubt, asynchron übertragen. Das heißt dass es keine Taktleitung gibt. Der UART besitzt im Ruhezustand einen High-Pegel, was der logischen 1 ent-spricht. [vgl. Pla uart] Die Daten werden mit einem Start- und Stoppbit versehen. Es kann zusätzlich einen Paritätsbit geben. Üblicherweise werden acht bis neun Bits übertragen [vgl. MUA]. Dabei werden also insgesamt 10 bis 12 Bits zusammen über-tragen. Diese werden auch als Wort bzw. Datenwort bezeichnet. Die Reihenfolge der übertragenen Bits sieht beim UART folgendermaßen aus:

Tabelle 3.4.: Aufbau des UART-Datenwortes

Startbit	D0	D1	D2	D3	D4	D5	D6	D7	(D8)	(Paritätsbit)	Stoppbit

Bei der obigen Abbildung zum Aufbau des UART-Datenwortes ist jedes Kästchen für ein Bit gedacht. Der Startbit wird als eine Logische 0 gesendet [vgl. Pla uart]. Zwischen zwei Datenworten können sich beliebig lange Pausen befinden, da jedes Wort am Startbit erkannt wird [vgl. Pla uart]. Die Bits D8, also der neunte Datenbit und die Paritätsbits sind gelb hintermalt, da diese optional sind.

Das Paritätbit überprüft, ob die Daten richtig übertragen wurden. Dabei wird zwischen odd parity und even parity unterschieden. Bei odd parity wird der Paritätsbit auf eins gesetzt, wenn es in den Datenbits eine gerade Anzahl an Einsen gibt [vgl. Mül2009 S. 6].

Even parity bildet den Gegensatz zur odd parity. Bei einer geraden Anzahl von Datenbits wird das Paritätsbit auf Null und bei einer ungeraden Anzahl auf eins gesetzt [vgl. Mül2009 S. 6].

Beim Stoppbit muss erwähnt werden, dass auch zwei Stoppbits hinter ein Daten-wort eingefügt werden können, um den Pausenpegel zwischen zwei Datenworten bei der Übertragung zu trennen [vgl. Bei2011 S. 262]. Da die Kommunikation beim UART asynchron abläuft synchronisieren sich die Kommunikationsteilnehmer für jeden Transfer neu [vgl. Pla uart]. Dies geschieht durch die oben erwähnten Start- und Stoppbits. Vor der Datenübertragung liegt der Pegel auf der Übertragungsleitung auf high [vgl. Pla uart]. Das bedeutet, dass die Leitung sich auf 3,3 Volt befindet.

Vor der Kommunikation zwischen Sender und Empfänger müssen diesen die Anzahl der Datenbits, Startbits und der Stoppbits sowie der Berechnung der Parität und die Frequenz des Übertragungstaktes der Daten kenntlich gemacht werden. Soll et-was übertragen werden, wird dies dem Empfänger der Daten durch den Startbit kenntlich gemacht. Anhand des High-Low-Pegels wird der Sendevorgang kenntlich gemacht, wodurch Empfänger und Sender synchronisiert sind. Der Sendevorgang wird durch einen Stoppbit beendet [vgl. Pla uart]. Der UART am Raspberry Pi besitzt die Leitungen TxD (am Pin 8) zum Senden und RxD (am Pin 10) zum Empfangen der Daten [vgl. Upt2014 S. 222]. Das Raspberry besitzt keine Leitung für einen Hardware-Handshake [vgl. Pla uart], zur Erkennung der Übertragung. Wird ein hardwarebasierter Handshake benötigt, kann auf einen freien GPIO-Pin zugegriffen werden [vgl. Pla uart].

Ein Baudratengenerator bzw. programmierbarer Timer sorgt für die Übertragungsgeschwindigkeit, auch Baud oder Baudrate genannt, der Daten [vgl. Bei2011 S. 262]. Die Baudraten müssen sowohl auf Empfängerseite als auch auf der Senderseite ü-bereinstimmen, da die Daten sonst nicht korrekt übertragen werden können. Das hat den Hintergrund, dass der UART - wie oben erwähnt - keine Taktleitung hat und der Empfänger nicht weiß, wann über UART Daten geschickt werden und wann nicht.

3.2.2. I²C

Das im Jahre 1982 von Philips Semiconductor entwickelte Bussystem Inter Intergratet Circuit (I^2C) wird für die Verbindung von integrierten Schaltungen auf Platinen bzw. innerhalb von Geräten eingesetzt [vgl. Bei2011 S. 209 und Dem2013 S. 185]. Beim I^2C handelt es sich um eine serielle Schnittstelle [vgl. Dem2013 S. 185], bei der die Datenübertragung sowohl synchron als auch asynchron erfolgt [vgl. Bei2011 263 bis S. 264].

Das Bussystem kann mit zwei bzw. drei Leitungen eingesetzt werden. Eine Leitung dient als Taktleitung, eine zweite als Datenleitung, wobei die dritte Leitung die Mas-seleitung ist, die als Bezugspegel dient [vgl. Bei2011 S. 209]. I^2C besitzt keine sepa-rate Leitung zum Auswählen der einzelnen Teilnehmer. Die einzelnen Teilnehmer werden durch Adressen ausgewählt [vgl. Kof2015 S.476]. Beim Raspberry Piwird der I^2C Datenbus auch vom BCM2835, also dem SoC, bereitgestellt [vgl. Upt2014 S. 223]. Der I^2C besitzt folgende Leitungen:

SDA Serial Data- Line hierüber werden die Daten zwischen den Kommunikationspartnern übertragen

SCL Serial Clock Line Taktleitung des Bussystems

Die Übertragung der Daten erfolgt aus einer Mischung aus synchroner als auch asynchroner Übertragung. Das bedeutet, dass bei der Übertragung der einzelnen Datenpakete diese jeweils mit Start und Stoppbits versehen werden [vgl. Bei2011 263 bis S. 264]. Die Übertragung der einzelnen Bits der Datenpakete wird jedoch mit dem Taktsignal der SCL-Leitung des jeweiligen Masters synchronisiert übertragen [vgl. Bei2011 S. 264].

An das Bussystem können ein oder mehrere Master angeschlossen werden, von denen jedoch mindestens einer die Kommunikation innerhalb des I^{2}C steuert [vgl. Bei2011 S. 209]. Ein Bus, der mehrere Master besitzt, wird auch als Multimasterbus bezeichnet [vgl. Dem2013 S. 187]. Am Bus können bis zu 128 Slaves angeschlossen werden, die eine im Bussystem eindeutige Adresse von sieben bzw. zehn Bit Länge besitzen [vgl. Dem2013 S. 189], wodurch die Slaves einzeln angesprochen werden können [vgl. Bei2011 S. 209 bis 210]. Die 10 Bit Adressierung soll kaum genutzt werden [vgl. Dem2013 S. 189]. Die Slaves können dadurch immer zum Senden bzw. Empfangen von Daten durch den Master aufgefordert werden. Der Empfang der Auf-forderung wird dem Master durch ein Quittierungsbit (auch Acknowledge bezeichnet) bestätigt [vgl. Bei2011 S. 210].

Da I^{2}C-Busse in der Regel Open-Collector-Eingänge besitzen, versorgen die I^{2}C-Bausteine keine Spannungspegel an ihren Pins. Beim Raspberry Pi werden die I^{2}C Pins jedoch durch Pull-Up-Widerstände auf 3,3 Volt gezogen, dieser bildet dann den positiven Signalpegel [vgl. Bei2011 210]. Bei der Kommunikation über die I^{2}C- Schnittstelle werden diese Leitungen auf Masse gezogen [vgl. Kof2015 S.476], das bedeutet, dass diese Leitungen dann auf 0 Volt herunter gezogen werden und somit auch einen LOW-Pegel besitzen [vgl. Kof2015 S.476].

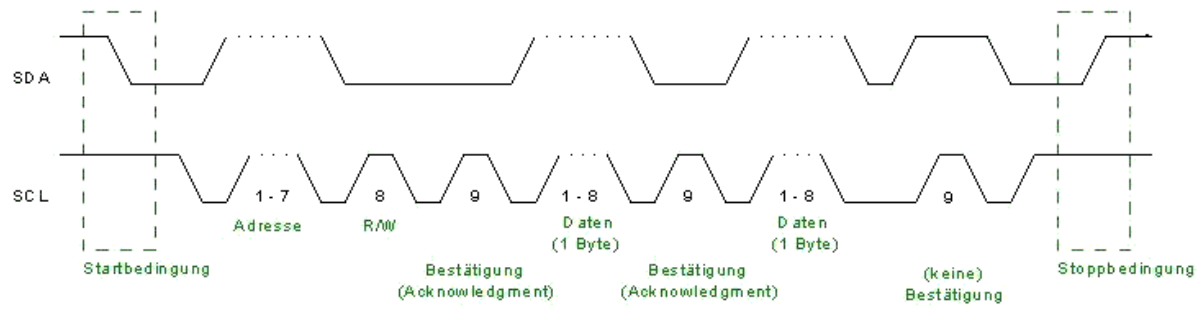
Tabelle 3.5.: Aufbau des UART-Datenwortes

Start	Slave	Adresse	Read/Write	Ackn.	Daten	Ackn.	Daten	Ackn.	Stop

In der Abbildung befindet sich der Startbit ganz links und der Stoppbit ganz rechts. Beim „Ackn.“ handelt es sich um den Quittierungsbit, was nach jeder Datenübertragung vom Slave an den Master erfolgt.

Beim I^{2}C kann jede Baugruppe, je nach Funktion, entweder als Sender oder als Empfänger arbeiten. Dabei können mehrere Master im System existieren. Die Erzeugung des Taktes erfolgt immer durch den Master, wobei jeder Master seine eige-ne Taktfrequenz besitzt [vgl. Dem2013 S. 187 bis S. 188].

Abbildung 3.4.: Prinzipielles Aufbau des I^{2}C Datenformates



Da der Bus vom Raspberry Pimit einer Spannung von 3,3 Volt versorgt wird, befinden sich die Leitungen SDL und SCL im High-Pegel wenn keine Busaktivität stattfin-det [vgl. Kof2015 S.476]. Es hat also den logischen Wert 1. Sollen Daten übertragen werden geht das Datensignal SDA von High auf Low über, gleichzeitig befindet sich jedoch die Taktleitung auf High. Dabei handelt es sich um den Startbit. Der Startbit ist in der Abbildung 3.8 auf der linken Seite umrahmt. Im Gegensatz dazu bildet der Stoppbit einen Low zum High-Übergang des SDL bei gleichzeitigem High Pegel der Taktleitung. Dieser ist in der Abbildung 3.8 auf der rechten Seite umrahmt [vgl. rni].

Mit der Slave Adresse, die in der Regel sieben Bit groß ist, wird das jeweilige Slave Bauelement des I^2C angesprochen. Der Read/Write Bit gibt die Richtung der Datenübertragungen an. Bei Read (logische 1), liest der Slave ein und bei Write (logische 0), schreibt der Slave. Die Richtung der Datenübertragung wird vom Slave an den Master gemeldet [vgl. rni und Dem2013 S. 189].

Bei der Kommunikation über den I^2C erfolgt die Bestätigung, also der Quittierungsbit (in der Abbildung 3.8 als „Ackn." dargestellt), für die Übertragungsrichtung durch den Slave. Bei der Datenübertragung an sich bestätigt der Empfänger der Daten dem Sender den Empfang der Daten. Falls der Slave die Daten empfängt, setzt der Master den SDA auf High und erwartet vom Slave, dass dieser den SDA auf Low zieht. Geschieht das nicht, wird die Übertragung gestoppt. Im umgekehrten Fall, wenn der Master der Empfänger ist, setzt der Slave den SDA auf High, falls es nicht bestätigt wird, erfolgt der Abbruch der Datenübertragung [vgl. Dem2013 S. 189]. Die Bestätigung muss innerhalb von neun Takten des jeweiligen Masters erfolgen [vgl. rni]. Bei der Datenübertragung an sich muss der Takt selbst High sein, wenn ein Bit übertragen wird und dieser als gültig gelten soll [vgl. rni].

Auf dem Raspberry Pi existieren zwei I^2C-Schnittstellen [vgl. Upt2014 S. 223]. Der erste befindet sich in der GPIO-Schnittstelle mit der Bezeichnung P1. Der zweite bestimmt ist [vgl. Upt2014 S. 223]. Der Aufbau der zweiten I^2C Schnittstelle wird befindet sich an der GPIO Schnittstelle P5, welcher nicht für den allgemeinen Gebrauch am Anfang der Kapitels GPIO-Anschlüsse beschrieben. Die erste I^2C Schnittstelle kann über die GPIO-Schnittstelle mit der Bezeichnung P1 über zwei Pins benutzt werden. Dazu dient der Pin 3 als SDA, also als Datenleitung und der Pin 5 als SCL, also um den Takt für die Synchronisation der Kommunikation zu Übertragen. Die Pins sind mit zwei Pull-Up-Widerständen verschaltet [vgl. Dem2013 S. 188].

Der I^2C arbeitet nach dem Master-Slave Prinzip. Beim Master handelt es sich um die aktive Komponente, beim Slave um die passive. Der Master steuert also die gesamte Kommunikation, sodass die Slaves lediglich auf die Anfragen des Masters an-sprechen müssen.

3.3. Beschreibung des Aufsatzes SD0

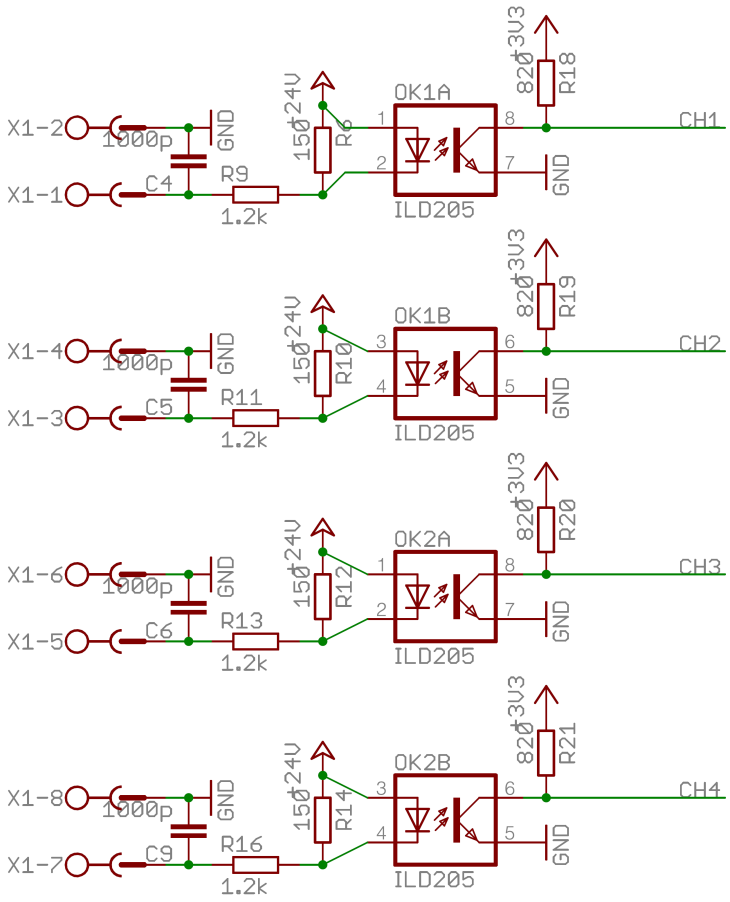
Beim Aufsatz handelt es sich um eine Erweiterung für den Raspberry Pi. Aufsätze werden bei Raspberry Pi dazu verwendet, um diesen mit zusätzlichen Funktionalitäten auszustatten oder zu erweitern.

Bei der benutzten Erweiterung handelt es sich um den Aufsatz SD0 des Unternehmens Busware. Der Aufsatz SD0 wird auf die als P1 bezeichneten General Purpose Input Output (GPIO) Pins des Raspberry Pi aufgesetzt. Diese GPIO-Leiste des Raspberry Pi wird zur Erweiterung des Raspberry Pi verwendet. Bei dieser Bachelo-rarbeit werden über den Aufsatz SD0 Energieverbrauchsdaten über dessen S0-Schnittstellen erfasst. Der SD0 wird verwendet, da dieser durch die Aufgabenstelölung vorgegeben ist. In diesem Kapitel wird der Aufsatz SD0 beschrieben. Da der aktuelle Schaltplan nicht zur Verfügung steht, wird der SD0 basierend auf dem Schaltplan *Version 1.1* beschrieben.

3.3.1. S0-Schnittstellen des SD0

Neben der Steckleiste, mit der der Aufsatz SD0 auf die GPIO Steckleiste mit der Bezeichnung P1 des Raspberrry Pi aufgesteckt wird, besitzt der Aufsatz vier S0-Schnittstellen, auch Chanels (engl. für Kanal) genannt, über die die Verbrauchsdaten erfasst werden.

Abbildung 3.5.: S0-kanäle des SD0



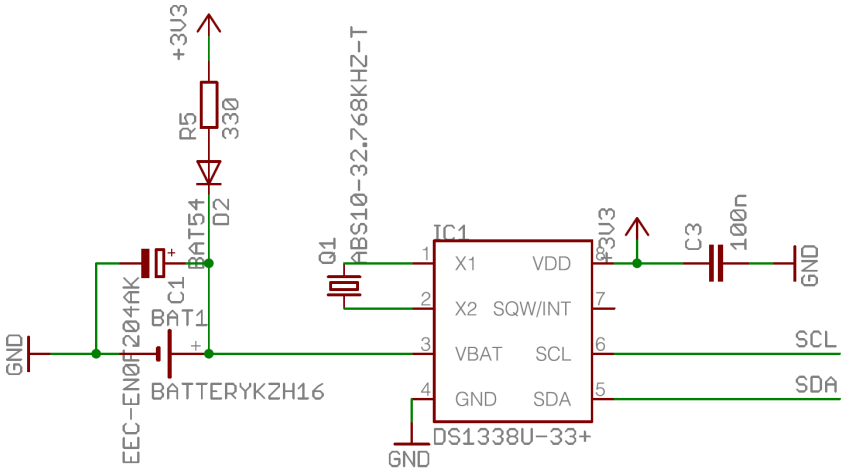
Zusätzlich befindet sich eine RJ-10-Schnittstelle sich auf dem SD0. Daneben ist die Erweiterung mit einem Mikrocontroller, dem Atmega1284p bestückt. Dieser ist mit der seriellen Schnittstelle RJ-10-Schnittstelle, den vier S0-Schnittstellen nach DIN 43864 / EN 62053-31 und der Steckverbindungsleiste verbunden. Zusätzlich besitzt die Erweiterung eine Real Time Clock (RTS), die für Echtzeitanwendungen oder hier dem Raspberry Pi zu gute kommen kann [vgl. bsd0 Specs].

3.3.2. Real Time Clock (RTC)

Bei Real Time Clock, kurz RTC bezeichnet, handelt es sich um Echtzeituhren die Systemzeiten bereitstellen. RTCs stellen also Datum und Uhrzeit bereit [vgl. Bei2011 278]. Beim Real Time Clock des SD0 handelt es sichn um den DS1338U33. Da der Raspberry Pi ohne einen Internetzugang seine Uhrzeit nicht abgleichen kann, bietet sich die Möglichkeit an, die Uhrzeit beim RTC des SD0 zu beziehen. Wobei der RTC durch den Atmega1284p dem Rasperry Pi bereitgestellt wird.

Die Verbindung zur Echtzeituhr wird über eine I^2C Verbindung erstellt. Dafür stehen wie in der Abbildung 3.1 dargestellte Leitungen SCL und SDA bereit, welche jeweils als Takt- und

Abbildung 3.6.: Echtzeituhr des SD0



Datenleitung des I^2C verwendet werden. Es bietet sowohl die Uhrzeit in den Einheiten Sekunden, Minuten, Stunden als auch den Tag, Monat und Jahr als Datum an. Die Uhrzeit wird entweder im 12 Stunden oder 24 Stunden Format bereit-gestellt. Beim 12 Stundenformat indiziert es die AM/PM [vgl. DS1338 S. 1 General Description]. Die Monatsenden werden bei Monaten mit weniger als 31 Tage aut-matisch angepasst. Es beinhaltet auch die Erkennung von Schaltjahren. Es besitzt einen Stromerkennungssensor. Es schaltet auf einen 220 mF Kondensator um, wenn es mit zu wenig Strom versorgt wird. [vgl. DS1338 S. 1]

3.3.3. Mikrocontroller des SD0

Neben dem im vorherigen Kapitel beschriebenen RTC ist der SD0 mit einem Mikrocontroller bestückt. Bei dem 8-Bit Mikrocontroller handelt es sich um den Atme-ga1284p. Der 8-Bit Mikrocontroller basiert auf dem AVR. [atm1284 S. 3] Über das Atmega1284p werden die an den S0-Schnittstellen empfangenen Daten aufbereitet und an den Ausgängen bereitgestellt [bsd0 Specs].

Wie der Raspberry besitzt der Atmega 1284p eine GPIO Schnittstelle, beim Amega1284p besitzt sie jedoch 32 Pins. Durch den Atmega1284p wird auch das RTC bereitgestellt. Der Mikrocontroller kann über zwei USART und eine SPI Schnittstelle kommunizieren. [atm1284 S. 4]

3.4. S0-Schnittstelle

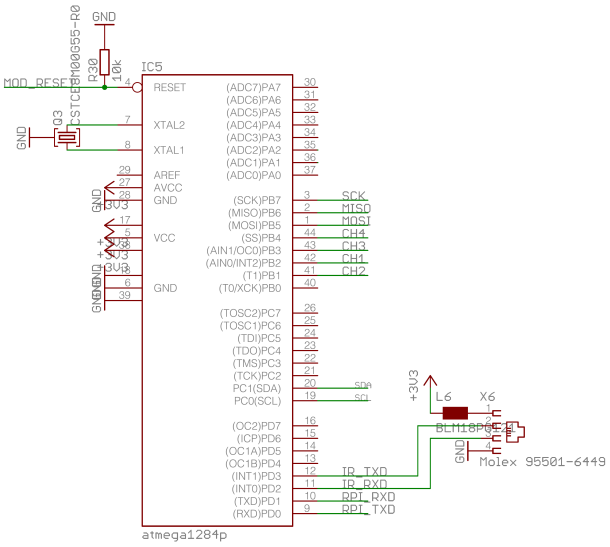
Die S0-Schnitstelle wird nach *DIN EN 62053-31 definiert* und in der Gebäudeautomatisierung verwendet. Die Schnittstelle wird in Elektrizitätszählern eingebaut. Mit Impulseinrichtungen wie die S0-Schnittstelle werden Impulse, die der aktuell verbrauchten Energiemenge entsprechen, an den Empfänger übertragen [**[]**s0norm S.4].

Die Impulseinrichtungen geben eine bestimme Anzahl von Impulsen pro Wattstunde. Dabei gibt es zwei Arten bzw. Klassen von Impulsausgängen [s0 norm S.5]:

Impulsausgang der Klasse A: Übertragung über größere Entfernungen;

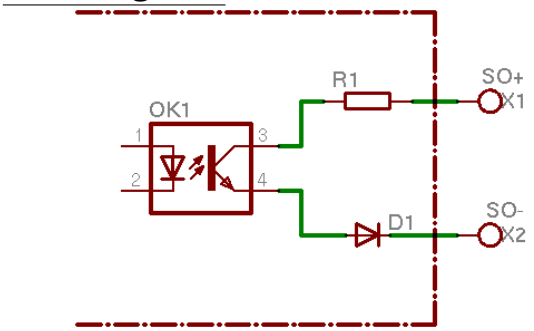
Impulsausgang der Klasse B: geringe Entfernung und geringer Energieverbrauch

Abbildung 3.7.: Mikrokontroller des SD0



(Quelle: [**sad**])

Abbildung 3.8.: S0-Schnitstelle in Stromzählern



Dabei kann die Schnittstelle sowohl im Stromzähler als auch im Gas- oder Wasserzähler eingesetzt werden. Bei der Übertragung von Messwerten wird für jedes verbrauchtes Watt pro Stunde (Wph) ein gewichteter Impuls übertragen. Die Gewichtung des Impulses ist von Zähler zu Zähler unterschiedlich [vgl. S0S]. Für die zwei Arten bzw. Klassen von Impulsausgängen sind den Impulsen verschiedene Grenzen gesetzt:

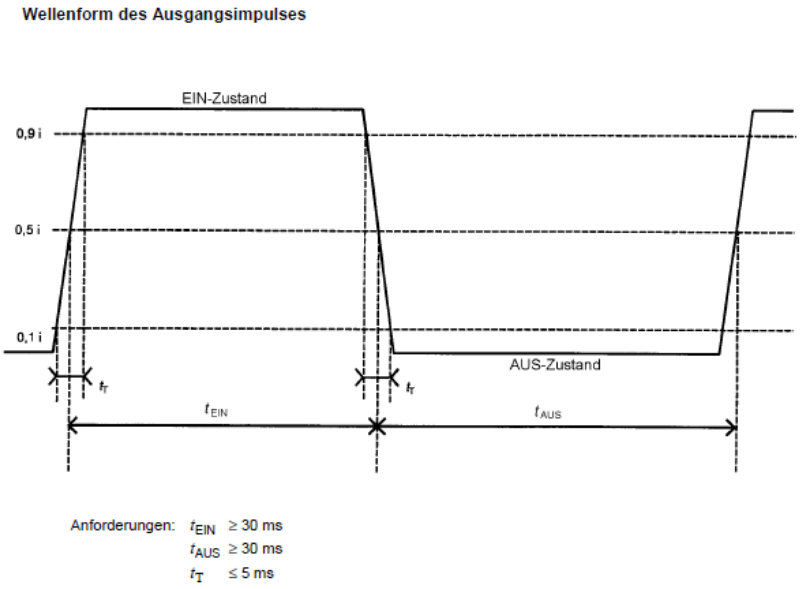
Tabelle 3.6.: S0-Schnitstelle in Stromzählern

Parameter	Impulseinrichtung der Klasse A	Impulseinrichtung der Klasse B
Maximale Spannung U_{max}	27 V DC	15 V DC
Maximaler Strom im EIN-Zustand	27 mA	15 mA
Minimaler Strom im EIN-Zustand	10 V DC	2 V DC
Maximaler Strom im Aus-Zustand	2 V DC	0,15 V DC

(Quelle: asd)

Die Impulse können zwei Zustände haben: High- bzw. Ein- Zustand und Low- bzw.- Aus-Zustand, für zwei aufeinanderfolgende Impulse.

Abbildung 3.9.: S0-Impulsform



(Quelle: [asd])

Für die Zustände sind die folgenden Impulslängen definiert:

Ein Impuls, im High- bzw. Ein- Zustand, ist mit $t_{EIN} \geq$ 30 ms, also größer gleich 30 ms definiert und der

Aus- bzw- Low-Zustand, für die Zeit zwischen zwei aufeinanderfolgenden Impulsen, mit $t_{AUS} \geq$ 30 ms, also größer gleich 30 ms

Dabei muss die Übergangszeit der Anstiegs- oder Abfallzeit zwischen den einzelnen Zuständen, also der Zustandswechsel 5 ms sein.

3.5. Datendarstellung

Bei der Datenhaltung, also der Datenspeicherung, gibt es verschiedene Möglichkeiten: die Daten können beispielsweise als CSV, also als Comma Separated Values, abgespeichert werden. Es gibt jedoch auch die Möglichkeiten die Daten in JSON oder aber auch in SQL abzulegen. Die Daten in SQL abzuspeichern lohnt sich erst, wenn die Daten auch lokal auf dem Raspberry Pi weiterverarbeitet werden sollen. Dabei muss auch bedacht werden, dass ein SQL-Server benötigt wird. Dies würde dazu führen, dass durch den SQL-Server mehr Ressourcen verwenden würden. Das heißt ein SQL-Server würde nicht nur Speicher auf der SD-Karte einnehmen, sondern auch, wenn der Server läuft, den Arbeitsspeicher und die CPU beanspruchen.

Beim JSON ist bei der Programmierung des Quellcodes, die auf dem Raspberry Pi läuft, nur ein Libary nötig, der die Daten in eine. JSON-Datei schreibt. JSON wird meist im Web Bereich eingesetzt.

3.5.1. Comma Separated Values (CSV)

emeinen Standard oder bestimmte Spezifikation, sondern es kann in unterschiedlichen Spezifikationen eingesetzt werden. Das CSV-Format kann dann eingesetzt, wenn die abgespeicherten Daten unabhängig von der Technologie eingesetzt werden sollen. Also wenn die Daten beispielsweise in einem Tabellenprogramm dargestellt und in einem Quellcode bearbeitet werden sollen [**RFC4180**].

Die Daten können entweder durch einfache Kommata voneinander getrennt in der .csv Datei abgelegt werden, die Werte an sich können zusätzlich in Anführungsstriche bzw -zeichen gesetzt werden. Genauso könnten die einzelnen Daten mit Semikolon voneinander getrennt abgespeichert werden [**RFC4180**]. Zu beachten ist jedoch, dass bei Kommazahlen im deutschen ein Komma benutzt wird und im englischsprachigen Raum ein Punkt. Dies könnte evtl. zu Problemen beim Lesen bzw. Parsen der Datei im Quellcode führen.

3.5.2. JavaScript Object Notation (JSON)

JavaScript Object Notation, kurz JSON, ist ein Datenaustauschformat. Dabei handelt es sich um ein von Programmiersprachen unabhängiges Textformat. Daten können also programmiersprachenunabhängig ausgetauscht werden. JSON folgt jedoch der JavaScript Notation. Dabei handelt es sich bei JSON um eine Untermenge der Skriptsprache JavaScript [**ecma2013**].

Das Datenaustauschformat basiert dabei auf zwei Strukturen [**ecma2013**]:

Name-Wert-Paare
: Dabei ist jedem Namen bzw. Bezeichnung mindestens ein Wert zugewiesen.

Geordnete Liste von Werten
: Neben den beiden Strukturen gibt es in JSON, wie auch in Programmiersprachen, Typen. Die Typen beziehen sich auf die Werte. Die folgende Auflistung mit entsprechenden Zeichen [vgl ecma2013]:

Objekte
: *Objekte* beinhaltet eine ungeordnete Menge an Name-Werte-Paare. Es beginnt mit einer öffnenden geschweiften Klammer „„ und einer schließenden geschweiften Klammer „“. Die Name-Werte-Paare sind mit einem Doppelpunkt getrennt [vgl. ecma2013]:

Name:Wert
: Jedes Paar ist wiederum zueinander mit Kommata (,) voneinander getrennt [vgl. ecma2013].

Arrays
: Ein Array beinhaltet eine geordnete Liste von Werten, die einem Namen zugewiesen werden. Arrays beginnen mit einer [(öffnenden eckigen Klammer) und enden mit einer] (schließenden eckigen Klammer). Die Werte an sich sind mit , (Komma) zu einander getrennt [vgl ecma2013].

Werte
: Werte werden durch Objekte, Arrays, Zeichenketten (Strings), Zahlen, den Wahrheitswerten *true* und false oder Null gebildet [vgl ecma2013].

Zeichenketten
: Strings, also Zeichenketten, sind Unicode Zeichen [vgl ecma2013].

Nummern
: Bei den Zahlen in JSON handelt es sich um Dezimalzahlen. Den Zahlen kann ein - (Minus) vorangestellt werden. Gefolgt von entweder einem 0 (Null) oder einer beliebigen Zahl von 1 bis 9 in beliebiger Weiderholung. Gleitkommerzahlen werden mit einem Punkt getrennt [vgl ecma2013]. [**ecma2013**]

Im Folgenden ist ein beispielhafter Aufbau einer JSON-Datei dargestellt:

Listing 3.1: Beispielhafter Aufbau einer JSON-Datei

```
{
    "`books"': [
        "`book"': {
            "`language"':"'Java"',
            "`edition"':"'second"',
            "`price"': 10,
            "`author"': ["`Hans"', "`Peter"', "`Müller"']
        },

        "`book"': {
            "`language"':"'C++"',
            "`lastName"':"'fifth"'
            "`price"': 11.12,
            "`author"': ["`Stroustroup"']
        },

        "`book"': {
            "`language"':"'C"',
            "`lastName"':"'third"'
            "`price"': 8,
            "`author"': ["`Georg"', "`Petrus"']
        }
    ]
}
```

(Quelle: [**asd**])

4. Anforderungsdefinition

Für das Monitoring von Energieverbrauchswerten soll eine Softwarelösung entwickelt werden. Es soll über die S0-Schnittstellen von Zählern Energieverbrauchswerte erfassen, anschließend verarbeiten und lokal abspeichern.

Die abgespeicherten Verbrauchswerte sollen an einen Server gesendet werden.

Die Visualisierung der Verbrauchswerte erfolgt grafisch über eine Weboberfläche. Dabei werden die Energieverbrauchswerte entweder für einen bestimmten, vom Be-nutzer ausgewählten, Zeitraum abgefragt. Oder die aktuellen Verbrauchswerte werden fortlaufend von der Messeinheit, also vom Aufsatz SD0 und dem Raspberry Pi abgefragt und auf dem Raspberry Pi dargestellt.

Die Verbrauchswerte werden über die S0-Schnittstellen des Aufsatzes SD0 gemessen. Die gemessenen Daten werden anschließend vom Aufsatz über die serielle Schnittstelle, also dem UART, des Raspberry Pi an diesen übertragen. Auf dem Raspberry Pi werden die S0-Impulse verarbeitet und anschließend mit dem jeweiligen Zeitstempel des Zeitpunktes der Messung versehen und abgespeichert.

Die abgespeicherten Verbrauchswerte werden anschließend an einen Server weitergeleitet, wo die Verbrauchswerte für einen, vom Benutzer bestimmten, Zeitraum grafisch dargestellt werden. Der Server kann sich physisch an einem anderen Ort als der Raspberry Pi befinden. Dabei erfolgt die Kommunikation über die Ethernet Schnittstelle. Alternativ kann die Webdarstellung auf dem Raspberry erfolgen. Für die fortlaufende grafische Darstellung des Energieverbrauchs muss sich die Webdarstellung auf dem Raspberry Pi befinden.

4.1. Muss-Kriterium

M-ERF010 Erfassen der S0-Impulse am Zähler
Durch S0-Impulse soll der Verbrauch am Stromzähler erfasst werden.

M-ERF010 -10 Anzahl der verschiedenen, gemessenen Stromzählern
Mit dem Aufbau muss möglich sein, an bis zu vier verschiedenen Stromzähler Stromzählern mit S0-Schnittstellen die S0-Impulse erfasst zu können.

M-ERF020 Übergabe der erfassten S0-Impusle an den Raspberry Pi
Die erfassten S0-Impulse werden nach dem Messvorgang dem Raspberry Pi über die GPIO-Schnittstelle übergeben.

M-VERA030 Umrechnung der übergebenen S0-Impusle
Die gemessenen S0-Impusle, die an den Raspberry Pi übergeben wurden, müssen auf dem Raspberry Pi in Energieverbrauchswerte (Watt pro Stunde) umgerechnet werden.

M-VERA040 Abspeichern von umgerechneten Energieverbrauchswerten.
Die von S0-Impuslen nach Energieverbrauchswerte umgerechneten Werte müssen auf dem Raspberry Pi abgespeichert werden.

M-SEN050 Senden der abgespeicherten Energieverbrauchsdaten.
Alle abgespeicherten Dateien sollen an einen Server versendet werden können.

M-SEN060 -10 Zeiten für das Senden der abgespeicherten Energieverbrauchsdaten
Die abgespeicherten Dateien sollen zu bestimmten Uhrzeiten an einen Server versendet werden können.

M-SEN070 -11 Zeiten für das Senden der abgespeicherten Energieverbrauchsdaten
Die abgespeicherten Dateien sollen permanent an einen Server versendet werden können.

M-BEA080 Betriebsart der Software
Die Software muss als Dienstprogramm laufen.

M-BEA090 Dauerhafter Betrieb der Software
Die Software muss dauerhaft auf dem Raspberry Pi laufen.

M-BEA0100 Starten der Software
Die Software muss mit dem Systemstart der Raspberry Pi bzw. dessen Betriebssys-tems gestartet werden.

M-BEA0110 Beenden der Software
Die Software muss vom Benutzer durch Angabe der Prozessnummer der Software beendet werden können.

M-BEA0120 Neustart der Software
Die Software muss vom Benutzer neugestartet werden können.

M-KONF0130 Einlesen einer Konfigurationsdatei
Die Software muss während der Laufzeit eine Konfigurationsdatei einlesen können.

M- KONF0140 Einlesen der Impulskonstante einer Konfigurationsdatei
Die Software muss während der Laufzeit der Impulskonstante von bis zu vier unter-schiedlichen Stromzählern aus einer Konfigurationsdatei einlesen können.

M- KONF0150 Einlesen der Empfängeradresse aus einer Konfigurationsdatei
Die Software muss während der Laufzeit die IP-Adresse, an die die gespeicherten Energiever-brauchswerte gesendet werden sollen, aus einer Konfigurationsdatei ein-lesen können.

M- KONF0160 -10 Einlesen der Empfängeradresse aus einer Konfigurationsdatei
Wird für die IP-Adresse, an denen die Daten gesendet werden sollen als Localhost oder 127.0.01 angegeben, verbleiben die Daten auf dem Raspberry Pi. Bei der Ad-resse handelt es sich um den Raspberry Pi selbst [vgl. loc].

M-KONF0170 : Einlesen einer Konfigurationsdatei Die Software muss während der Laufzeit eine Konfigurationsdatei einlesen können.

M- KONF0180 : Einlesen der Impulskonstante einer Konfigurationsdatei Die Software muss während der Laufzeit der Impulskonstante von bis zu vier unter-schiedlichen Stromzählern aus einer Konfigurationsdatei einlesen können.

M- KONF0190 : Einlesen der Empfängeradresse aus einer Konfigurationsdatei Die Software muss während der Laufzeit die IP-Adresse, an die die gespeicherten Energieverbrauchswerte gesendet werden sollen, aus einer Konfigurationsdatei ein-lesen können.

M- KONF0200 -10: Einlesen der Empfängeradresse aus einer Konfigurationsdatei Wird für die IP-Adresse, an denen die Daten gesendet werden sollen als Localhost oder 127.0.01 angegeben, verbleiben die Daten auf dem Raspberry Pi. Bei der Ad-resse handelt es sich um den Raspberry Pi selbst [vgl. loc].

M-DAR0210 : Darstellung der umgerechneten Energieverbrauchswerte Dem Benutzer sollen die umgerechneten Energieverbrauchsdaten grafisch darge-stellt werden.

M-DAR0220 -10: Darstellung der umgerechneten Energieverbrauchswerten Die grafische Darstellung der umgerechneten Energieverbrauchsdaten muss auf Webbasis geschehen.

M-DAR0230 : Grafische Darstellung der umgerechneten Energieverbrauchswerten Die grafische Darstellung der umgerechneten Energieverbrauchsdaten muss auf ei-nem Koordinatensystem geschehen.

M-DAR0240 : Auswahl der Zähler für die Darstellung der Energieverbrauchswerte Der Benutzer soll für die grafische Darstellung die einzelnen Zähler auswählen kön-nen. Die Verbrauchswerte der ausgewählten Zähler werden dann auf einem Koordi-natensystem dargestellt.

M-DAR0250 -10: Auswahl mehrerer Zähler für die Darstellung der Energieverbrauchswerte Der Benutzer soll für die grafische Darstellung mehrere Zähler gleichzeitig auswäh-len können. Die Verbrauchswerte der ausgewählten Zähler werden dann gleichzeitig grafisch auf einem Koordinatensystem dargestellt.

M-DAR0260 -20: Auswahl aller Zähler für die Darstellung der Energieverbrauchswerte Der Benutzer soll für die grafische Darstellung alle Zähler auswählen können. Die Verbrauchswerte aller Zähler werden, auf einem Koordinatensystem dargestellt.

M-DAR0270 : Auswahl des Zeitraumes für die Darstellung der Energieverbrauchswerte Der Benutzer muss für die grafische Darstellung den Zeitraum auswählen können, in dem die Verbrauchsdaten angezeigt werden sollen.

M-DAR0280 -10: Auswahl des Startdatums für die Darstellung der Energieverbrauchswerte Der Benutzer muss für die grafische Darstellung das Startdatum für den Zeitraum Auswählen können, bei dem die Verbrauchswerte angezeigt werden sollen, die ab dem ausgewählten Startdatum abgespeichert wurden.

M-DAR0290 -20: Auswahl des Enddatums für die Darstellung der Energieverbrauchswerte Der Benutzer muss für die grafische Darstellung das Enddatum für den Zeitraum auswählen

können, bei dem die Verbrauchswerte angezeigt werden sollen, die bis zum dem ausgewählten Enddatum abgespeichert wurden.

M-DAR0300 -30: Keine Auswahl des Datums für die Darstellung der Energieverbrauchswerte Ein bestimmtes Datum kann nicht ausgewählt werden, wenn für dieses Datum keine Energieverbrauchsdaten abgespeichert wurden.

4.2. Kann-Kriterien

K-DAR010 Darstellung der aktuellen Energieverbrauchswerte
Dem Benutzer kann die Möglichkeit gegeben werden, die Energieverbrauchswerte darstellen zu lassen, die aktuell verbraucht werden.

K-DAR020 Export der Verbrauchsdatendarstellung in Bilddatei
Dem Benutzer kann die Möglichkeit gegeben werden, die von diesem betrachtete grafische Darstellung der Energieverbrauchsdaten in eine Bilddatei zu exportieren.

4.3. Technische Kriterien

MT030 : Den Aufsatz SD0 benutzen Für die Messung der S0-Impulse muss der Aufsatz SD0 von Busware benutzt wer-den.

MT040 : Datenempfang vom Aufsatz SD0 Die gemessenen S0-Impulse sollen vom Aufsatz SD0 von Busware über die serielle Schnittstelle empfangen werden.

5. Entwurf

5.1. Anwendungsfälle der zu entwickelnden Software

6. Realisierung

In diesem Kapitel wird die Realisierung der prototypischen Entwicklung beschrieben.

Um die Verbrauchswerte zu messen gibt es schon ein Projekt, den Volkszähler. Da-bei handelt es sich um ein Open Source Projekt [vgl. vza]. Mit dem Volkszähler kön-nen Energieverbrauchswerte gemessen und auf dessen Darstellung Verbrauchswer-te von Wasser, Strom Gas usw. angezeigt werden [vgl. Zoe], mit dem die Nutzer die eigenen Verbrauchdaten wie Strom, Temperatur, Wasser am Zähler messen können. Die gemessenen Verbrauchsdaten werden dann über die Schnittstellen an das Messgerät übertragen, dort gespeichert und auf einer Website ausgewertet. Ein ge-eignetes Messgerät muss vorhanden sein. In dieser Arbeit handelt es sich dabei um den SD0 von Busware, zusammen mit dem Raspberry Pi. Die Messung findet dabei am Zähler über die S0-Schnittstelle statt. Die Übertragung findet über Kupferkabel statt. Die Speicherung und Auswertung finden dann durch die Software statt. Bei der Software handelt es sich um den Volkszähler. Mit dem Volkszähler können auch Verbrauchsdaten über S0-Schnittstellen gemessen werden [vgl. vzg]. Dazu gibt es zwei Softwares, um S0-Impulse zu messen. Dabei handelt es sich um s0vz [vgl. s0vz] und s0enow. Aus dem Quellcode bzw. der Datei s0enow.cpp, in der sich auch die Main-Funktion befindet, geht hervor, dass sich beim Projekt s0enow um einen nach GNU General Public Licence der Version 3 Lizenzierte Software handelt. Das bedeutet, dass der Quellcode verwendet bzw. Modifiziert werden darf. Dabei muss aber die Software, die Teile der Software, die nach GNU General Public Licence der Version 3 lizenziert wurde, verwendet auch als solche lizenziert werden [vgl. s0ec Codezeilen 23 bis 38] [**s0ec**].

In dieser Arbeit werden einige Funktionen, die im Softwareprojekt s0enow vorhanden sind, verwendet. Dadurch müssen diese nicht von neuem entwickelt werden. Neben der Bibliothek, die zum Einlesen der Konfigurationsdatei dient, werden ähnliche Da-ten eingelesen, wie im Projekt s0enow. Dies ist der Tatsache geschuldet, dass die Software die in dieser Arbeit entwickelt wird, eine ähnliche Aufgabe erfüllt, wie das Projekt s0enow. Dabei handelt es sich um die Funktion cfile(). Diese Datei befindet sich in den Codezeilen 204 bis 309 im s0enow Projekt [vgl. s0ec]. In der Software, die in dieser Arbeit entwickelt wird, werden neben dem Datafolder, die Messstellenname, die Mittelwertszeit auch die Impulskonstanten ausgelesen.

Zusätzlich gibt es Überschneidungen mit den Projekten s0enow und s0vz. Wie in diesen Projek-ten wird die Software, die auf dem Raspberry Pi laufen soll, als Dienst-programm bzw. Deamon entwickelt So genannte Deamons laufen unter Linux im Hintergrund und starten, wenn das Be-triebssystem hochgefahren wird [vgl. Wol2006 dpz und Wol2006 af]. Diese Funktionen befinden sich in den Codezeilen 111 bis 202, im Projekt s0enow [vgl. s0ec]. Bei den Funktionen handelt es sich um die drei fol-genden Funktionen,

7. Inbetriebnahme

7.1. Zielbeschreibung

8. Test

9. Zusammenfassung und Ausblick

In dieser Abschlussarbeit wurde eine, aus zwei Bereichen bestehende, Softwarelösung entwickelt, die den eigenen Energieverbrauch visualisiert und dadurch die Möglichkeit aufgreift den eigenen Energieverbrauch zu analysieren. Die auf der Grundlage eines Linux-Betriebssystems entwickelte Software, welche auf einem Raspberry Pi läuft, wertet die empfangenen S0-Impulse in Energieverbrauchswerte um. Auf der Grundlage, von diesen als Linuxtreiber (auch Deamon bezeichnet) entwickelten Software ausgewerteten Verbrauchswerten, findet die Visualisierung, der ausgewerteten Energieverbrauchswerte statt. Die Visualisierungseinheit, also die Webseite befindet sich dann auf einem Webserver, der entweder auf einem Raspberry Pi oder einem vom Raspberry Pi physisch unabhängigen Server läuft. Die entwickelte Visualisierungseinheit bietet den Vorteil, dass nicht nur der bisherige Gesamtverbrauch, vom Zähler als Zahl abgelesen werden kann, sondern die Verbrauchswerte für einen bestimmten Zeitraum grafisch darzustellen werden. Zudem ist es durch die Auswahl der einzelnen Kanälen, die der Anzahl der S0-Anschlüssen der Erweiterung SD0 entsprechen, möglich die Verbrauchswerte verschiedener Zähler, gleichzeitig zu analysieren und miteinander zu vergleichen. Es wurde eine Anforderungsanalyse durchgeführt, die die Anforderungen an die entwickelte Softwarelösungen definieren soll. Mit deren Hilfe wurden dann im anschließenden Kapitel die Lösungen entworfen. Ausgehend vom Entwurf, der die Grundlage bzw. den Ausgangspunkt der realisierten Lösungen bildet, wurde die Lösung realisiert. Es wurde auch ein Parser zum Einlesen von Konfigurationsdateien verwendet. Dadurch kann die Software während der Laufzeit an verschiedene Zähler angepasst werden. Die Software, die auf dem Raspberry Pi läuft, wurde modular aufgebaut. Dadurch ist später eine einfache Erweiterung der schon entwickelten Software möglich.

Der Vorteil bei der in dieser Abschlussarbeit eingesetzten S0-Schnittstelle ist, dass die Schnittstelle nicht nur für den Energieverbrauch, sondern auch für andere Verbrauchswerte wie Wärme, Wasser und Gas verwendet werden kann. Die entwickelte Software kann also in Verbindung mit den S0-Schnittstellen vielseitig verwendet werden.

Eine Möglichkeit den Prototyp zu erweitern wäre, weitere Darstellungsformen und -varianten hinzuzufügen. Das wäre beispielsweise die Darstellung der Verbrauchsdaten in einer App, also auf einem Smartphone oder Tablet. Es wäre auch möglich, weitere Daten, wie Tarifinformationen und bisherige Kosten für den Energie-verbrauch, darzustellen. Es wäre auch möglich, die aktuellen Energieverbrauchswerte mit den Verbrauchswerten des selben Zeitraums vergangener Jahre zu vergleichen.

Tabellenverzeichnis

Abbildungsverzeichnis

Listings

Literatur

Anhang

A. Schaltplan des Raspberry Pi Model B

B. Schaltplan des SD0

C.
Hilfsfunktionen zum zeichnen der Energieverbrauchsdaten

D. Inhalt und Struktur der Beiliegenden CD