

|  |
| --- |
| **Berrycase** |
| Berufsübergreifendes Projektarbeit |

Eine Dokumentation über die Entwicklung ein multifunktionales Gehäuse für den Einplatinencomputer Raspberry Pi 4B.

Autor: Enrico Cirignaco  
Fachvorgesetzte: Matthias Burri   
Auftraggeber: Matthias Burri 11.12.20

**Abstract**

Die im nachfolgenden Dokument beschriebene Arbeit wurde im Ramen einem Berufsübergreifendes Projektarbeit (BüP) durchgeführt:

Die Aufgabe lautete, ein Gerät zu entwickelt, dass ermöglicht unterwegs mit dem Raspberry zu arbeiten und die Allgemeine Produktivität der Benutzer zu steigern. Daher muss eine ansteckbarer Leiterplatte und ein Gehäuse entwickelt werden. Im Gehäuse muss einen Akku integriert werden, sodass das Gerät auch ohne Netzspeisung benutzt werden kann.

Als Vorarbeit zu dieser Aufgabe wurde eine kleine Marktanalyse gemacht, um zu wissen ob ähnliche Lösungen auf dem Markt schon vorhanden sind. Dazu wurde auch die technische Machbarkeit untersucht.

Das Gehäuse ist mit einem OLED Display und ein Navigationsknopf ausgestattet. Damit können Einstellungen gemacht werden und Systeminformationen eingeblendet werden. Um der Rechner Kühl zu behalten wurde im Gerät ein 20mm Lüfter eingebaut. Die Geschwindigkeit des Lüfters kann mit einem automatischen oder einem manuellen Modus angesteuert werden. Die Kerneigenschaft vom berrycase ist die UPS Funktion (aus dem English Unterbrechungsfreie Stromversorgung). Dies ermöglich vom Akkubetrieb auf Netzbetrieb (und umgekehrt) zu wechseln, ohne das System herunterfahren zu müssen.

Am Ende der Arbeit wurden alle grundlegenden Ziele erreicht und es konnte sogar angefangen am optionalen Ziel zu arbeiten. Diese wurde aber nicht abgeschlossen. Verbesserungen, die noch implementiert werden könnten, wären das 3D Gehäuse fertig entwickeln und ausdrucken. Zudem könnte eine Grössere Akku eingebaut werden. Das Kühlungssystem könnte auch so erweitert werden, dass die CPU/GPU übertackt werden kann.

**Inhaltsverzeichnis**

[1 Einleitung 8](#_Toc58238191)

[1.1 Aufgabestellung 8](#_Toc58238192)

[1.2 Vorgehen 8](#_Toc58238193)

[1.3 Zielsetzung 9](#_Toc58238194)

[1.3.1 Grundfunktionen 9](#_Toc58238195)

[1.3.2 Optional 9](#_Toc58238196)

[2 Hardware 10](#_Toc58238197)

[2.1 Konzept 10](#_Toc58238198)

[2.1.1 Mikrokontroller 10](#_Toc58238199)

[2.1.2 OLED Display 10](#_Toc58238200)

[2.1.3 UPS & BMS 10](#_Toc58238201)

[2.1.4 Soft-Power OFF 11](#_Toc58238202)

[2.1.5 Batterie-Spannungsmessung 11](#_Toc58238203)

[2.2 Realisierung 11](#_Toc58238204)

[2.2.1 Blockschaltbild 11](#_Toc58238205)

[2.2.2 Mikrokontroller 11](#_Toc58238206)

[2.2.3 Speisung 12](#_Toc58238207)

[2.2.4 Batterie Spannung Messung 13](#_Toc58238208)

[2.2.5 Soft Power ON/OFF Schaltung 14](#_Toc58238209)

[2.2.6 Lüfter Steuerung 15](#_Toc58238210)

[2.2.7 Bedienung Komponenten 15](#_Toc58238211)

[3 Software 16](#_Toc58238212)

[3.1 Node-RED 16](#_Toc58238213)

[3.1.1 Wake-up Light 16](#_Toc58238214)

[3.1.2 Light Control 19](#_Toc58238215)

[3.1.3 Resources 21](#_Toc58238216)

[3.1.4 Control Panel 22](#_Toc58238217)

[3.1.5 Power Usage 23](#_Toc58238218)

[3.1.6 Plattform abhängiges Interface 24](#_Toc58238219)

[3.1.7 Applikation URL 25](#_Toc58238220)

[3.1.8 WPS Button 26](#_Toc58238221)

**Abbildungsverzeichnis**

Abbildung 1: USB Type-C und Schutzschaltung 12

Abbildung 2: Ausschnitt aus dem PDF «Introduction to USB Type-C™» 12

Abbildung 3: PowerBost 1000c 13

Abbildung 4: Spannungsmessung 13

Abbildung 5: : Ausschnitt aus dem Datenblatt vom TPS61090 14

Abbildung 6: Ausschnitt aus dem Datenblatt vom NC7SZ74K8X 14

Abbildung 7: Soft ON/OFF Schaltung 14

Abbildung 8: Lüfter Steuerung Schaltung 15

Abbildung 9: Bedienung Komponenten Schaltung 15

**Abkürzungsverzeichnis**

**IPA** Individuelle Projektarbeit

**OLED** Organic Light Emitting Diode

**UPS** Uninterruptible Power Supply

**CPU** Central Processing Unit

**GPU** Graphics Processing Unit

**UX**  User Experience

**IP** Internet Protocol

**GPIO** General-purpose input/output

**BüP** Berufsübergreifendes Projekt

**BMS** Battery management system

**AP** Access Point

**I2C** Inter-Integrated Circuit

**PCB** Printed Circuit Board

**PWM** Pulse-Width Modulation

**RS-Flip-Flop** Reset Set Flip Flop

**D-Flip-Flop** Data Flip Flop

**MOSFET** Metal Oxide Semiconductor Field Effect Transistor

**kV** Kilovolts

**BVDSS** Drain-Source Breakdown Voltage

**VDS**  Drain-Source Voltage

**SoC** System on a chip

**PIR** Passive Infrared Sensor

**TVS** Transient-voltage-suppression diode

**PTC** Positive Temperature Coefficient  
**DC** Direct Current  
**AC** Alternating Current **ADC** Analog-Digital-Converter  
**USB** Universal Serial Bus  
**I/O** Inputs/Outputs  
**LED** Light Emitting Diode  
**uC** Mikrocontroller  
**GND** Ground  
**V** Volt  
**A** Ampere  
**IC** Integrated Circuit  
**F** Farad  
**GUI** Graphical User Interface  
**RC** Widerstand **+** Kondensator

# Einleitung

Im Rahmen einer Vorbereitung für die zukünftige Individuelle Projektarbeit soll dieses Projekt durchgeführt werden.

## Aufgabestellung

Der Raspberry Pi unterwegs zu benutzen ist sehr umständlich. Ein Powerbank ist nötig und ein Netzwerk muss aufgesetzt werden. Um die UX (Nutzererfahrung) zu steigern muss ein Gehäuse für der Einplatinencomputer entwickelt und gebaut werden.

Das Gehäuse muss klein gehalten werden. Eine UPS Funktion muss implementiert werden. Man muss Minimum zwei Stunden im Akkubetrieb arbeiten können. Das Gehäuse muss mit einem Kühlungssystem für die CPU und einem kleinen Display, um die IP-Adresse anzuzeigen ausgestattet werden. Die GPIOs von Raspberry müssen zugänglich bleiben.

## Vorgehen

Während der Dauer der BüP soll das Konzept erarbeitet werden, die Elektronik entwickelt und gefertigt werden.

Zusätzlich muss die Software geschrieben werden. Folgende Arbeiten gehören zur BüP:

• Grobplanung

• Vorstudie

• Konzept erarbeiten

• Hardware Entwicklung

• Firmware / Software Entwicklung

• Testing / Bugfixing

• Dokumentation

Das Projekt wird mit der „Agilen Arbeitsmethodik“ geleitet. Dementsprechend wird in der Startphase nur eine Grobplanung gemacht und die Meilensteine werden festgesetzt. Ein SCRUM Board und das Project Management Tool „Trello“ werden eingesetzt.

Das Projekt wird mit GIT verwaltet und auf Github dokumentiert. Alle Projektunterlagen sind Open-Source. Das Arbeitsjournal wird elektronisch geführt.

## Zielsetzung

Das Projekt wurde in zwei Zielbereiche unterteilt. In einen grundlegenden und in einen optionalen Zielbereich.

### Grundfunktionen

* Batteriebetrieb (Minimum zwei Stunden Autonomie)
* Aktives Kühlungssystem
* Einschaltknopf
* Ausschaltknopf (mit soft Shutdown)
* Unterbrechungsfreie Stromversorgung (UPS)
* Batteriemanagementsystem (BMS)
* Die GPIOs vom Raspberry müssen zugänglich bleiben
* Das Gehäuse darf maximal doppelt so gross wie der Raspberry sein
* Display, um Informationen anzuzeigen
* Stromversorgung über USB Type-C

### Optional

Die optionalen Ziele sollen erst realisiert werden, sobald die Grundfunktionen alle einwandfrei funktionieren. Optional soll mithilfe eines CAD Tools ein passendes Kunststoff Gehäuse gezeichnet und mit einem 3D Drucker ausgedruckt werden.

# Hardware

## Konzept

In diesem Abschnitt sollen die einzelnen Schaltungsteile analysiert und evaluiert werden. Hierbei ist darauf zu Achten, dass einige Teile der Schaltung von vorhergegangenen und Open Source Projekten übernommen wurden und daher nicht überall neue Evaluationen durchgeführt wurden.

### Mikrokontroller

Die Funktion der Mikrokontroller übernimmt in diesem Fall Der Raspberry Pi selber. Der Mikrokontroller dient in dieser Anwendung zur Ansteuerung der einzelnen Hardwarekomponenten und als Router für das aufgesetzte Netzwerk.

#### Vorgaben

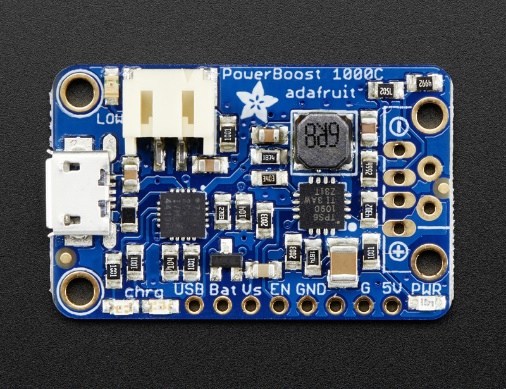
Der Kontroller muss folgende Punkten erfüllen:

* Auf dem Kontroller muss einen Wi-Fi AP (Access Point) aufgesetzt werden können
* PWM zu Ansteuerung der Lüfter
* I2C zu Ansteuerung des Displays und der ADC
* Es werden insgesamt 7 I/Os (Inputs/Outputs) benötigt, um die Hardwareteile anzusteuern.

### OLED Display

Um nutzvolle Informationen anzuzeigen wurde entschieden ein OLED Display einzusetzen. Die Hauptvorteile diese Technologie sind, dass keine Hintergrundbeleuchtung benötigt wird und seine hervorragender Kontrast Eigenschaft. Dazu sind diese Displays preisgünstig und Stromsparender. Der ausgewählte OLED Display hat eine bilddiagonale von 0.91 Zoll und eine Auflösung von 128x32 Pixels. Die Anzeige kann mit 3.3v und 5v gespienes werden. Angesteuert wird sie über das I2C Protokoll.

### UPS & BMS

Auf dem Markt gibt es schon eine Lösung, die auf ein winzige PCB implementiert genau was in diesem Projekt benötigt wird. Der PowerBoost 1000c von der Hersteller Adafruit. Die Entwicklungsunterlagen von diesem Produkt sind freizugänglich und Open-Source. Es wurde zuerst evaluiert ob es Sinnvoll wäre die Schaltung auf die Eigene Leiterplatte zu integrieren oder das Produkt von Adafruit zu übernehmen. Zwei kritischen Bauteilen der Schaltung können nur mit speziellen Werkzeugen bestückt werden. Dazu der Preisunterschied zwischen den zwei Varianten ist nicht sehr gross. Es wurde darum entschieden die fertige Leiterplatten von Adafruit einzukaufen und im Projekt zu integrieren.

### Soft-Power OFF

Der Raspberry muss mithilfe der Benutz Oberfläche ausgeschaltet werden können. Nachdem der Raspberry heruntergefahren wird, verbraucht das System Energie weiter. Folgt die Batterie wurde innerhalb ein paar Stunden komplett leer sein. Um dieses Problem umzugehen muss die Speisung ausgeschaltet werden. Das kann mit der Ansteuerung der «Enable» Pin der PowerBoost 1000c Board realisiert werden.

### Batterie-Spannungsmessung

Die Spannung kann einfach mit einem Spannungsteiler umgewandelt werden und mit einem ADC ausgewertet werden. Da der Raspberry Pi nicht mit einem ADC ausgestattet ist, muss ein Externes evaluiert werden. Für die Kommunikation zwischen ADC und uC soll das Protokoll I2C angewendet. Der MCP3426 vom Hersteller Microchip wurde ausgewählt.

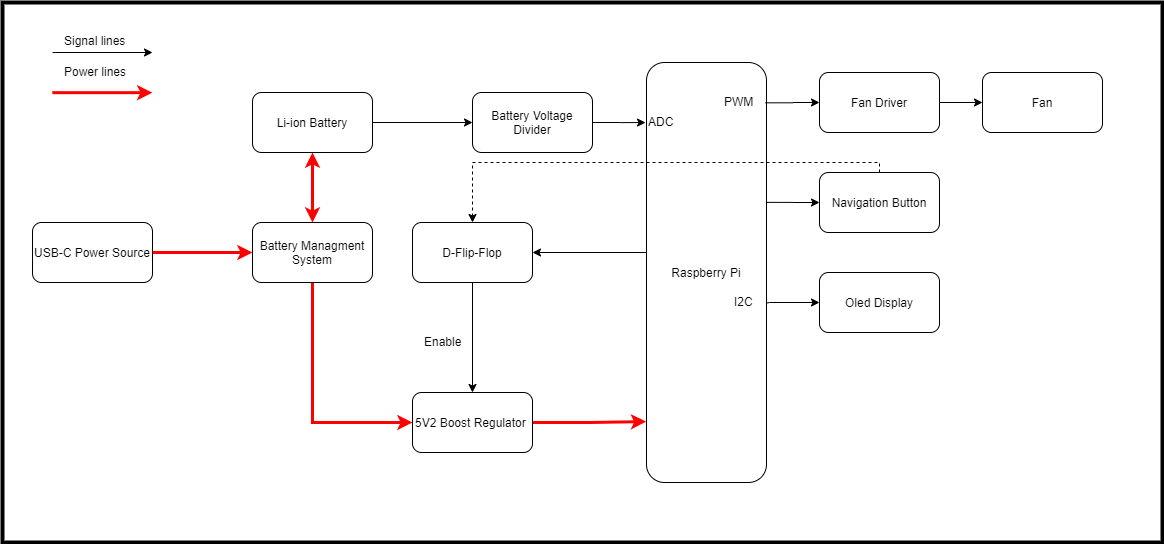
Der gewählte AD-Wandler besitzt folgende Eigenschaften:

* 16bit Auflösung
* 2 Differenzeingänge
* Delta-sigma verfahren
* I2C Schnittstelle

## Realisierung

In diesem Abschnitt wird auf die Realisierung der Hardware genauer eingegangen und die einzelnen Schaltungsblöcke werden genauer erklärt.

### Blockschaltbild



### Mikrokontroller

Der Mikrocontroller ist der Kern der Hardware und steuert alle anderen Baugruppen. Integriert auf dem Board ist auch die 3.3V Speisung.

### Speisung

Das Gerät ist über USB gespiesen. Folgt die Speisung ist schon vorhanden. Was noch implementiert worden ist, ist eine Schutz Schaltung, um das Gerät gegen Überspannungen und Überlastungen zu schützen. Die Schaltung wurde von einem vorherigen Projekt übernommen. Der TVS Diode D1 Schutz gegen Überspannungen. Der Widerstand R1 ist ein PTC Rücksetzbare Sicherung, die bei einem Strom von 2.5A oder höher ihr interne Widerstand erhöht und konsequent die Schaltung trennt.

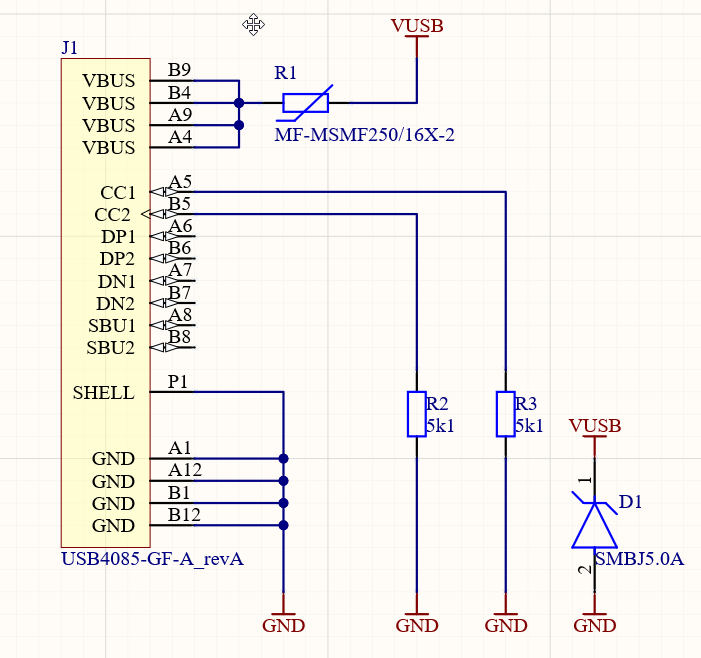


Abbildung 1: USB Type-C und Schutzschaltung

Wie im Dokument «Introduction to USB Type-C™» von Microchip empfohlen wird wurden zwei 5k1 Widerstände zwischen Ground und Pin CC1 und CC2 von der USB Buchse bestückt. Diesen stellen sicher, dass das Gerät mit genügend Leistung von der USB-Speisung versorgt wird.

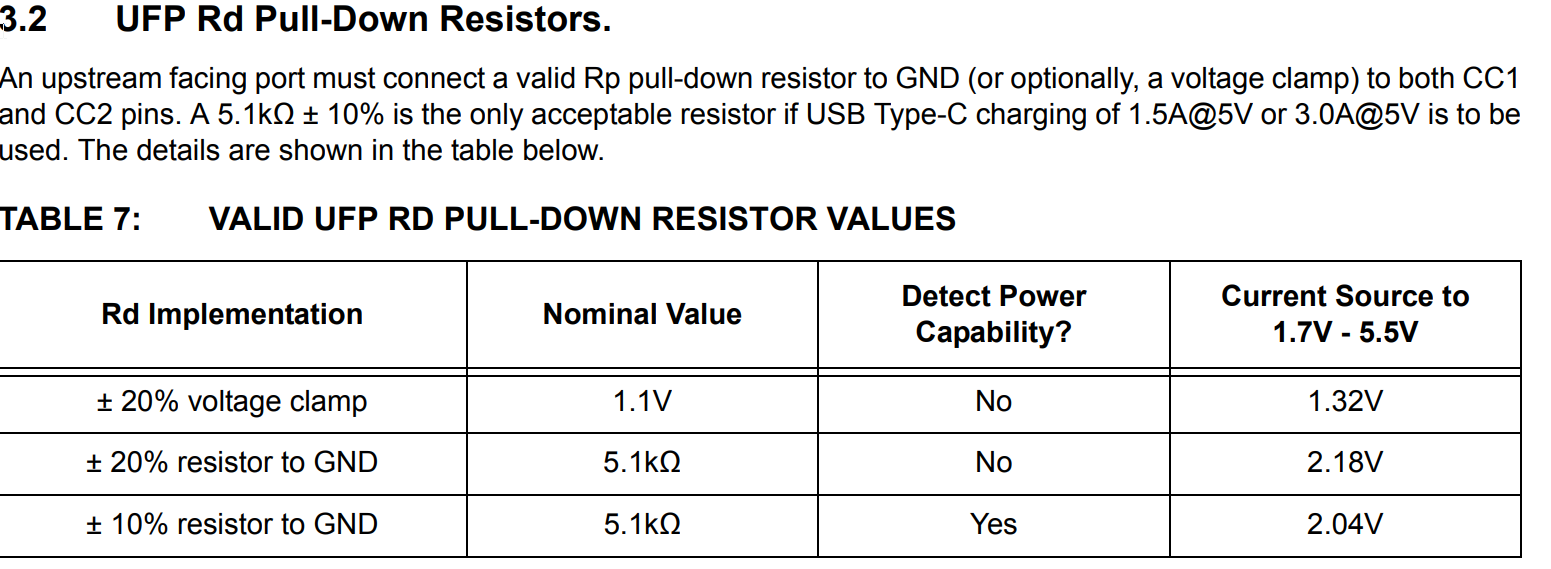


Abbildung 2: Ausschnitt aus dem PDF «Introduction to USB Type-C™»

Teil der Speisung ist auch das externe Board PowerBoost 1000c. Die externe Speisung, die Batterie und die 5V Ausgang müssen an diese Schaltung angeschlossen werden. Zusätzlich sind eine Leitung für den Enable Pin und eine Leitung für der Batterie Zustand Melder mit dem Board verbunden.

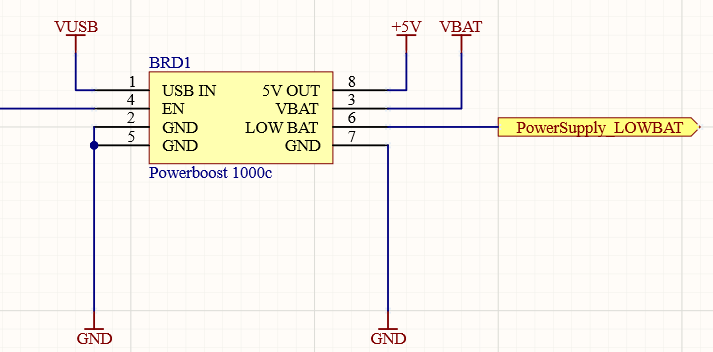


Abbildung 3: PowerBost 1000c

### Batterie Spannung Messung

Die Messung der Spannung erfolgt über dem Spannungsteiler (R10-R12-R14). Da die drei Widerstände genau der gleiche Werte haben (±1%), ist die Ausgangspannung gleich die Eingangsspannung geteilt durch drei. Diese Pegel ist dann mit dem ADC ausgewertet. Der ADC arbeitet mit eine Interne Spannungsreferenz von 2,048V. Die maximale Spannung an den Differenzeingänge darf diesen Wert nicht überschritten. C1 und C2 schützen der ADC gegen Störungen.

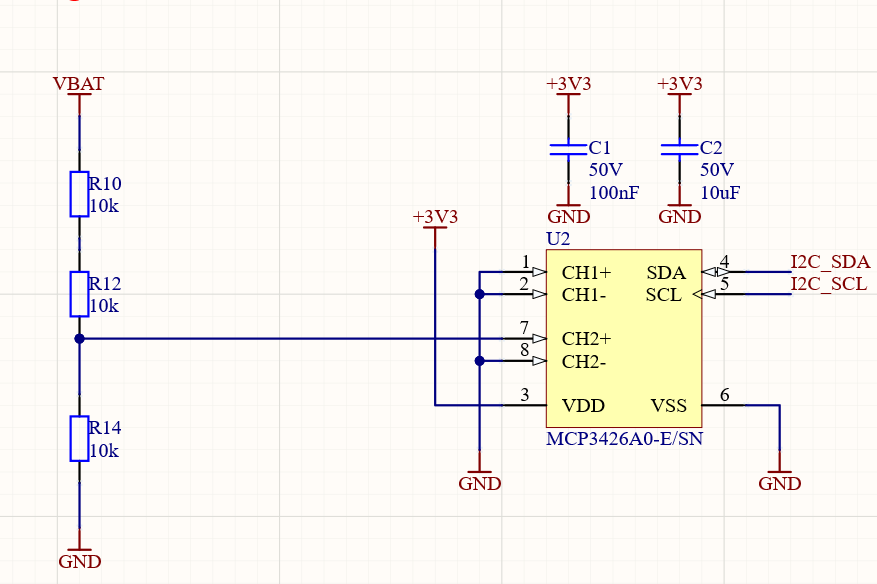


Abbildung 4: Spannungsmessung

### Soft Power ON/OFF Schaltung

Der Schaltregler auf dem PowerBoost Board kann mit der Ansteuerung der Enable Pin ein- und ausgeschaltet werden. So kann die Speisung der Raspberry Pi komplett trennen.

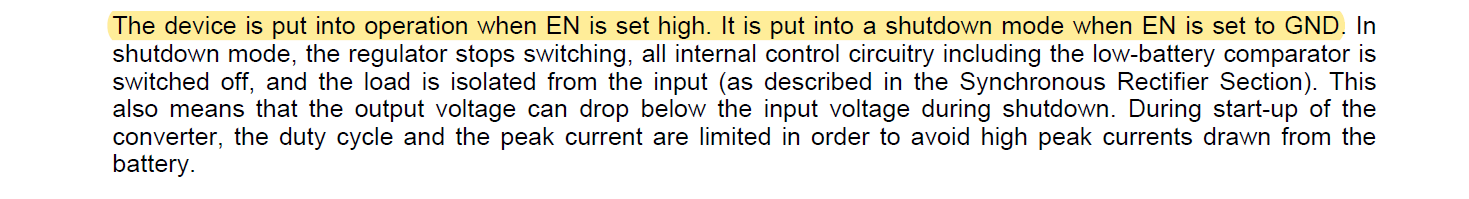


Abbildung 5: : Ausschnitt aus dem Datenblatt vom TPS61090

Um der Enable Pin immer in einen definierten Zustand ohne externe Handlung zu behalten, wird er mit ein RS-Flip-Flop angesteuert. Diese Flip-Flop Typ ändert sein Ausgangszustand nur wenn eine Flanke auf dem Set oder Reset Pin detektiert wird. Der Ausgewählte Flip-Flop ist der NC7SZ74K8X der Hersteller ON Semiconductor. Dieses Model kann entweder als D- oder RS-Flip-Flop verwendet werden. Für diese Anwendung wird ein RS Typ benötigt. Anhand der unterstehende Wahrheitstabelle kann man sagen, dass wenn eine negative Flank auf dem Clear Pin ist, wird die Speisung ausgeschaltet und wenn eine negative Flanke auf dem Preset Pin ist, wird die Speisung eingeschaltet.

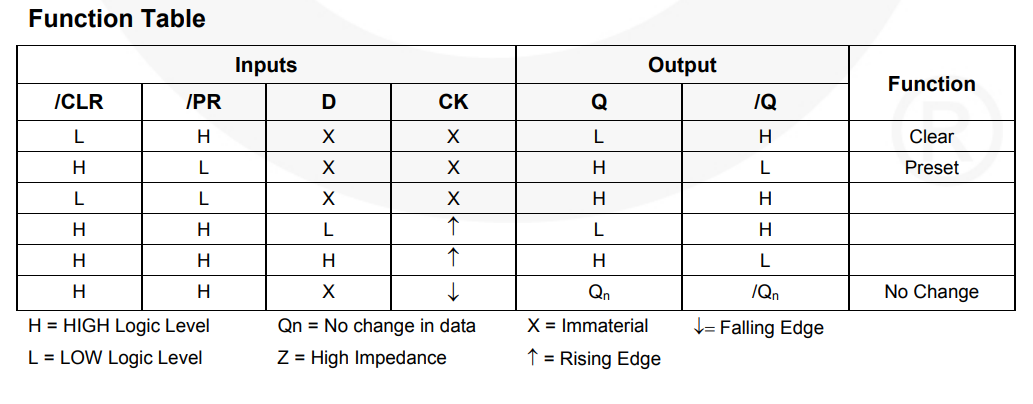


Abbildung 6: Ausschnitt aus dem Datenblatt vom NC7SZ74K8X

Da die Logik Pegel der Raspberry niedriger als die Spannung der Batterie sind, ist ein Level Shifter benötigt. Dieses wurde mit zwei MOSFET (Q1 und Q2) realisiert.

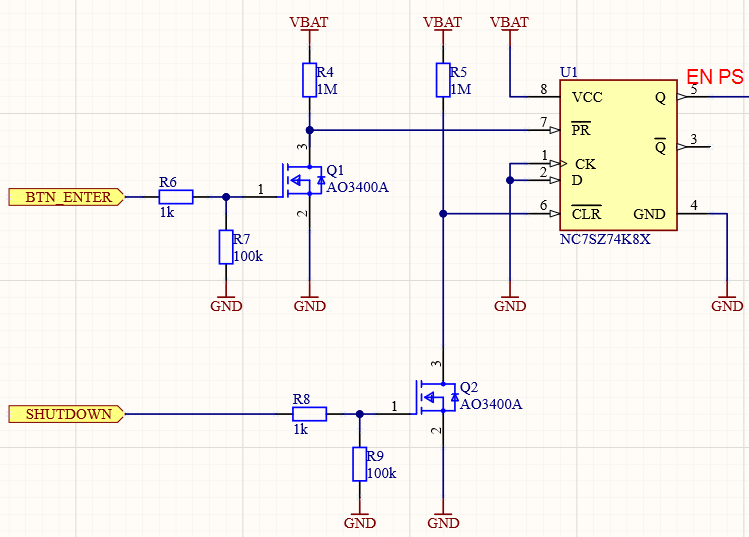


Abbildung 7: Soft ON/OFF Schaltung

### Lüfter Steuerung

Der Lüfter wird mit einem PWM Signal angesteuert. So kann die Geschwindigkeit der Lüfter fein eingestellt werden. Da ein GPIO vom Raspberry nicht die benötigte Leistung treiben kann, wird die Leistung mit ein MOSFET angesteuert. Der Motor des Lüfters ist ein Induktives Last. Folgt, wenn die Schaltung geöffnet wird und keinen elektrischen Strom mehr fliesst, das Magnetische Feld in den Wicklungen des Motors bricht zusammen und eine elektrische Spannung wird induziert. Diese Spannung kann bis zu einige kV (Kilovolts) gross werden. Die MOSFET sind allgemein auf grosse Spannungen sehr empfindlich. Die in diesem Projekt angewendete AO3400A hat eine BVDSS (Drain-Source Breakdown Voltage) von nur 30V. Ohne ein Überspannungsschutz würde der MOSFET nach einigen Schaltvorgängen kaputt gehen. Um dieses Problem zu umgehen wurde die Freilaufdiode D2 eingebaut. Damit wird sichergestellt, dass die VDS Spannung (Drain-Source Voltage) von Q3 nie grösser als 5.7V wird.

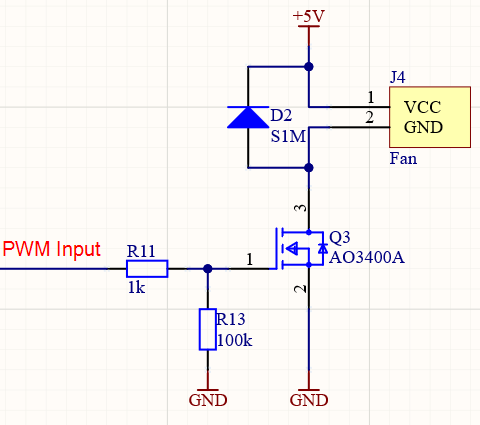


Abbildung 8: Lüfter Steuerung Schaltung

### Bedienung Komponenten

Die Bedienung des Geräts erfolgt über ein OLED Display und ein 5-Faches Navigationsknopf. Das Display wird über I2C angesteuert und mit 3.3V gespiesen. Die Pull-up Widerstände für die I2C sind auf dem Raspberry schon vorhanden. Der Navigationsbutton ist direkt mit 5 GPIOs der Raspberry Pi verbunden. Pull-up Widerstände werden dann Softwaremässig eingeschaltet. Die Entprellung wird auch Softwaremässig implementiert.

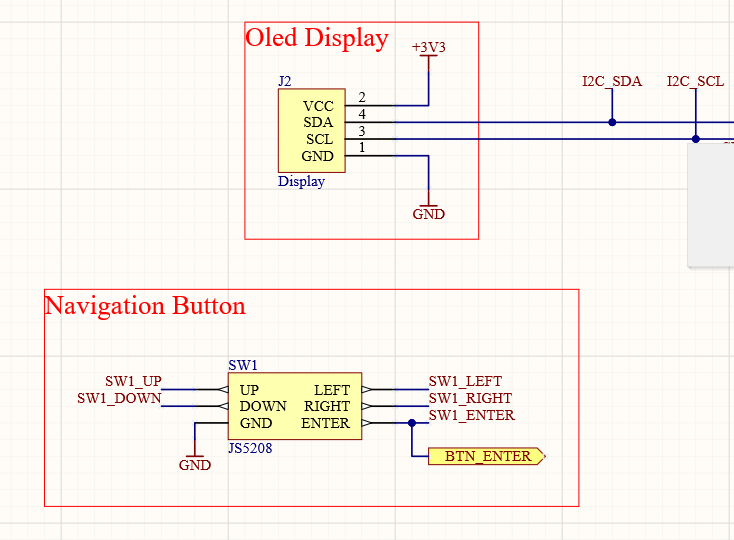


Abbildung 9: Bedienung Komponenten Schaltung

# Software

In diesem Abschnitt der Dokumentation sollen die einzelnen Teile der Software analysiert und erklärt werden. Die Software ist in ein Hauptprogramm und verschiedenen Kleinere unabhängigen Skripts aufgebaut.

## Hauptprogramm

Die Software für die Steuerung der Hardware wurde in Form von Python Scripts geschrieben. Das Hauptprogramm ist zuständig für die Interaktion mit dem Benutzer.

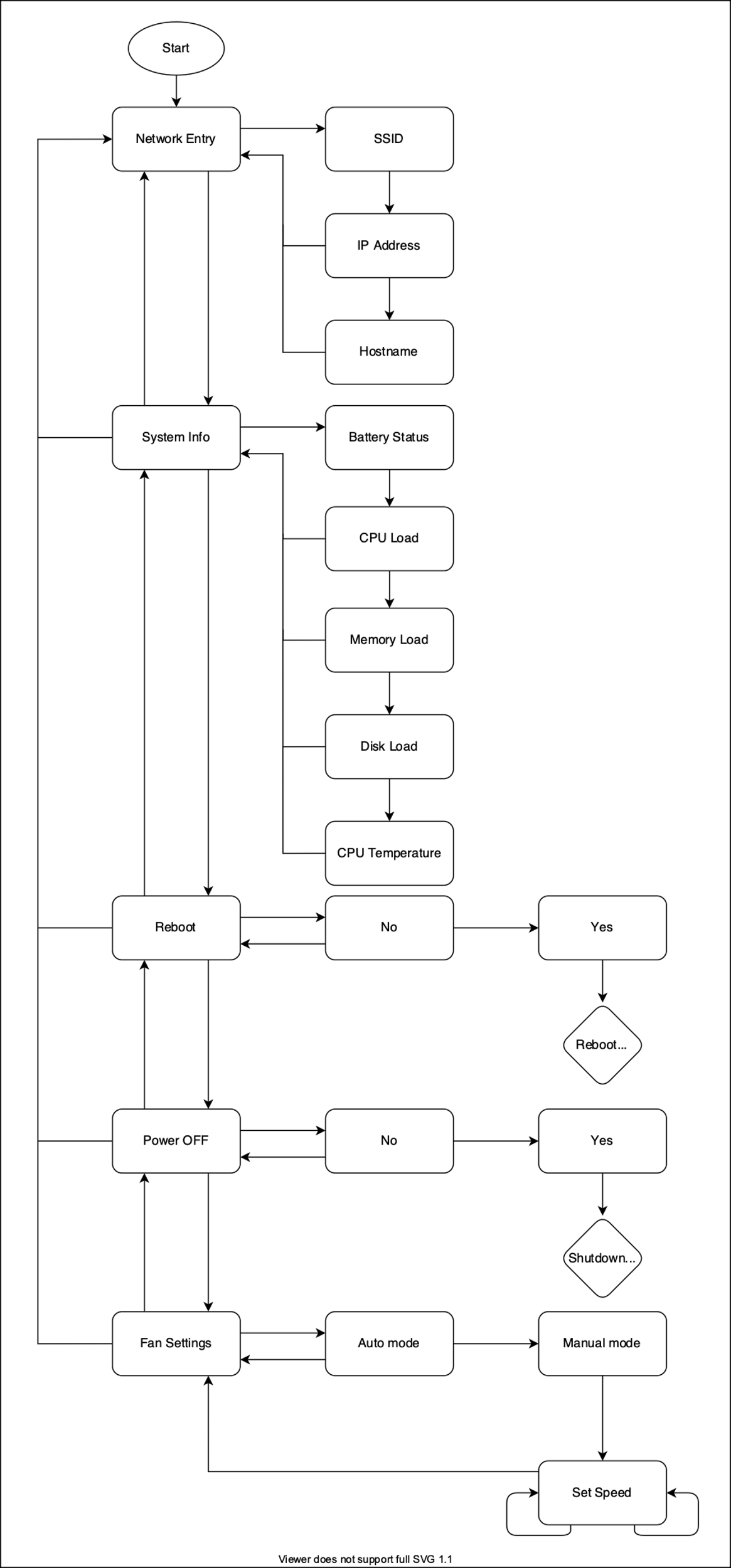
### Aufbau

Die Software wurde modular strukturiert. Das bedeutet, dass jede Peripherie in ein eigenes Modul aufgeteilt ist und alle Funktionen, welche für die Ansteuerung der jeweiligen Peripherie benötigt werden, sind darin enthalten. Diese Files können im Hauptprogramm als Python Module eingebunden werden. Lediglich alle Variablen und Parameter wurden in den entsprechenden Module zusammengefasst.

### Ablauf Flussdiagramm

Der Softwareablauf entspricht grundsätzlich dem Modell eines Flussdiagramms. Dies liegt daran, dass solange keine Taster gedrückt werden, das Programm immer denselben Ablauf durchführt. Erst sobald eine Callback von der GPIOs ausgelöst wird, wird gehandelt. Aus diesem Grund wird die Software mit Hilfe des Flussdiagramms in Abbildung 10 dargestellt.

Wenn das Programm gestartet wird, wird automatisch die Erste Eintrag des Hauptmenus auf dem Display dargestellt. Mit den Tasten Hoch und Runter kann man zwischen den verschiedenen Einträgen navigieren. Um in ein Untermenü zu gehen muss der Knopf in der Mitte gedrückt werden. Analog zu Hauptmenü können die Untermenüs mit den Tester Hoch und Unten gesteuert werden. Um zurück in dem Hauptmenü zu gelangen muss der Taster in der Mitte gedrückt werden.



### Konfiguration

Damit der Python Script fehlerfrei funktionieren kann müssen die Benötigte Python Libraries installiert werden. Zusätzlich muss die Hardware einmalig konfiguriert werden. Um diese Aufgaben zu erledigen wurde ein Shell Skript geschrieben. Beim Ausführen wird zuerst das Betriebssystem auf dem neusten Stand aktualisiert. Danach werden die Benötigte Programme, falls nicht vorhanden, installiert. Gleichzeitig wird Python3 als Standard Python Interpreter eingestellt. Danach werden die Python Libraries installiert, um die GPIOs und das Display ansteuern zu können. Schliesslich wird die I2C Bus aktiviert und die I2C Clock von der Standard 100kHz auf 1Mhz eingestellt.

### System Befehle

Um an Systeminformationen zuzugreifen wurden Bash Befehle benutzt. Bash ist die Standard Linux Shell. Mithilfe einer Funktion der «subprocess» Standardmodul können innerhalb ein Python Script Bash Befehle ausgeführt werden. Das Resultat wird dann in Variablen gespeichert.

#### SSID

iwgetid -r

Mit diesem Befehl kann der Name der Wi-Fi Netzwerk, mit dem das Gerät momentan verbunden ist, ausgeben.

#### IP-Adresse

hostname -I | cut -d' ' -f1

Mit diesem Befehl kann die IP-Adresse ausgegeben werden. Mithilfe des «cut» Befehls wird die Ausgabe formatiert.

#### CPU-Auslastung

top -bn1 | grep load | awk '{printf \"%.2f%%\", $(NF-2)}'

Mit diesem Befehl wird die CPU-Auslastung in Prozent ausgegeben. Mit dem Befehl «awk» wird den Wert mit zwei nachkommastellen formatiert und danach der Prozent Zeichen «%» angehängt.

#### CPU-Temperatur

cat /sys/class/thermal/thermal\_zone0/temp | awk '{printf \"%.2f\", $1/1000}'

Mit diesem Befehl wird die CPU-Temperatur in Milli Grad Celsius ausgegeben. Mithilfe des «awk» Befehls wird den Wert in Grad Celsius umgerechnet und mit zwei nachkommastellen ausgegeben.

#### RAM-Auslastung

free -m | awk 'NR==2{printf \"%.2f%%\", $3\*100/$2 }'

Mit diesem Befehlt werden die frei verfügbare Programspeicher und die gesamten physisch installierten Speicher ausgegeben. Beide Werten in Megabytes. Mit dem «awk» Befehl wird die frei verfügbarer RAM in Prozent ausgerechnet und mit zwei relevante nachkommastellen ausgegeben.

#### Flash Speicher Auslastung

df -h | awk \'$NF=="/"{printf "%d/%dGB %s", $3,$2,$5}\'

Mit diesem Befehl werden die gesamten physisch installierten Speicher in Gigabytes und die frei Verfügbare Speicher in Gigabytes und in Prozent ausgegeben. Mit dem Befehl «awk» werden die Werten formatiert ausgegeben.

## Auto Startup

Das Hauptprogramm muss nach jedem Neustart automatisch gestartet werden. Zudem muss das Skript in Hintergrund laufen. Es hat verschieden Möglichkeiten, dieser Vorgang zu automatisieren. Es wurde entschieden der Tool Cron einzusetzen.

### Crontab

Aus dem Wikipedia Artikel über Cron:

«Der Cron-Daemon dient der zeitbasierten Ausführung von Prozessen in Unix und unixartigen Betriebssystemen wie Linux, BSD oder macOS, um wiederkehrende Aufgaben – Cronjobs – zu automatisieren.»

Grundsätzlich können Aufgaben in ein Sogenannten Crontab File hinzugefügt werden, dazu muss spezifiziert, wann die Aufgaben ausgeführt werden müssen (Zeit oder Ereignisse wie z.B. Neustart). Jeder Benutzer hat ein Cron File.

Um dieser File zu öffnen muss folgende Kommando ausgeführt werden:

crontab -e

In diesem Fall sieht die Aufgabe Eintrag so aus:

@reboot /home/dev/documents/berry\_case/firmware/python/autostart.sh

In andere Wörter, nach jedem Neustart führt das Skript «autostart.sh» aus.

### Autostart Skript

Das Skript sieht so aus:

# Project: berry\_case

# File: autostart.sh

# Autor: Enrico Cirignaco

# Created: 11.12.2020

# Description: This script need to be added to the crontab of the pi user.

# crontab -e --> @reboot /home/dev/documents/berry\_case/firmware/python/autostart.sh

# This script just run the main Python script wit sudo rights in background.

#!/bin/bash

cd /home/dev/documents/berry\_case/firmware/python/

sudo -E python3 ./main.py &

Bevor das Hauptprogramm ausgeführt werden kann, muss sichergestellt werden, dass man sich in das richtige Directory befindet. Das geschieht mit dem «cd» Kommando.

Danach wird der der Python Skript dem Superuser Benutzer im Hintergrund ausgeführt.