

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

05.02.21

Abstract

Die im nachfolgenden Dokument beschriebene Arbeit wurde im Rahmen einer Berufsübergreifenden Projektarbeit (BüP) durchgeführt:

Die Aufgabe lautete, ein Gerät zu entwickeln das 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, so dass 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 den 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 Englischen Unterbrechungsfreie Stromversorgung). Dies ermöglicht 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 begonnen werden, am optionalen Ziel zu arbeiten. Dieses wurde aber nicht abgeschlossen. Verbesserungen, die noch implementiert werden könnten, wären das 3D Gehäuse fertig zu entwickeln und auszudrucken. Zudem könnte ein grösserer Akku eingebaut werden. Das Kühlungssystem könnte auch so erweitert werden, dass die CPU/GPU übertaktet werden kann.

Inhaltsverzeichnis

1	Einleitung	7
1.1	Aufgabenstellung	7
1.2	Vorgehen.....	7
1.3	Zielsetzung.....	8
1.3.1	Grundfunktionen	8
1.3.2	Optional.....	8
2	Hardware	9
2.1	Konzept.....	9
2.1.1	Mikrokontroller	9
2.1.2	OLED Display	9
2.1.3	UPS & BMS	9
2.1.4	Soft-Power OFF	10
2.1.5	Batterie-Spannungsmessung.....	10
2.2	Realisierung	10
2.2.1	Blockschaltbild	10
2.2.2	Mikrokontroller	11
2.2.3	Speisung	11
2.2.4	Batterie Spannung Messung.....	12
2.2.5	Soft Power ON/OFF Schaltung.....	13
2.2.6	Lüfter Steuerung	14
2.2.7	Bedienung Komponenten	14
3	Software.....	15
3.1	Hauptprogramm	15
3.1.1	Aufbau.....	15
3.1.2	Ablauf Flussdiagramm	15
3.1.3	Konfiguration.....	17
3.1.4	System Befehle.....	17
3.2	Auto Startup	18
3.2.1	Crontab	18
3.2.2	Autostart Skript	19
3.3	Netzwerk	19
3.3.1	Netzwerktopologie	20

4	Schlussbetrachtung	21
4.1	Fazit	21
4.2	Verbesserungsmöglichkeiten	21
4.2.1	Hardware	21
4.2.2	Software	21
4.3	Ausblicke	21
5	Glossar	22
6	Literatur- und Quellenverzeichnis	23
7	Anhang	24
7.1	Zeitplan	24
7.2	Aufgabestellung	25

Abbildungsverzeichnis

Abbildung 1: OLED Display	9
Abbildung 2: PowerBoost 1000C.....	9
Abbildung 3: Blockschaltbild.....	10
Abbildung 4: USB Type-C und Schutzschaltung	11
Abbildung 5: Ausschnitt aus dem PDF «Introduction to USB Type-C™»	11
Abbildung 6: PowerBost 1000c	12
Abbildung 7: Spannungsmessung.....	12
Abbildung 8: : Ausschnitt aus dem Datenblatt vom TPS61090	13
Abbildung 9: Ausschnitt aus dem Datenblatt vom NC7SZ74K8X	13
Abbildung 10: Soft ON/OFF Schaltung.....	13
Abbildung 11: Lüfter Steuerung Schaltung.....	14
Abbildung 12: Bedienung Komponenten Schaltung	14
Abbildung 13: Flussdiagramm Hauptprogramm	16
Abbildung 14: Netzwerktopologie Access Point	20
Abbildung 15: Netzwerktopologie eigene WiFi Network.....	20

Abkürzungsverzeichnis

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
BV_{DSS}	Drain-Source Breakdown Voltage
V_{DS}	Drain-Source Voltage
SoC	System on a chip
PIR	Passive Infrared Sensor
TVS	Transient-voltage-suppression diode
PTC	Positive Temperature Coefficient
ADC	Analog-Digital-Converter
I/O	Inputs/Outputs
LED	Light Emitting Diode
uC	Mikrocontroller
GND	Ground
V	Volt
A	Ampere
IC	Integrated Circuit

1 Einleitung

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

1.1 Aufgabenstellung

Den Raspberry Pi unterwegs zu benutzen ist sehr umständlich. Eine Powerbank ist nötig und ein Netzwerk muss aufgesetzt werden. Um die UX (Nutzererfahrung) zu steigern, muss ein Gehäuse für den 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.

1.2 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.

1.3 Zielsetzung

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

1.3.1 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

1.3.2 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.

2 Hardware

2.1 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.

2.1.1 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.

2.1.1.1 Vorgaben

Der Kontroller muss folgende Punkte erfüllen:

- Auf dem Kontroller muss ein 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.

2.1.2 OLED Display

Um nutzbare Informationen anzuzeigen wurde entschieden, ein OLED Display einzusetzen. Die Hauptvorteile diese Technologie sind, dass keine Hintergrundbeleuchtung benötigt wird und seine hervorragender Kontrasteigenschaft. Dazu sind diese Displays preisgünstig und stromsparender. Das ausgewählte OLED Display hat eine Bilddiagonale von 0.91 Zoll und eine Auflösung von 128x32 Pixels. Die Anzeige kann mit 3.3v und 5v gespeist werden. Angesteuert wird sie über das I2C Protokoll.



Abbildung 1: OLED Display

2.1.3 UPS & BMS

Auf dem Markt gibt es schon eine Lösung, die genau die in diesem Projekt benötigten Funktionalitäten aufweist. Der PowerBoost 1000c des Herstellers Adafruit. Die Entwicklungsunterlagen dieses Produkts sind frei zugänglich und Open-Source. Es wurde zuerst evaluiert ob es sinnvoll wäre, die Schaltung auf eine eigene Leiterplatte zu integrieren oder das Produkt von Adafruit zu übernehmen. Zwei kritische Bauteile der Schaltung

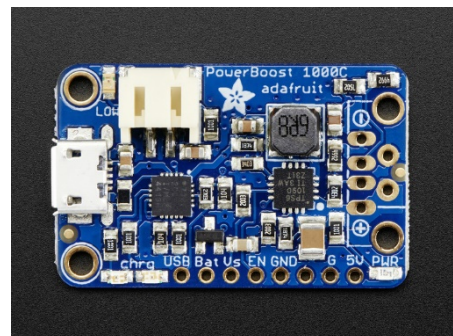


Abbildung 2: PowerBoost 1000C

können nur mit speziellen Werkzeugen bestückt werden. Dazu ist der Preisunterschied zwischen den zwei Varianten nicht sehr gross. Es wurde darum entschieden die fertige Leiterplatten von Adafruit einzukaufen und im Projekt zu integrieren.

2.1.4 Soft-Power OFF

Der Raspberry muss mithilfe der Benutzeroberfläche ausgeschaltet werden können. Nachdem der Raspberry heruntergefahren wird, verbraucht das System weiter Energie. Als Folge wäre die Batterie innerhalb ein paar Stunden komplett leer. Um dieses Problem zu umgehen, muss die Speisung ausgeschaltet werden. Das kann mit der Ansteuerung der «Enable» Pin der PowerBoost 1000c Board realisiert werden.

2.1.5 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 werden. Der MCP3426 vom Hersteller Microchip wurde ausgewählt.

Der gewählte AD-Wandler besitzt folgende Eigenschaften:

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

2.2 Realisierung

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

2.2.1 Blockschaltbild

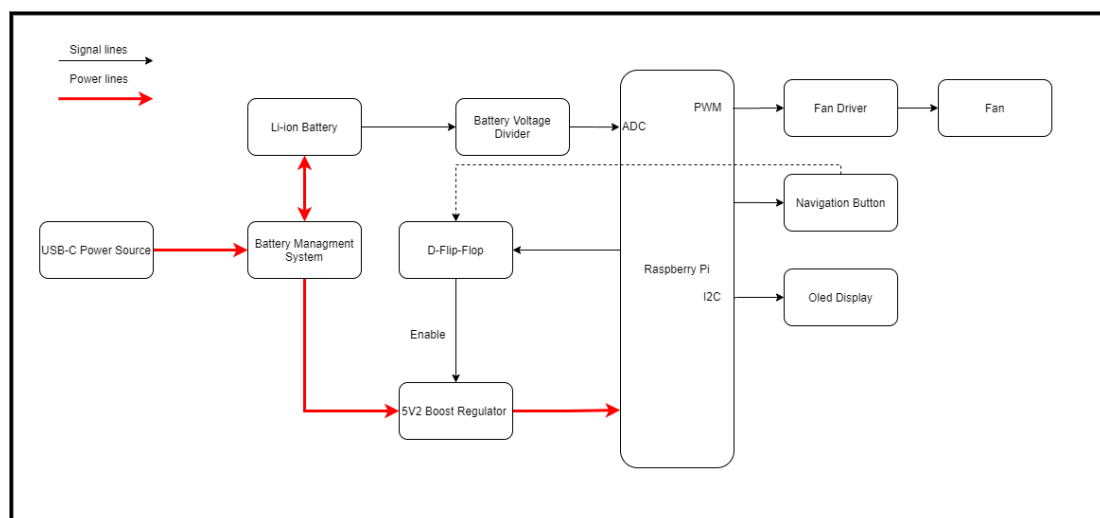


Abbildung 3: Blockschaltbild

2.2.2 Mikrocontroller

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

2.2.3 Speisung

Das Gerät wird über USB gespeist, somit ist die Speisung schon vorhanden. Was noch implementiert worden ist, ist eine Schutzschaltung, um das Gerät gegen Überspannungen und Überlastungen zu schützen. Die Schaltung wurde von einem vorherigen Projekt übernommen. Die TVS Diode D1 dient als Schutz gegen Überspannungen. Der Widerstand R1 ist eine PTC rücksetzbare Sicherung, die bei einem Strom von 2.5A oder höher ihr interner Widerstand erhöht und konsequent die Schaltung trennt.

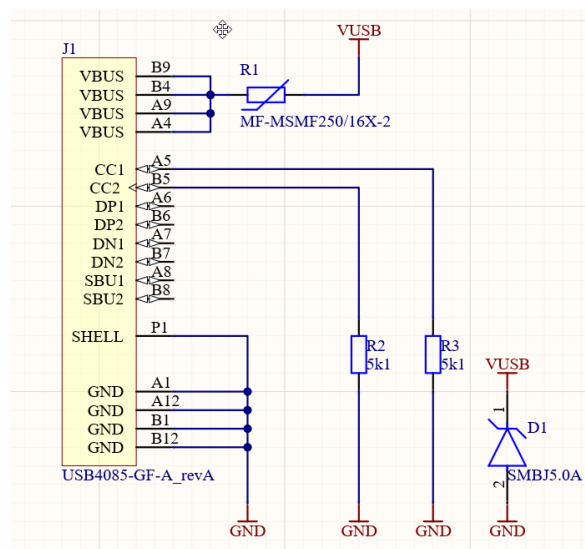


Abbildung 4: USB Type-C und Schutzschaltung

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. Diese stellen sicher, dass das Gerät mit genügend Leistung von der USB-Speisung versorgt wird.

3.2 UFP Rd Pull-Down Resistors.

An upstream facing port must connect a valid R_p pull-down resistor to GND (or optionally, a voltage clamp) to both CC1 and CC2 pins. A $5.1k\Omega \pm 10\%$ is the only acceptable resistor if USB Type-C charging of $1.5A@5V$ or $3.0A@5V$ is to be used. The details are shown in the table below.

TABLE 7: VALID UFP RD PULL-DOWN RESISTOR VALUES

Rd Implementation	Nominal Value	Detect Power Capability?	Current Source to 1.7V - 5.5V
$\pm 20\%$ voltage clamp	1.1V	No	1.32V
$\pm 20\%$ resistor to GND	5.1k Ω	No	2.18V
$\pm 10\%$ resistor to GND	5.1k Ω	Yes	2.04V

Abbildung 5: 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 der 5V Ausgang müssen an diese Schaltung angeschlossen werden. Zusätzlich sind eine Leitung für den Enable Pin und eine Leitung für den Batteriezustandsmelder mit dem Board verbunden.

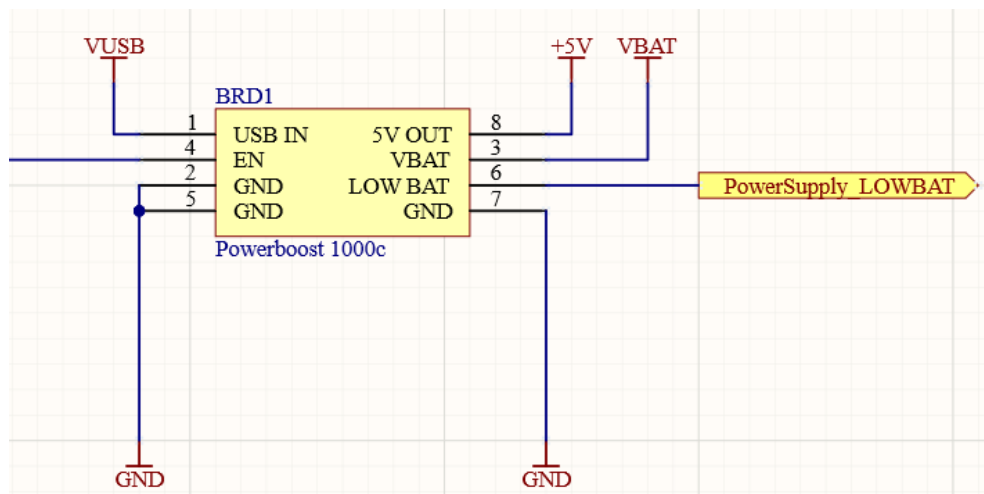


Abbildung 6: PowerBoost 1000c

2.2.4 Batterie Spannung Messung

Die Messung der Spannung erfolgt über den Spannungsteiler (R10-R12-R14). Da die drei Widerstände genau den gleichen Wert haben ($\pm 1\%$), entspricht die Ausgangsspannung einem Drittel der Eingangsspannung. Dieser Pegel wird dann mit dem ADC ausgewertet. Der ADC arbeitet mit einer internen Spannungsreferenz von 2,048V. Die maximale Spannung an den Differenzeingängen darf diesen Wert nicht überschreiten. C1 und C2 schützen den ADC gegen Störungen.

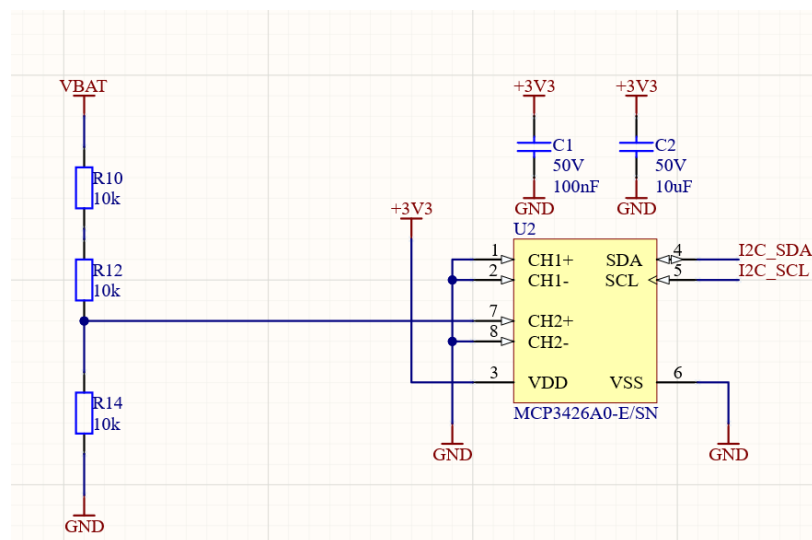


Abbildung 7: Spannungsmessung

2.2.5 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 den Raspberry Pi komplett trennen.

The device is put into operation when EN is set high. It is put into a shutdown mode when EN is set to GND. In shutdown mode, the regulator stops switching, all internal control circuitry including the low-battery comparator is switched off, and the load is isolated from the input (as described in the Synchronous Rectifier Section). This also means that the output voltage can drop below the input voltage during shutdown. During start-up of the converter, the duty cycle and the peak current are limited in order to avoid high peak currents drawn from the battery.

Abbildung 8: : Ausschnitt aus dem Datenblatt vom TPS61090

Um den Enable Pin immer in einem definierten Zustand ohne externe Handlung zu behalten, wird er mit einem RS-Flip-Flop angesteuert. Dieser Flip-Flop Typ ändert seinen Ausgangszustand nur, wenn eine Flanke auf dem Set oder Reset Pin detektiert wird. Das ausgewählte Flip-Flop ist der NC7SZ74K8X des Herstellers ON Semiconductor. Dieses Model kann entweder als D- oder RS-Flip-Flop verwendet werden. Für diese Anwendung wird ein RS Typ benötigt. Anhand der unterstehenden Wahrheitstabelle kann man sagen, dass wenn eine negative Flanke auf dem Clear Pin ist, wird die Speisung ausgeschaltet und wenn eine negative Flanke auf dem Preset Pin ist, wird die Speisung eingeschaltet.

Function Table

Inputs				Output		Function
/CLR	/PR	D	CK	Q	/Q	
L	H	X	X	L	H	Clear
H	L	X	X	H	L	Preset
L	L	X	X	H	H	
H	H	L	↑	L	H	
H	H	H	↑	H	L	
H	H	X	↓	Q _n	/Q _n	No Change

H = HIGH Logic Level

Q_n = No change in data

X = Immaterial

↓ = Falling Edge

L = LOW Logic Level

Z = High Impedance

↑ = Rising Edge

Abbildung 9: Ausschnitt aus dem Datenblatt vom NC7SZ74K8X

Da der Logikpegel des Raspberry niedriger als die Spannung der Batterie ist, wird ein Pegelwandler benötigt. Dieser wurde mit zwei MOSFET (Q1 und Q2) realisiert.

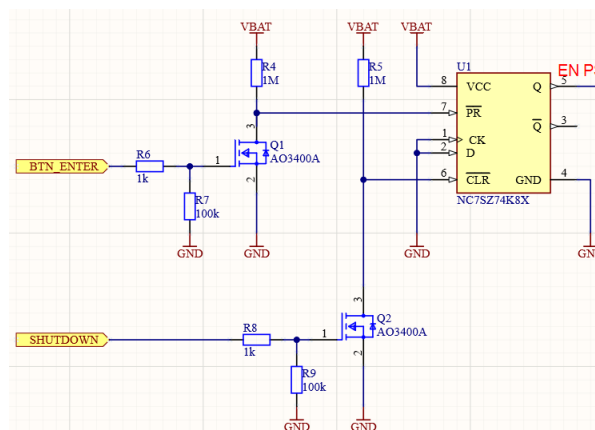


Abbildung 10: Soft ON/OFF Schaltung

2.2.6 Lüfter Steuerung

Der Lüfter wird mit einem PWM Signal angesteuert. So kann die Geschwindigkeit des Lüfters fein eingestellt werden. Da ein GPIO vom Raspberry nicht die benötigte Leistung erbringen kann, wird die Leistung mit einem MOSFET angesteuert. Der Motor des Lüfters ist eine induktive Last. Folglich, wenn die Schaltung geöffnet wird und kein elektrischer Strom mehr fließt, bricht das magnetische Feld in den Wicklungen des Motors zusammen und eine elektrische Spannung wird induziert. Diese Spannung kann bis zu einigen kV (Kilovolts) gross werden. Die MOSFET sind allgemein auf grosse Spannungen sehr empfindlich. Die in diesem Projekt angewendete AO3400A hat eine BV_{DSS} (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 V_{DS} Spannung (Drain-Source Voltage) von Q3 nie grösser als 5.7V wird.

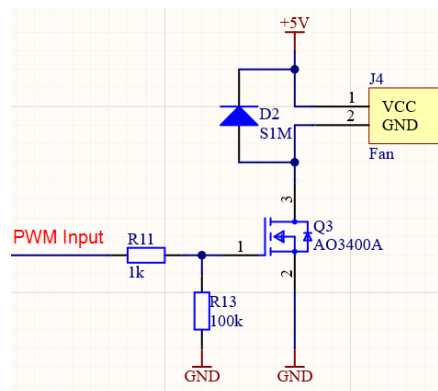


Abbildung 11: Lüfter Steuerung Schaltung

2.2.7 Bedienung Komponenten

Die Bedienung des Geräts erfolgt über ein OLED Display und einen 5-fachen Navigationsknopf. Das Display wird über I2C angesteuert und mit 3.3V gespeist. 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.

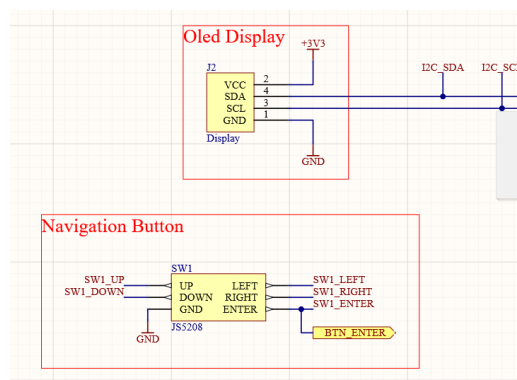


Abbildung 12: Bedienung Komponenten Schaltung

3 Software

In diesem Abschnitt der Dokumentation sollen die einzelnen Teile der Software analysiert und erklärt werden. Die Software besteht aus einem Hauptprogramm und verschiedenen kleineren unabhängigen Skripts.

3.1 Hauptprogramm

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

3.1.1 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, darin enthalten sind. Diese Files können im Hauptprogramm als Python Module eingebunden werden. Lediglich alle Variablen und Parameter wurden in den entsprechenden Modulen zusammengefasst.

3.1.2 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 der erste Eintrag des Hauptmenüs 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 Tasten Hoch und Runter gesteuert werden. Um zurück in das Hauptmenü zu gelangen muss die Taste in der Mitte gedrückt werden.

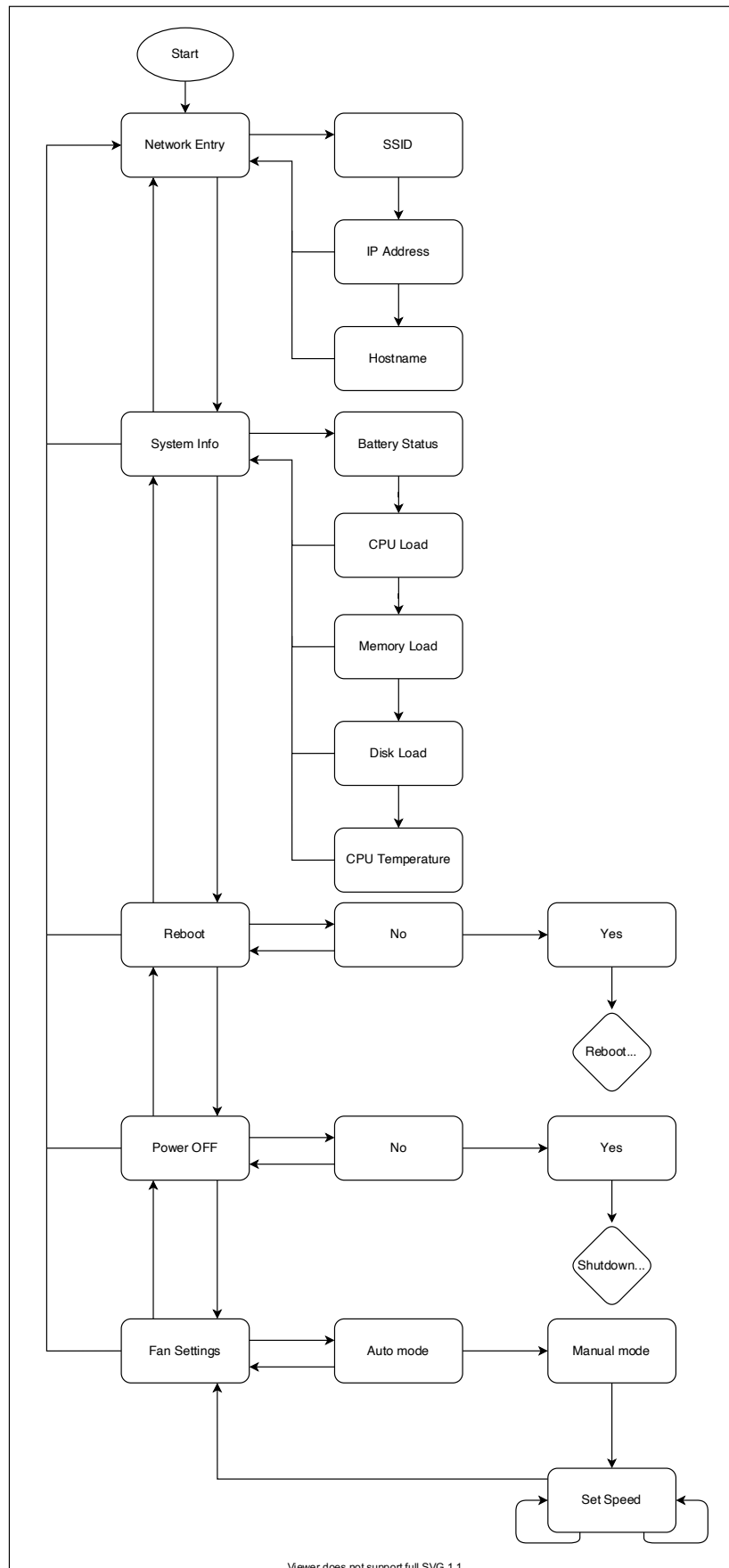


Abbildung 13: Flussdiagramm Hauptprogramm

3.1.3 Konfiguration

Damit der Python Script fehlerfrei funktionieren kann, müssen die benötigten 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 den neusten Stand aktualisiert. Danach werden die benötigten 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 der I2C Bus aktiviert und die I2C Clock vom Standardwert von 100kHz auf 1Mhz eingestellt.

3.1.4 System Befehle

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

3.1.4.1 SSID

```
iwgetid -r
```

Mit diesem Befehl kann der Name des Wi-Fi Netzwerks, mit dem das Gerät momentan verbunden ist, ausgegeben.

3.1.4.2 IP-Adresse

```
hostname -I | cut -d' ' -f1
```

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

3.1.4.3 CPU-Auslastung

```
top -bn1 | grep load | awk '{printf "%.2f%\n", $(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 das Prozentzeichen «%» angehängt.

3.1.4.4 CPU-Temperatur

```
cat /sys/class/thermal/thermal_zone0/temp | awk '{printf "%.2f\n", $1/1000}'
```

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

3.1.4.5 RAM-Auslastung

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

Mit diesem Befehl werden die frei verfügbare Programmspeicher 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 relevanten Nachkommastellen ausgegeben.

3.1.4.6 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 der frei verfügbare Speicher in Gigabytes und in Prozent ausgegeben. Mit dem Befehl «awk» werden die Werte formatiert ausgegeben.

3.2 Auto Startup

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

3.2.1 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 werden, wann die Aufgaben ausgeführt werden müssen (Zeit oder Ereignisse wie z.B. Neustart). Jeder Benutzer hat ein Cron File.

Um dieses File zu öffnen muss folgendes Kommando ausgeführt werden:

```
crontab -e
```

In diesem Fall sieht der Eintrag für diese Aufgabe so aus:

```
@reboot /home/dev/documents/berry_case/firmware/python/autostart.sh
```

In anderen Worten, nach jedem Neustart führt das Skript «autostart.sh» aus.

3.2.2 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 im richtigen Directory befindet. Das geschieht mit dem «cd» Kommando. Danach wird das Hauptprogramm mit Python, als Administrator im Hintergrund ausgeführt.

3.3 Netzwerk

Das Netzwerk von Raspberry muss so aufgesetzt werden, dass beim Neustart nach einem bekannten Wlan-Netzwerk gesucht wird. Wenn eins gefunden wird, wird eine Verbindung hergestellt. Wenn kein bekanntes Netzwerk gefunden wurde, wird auf dem Raspberry ein Access Point eingerichtet.

Somit kann sich der Benutzer in jeder Situation mit dem Raspberry verbinden.

Auf dem Internet wurde ein Tool gefunden, mit dem das gewünschte Netzwerk eingerichtet werden kann. Dieses Tool wurde netterweise vom Besitzer der Webseite [_rasp-berrycorconnect.com](https://www.raspberrycorconnect.com) zu Verfügung gestellt.

Die Inbetriebnahme ist sehr intuitiv:

- Dateien mit folgendem Kommando herunterladen:

```
curl
"https://www.raspberrycorconnect.com/images/hsinstaller/AutoHotspotSetup.tar.gz" -o
AutoHotspot-Setup.tar.gz
```

- Dateien auspacken:

```
tar -xzf AutoHotspot-Setup.tar.gz
```

- Entpackte Ordner öffnen:

```
cd Autohotspot
```

- Installationsskript mit Administrationsrechte ausführen:

```
sudo ./autohotspot-setup.sh
```

- Ein Menü mit acht Einträgen wird angezeigt. «1» muss eingegeben werden, um den ersten Eintrag auszuwählen. Die Software wird nun installiert.

- Nachdem die Installation abgeschlossen wurde, muss das System neu gestartet werden.

3.3.1 Netzwerktopologie

3.3.1.1 Access Point

Falls keine bekannten Wlan-Netzwerke verfügbar sind, sieht die Netzwerktopologie wie folgt aus. Die Ethernet Verbindung ist fakultativ. Die DHCP wird vom Raspberry verwaltet.

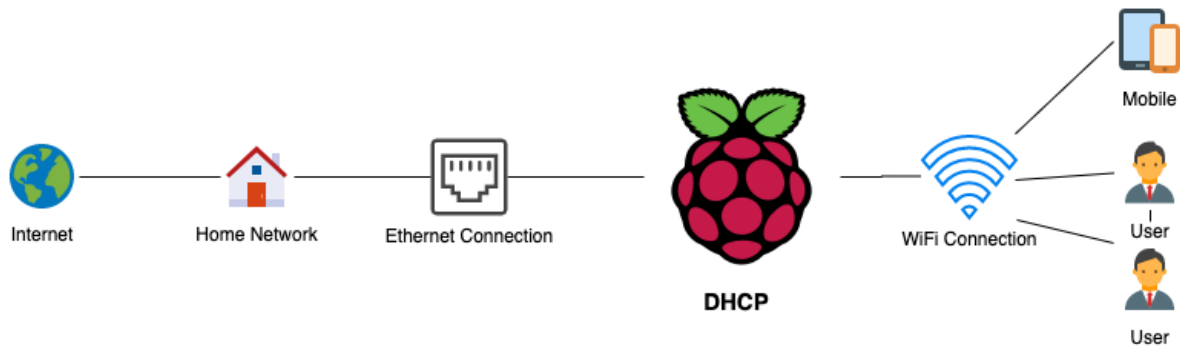


Abbildung 14: Netzwerktopologie Access Point

3.3.1.2 Eigene WiFi Netzwerk

Falls eine Verbindung zu einem bekannten Wlan-Netzwerk hergestellt werden kann, sieht die Netzwerktopologie wie folgt aus. Die DHCP wird vom Besitzer der WiFi AP verwaltet.

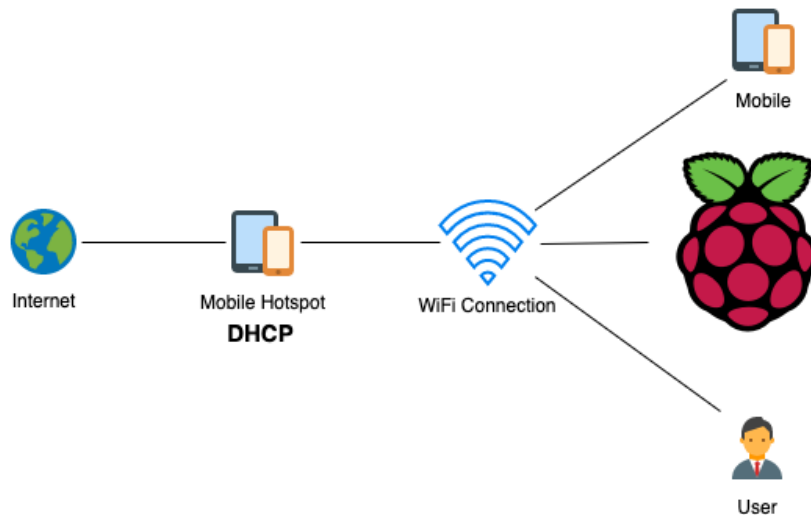


Abbildung 15: Netzwerktopologie eigene WiFi Network

4 Schlussbetrachtung

4.1 Fazit

Das berrycase Gehäuse erfüllt alle grundlegenden Ziele, welche im Pflichtenheft angegeben wurden. Es konnte auch angefangen werden, am optionalen Ziel zu Arbeiten. Dieses wurde aber nicht abgeschlossen. Das Gerät kann mit dem OLED Display und dem Navigationstaster angesteuert werden. Mit einem Akku kann der Raspberry auch unterwegs benutzt werden.

Alles in allem würde ich die Arbeit als gelungen betrachten. Ich konnte das Wissen, welches ich aus der Vorarbeit erarbeitet habe, gut anwenden und habe auch noch einiges an neuem Wissen dazugewonnen. Vor allem im Bereich Networking konnte ich noch einiges dazulernen. Obwohl ich nicht ein 3D Gehäuse zeichnen und erstellen konnte, war die Arbeit mit viel Spass verbunden und sehr abwechslungsreich.

4.2 Verbesserungsmöglichkeiten

Trotz sauberem Vorgehen während der Arbeit konnten noch einige kleine Fehler entdeckt werden, welche bei späteren Versionen korrigiert werden sollten.

4.2.1 Hardware

- Der Footprint des PowerBoost Bauteils muss korrigiert werden. Pin 6 und Pin 7 wurden vertauscht gezeichnet.
- Die Flip-Flop Schaltung kann verbessert werden. Der MOSFET Schaltung für das Set Pin das Flip-Flop kann mit ein Schottky Diode ersetzt werden.

4.2.2 Software

- Die Steuerung der Lüfter kann verbessert werden. Das PWM Signal ist nicht sehr sauber und präzise. Grund dafür ist, dass das Signal softwaremässig mit einer höheren Programmiersprache (Python) erzeugt wird. Die Steuerung der Lüfter könnte in C geschrieben werden.
- Die Entladekurve des eingesetzten Akkus kann in Form von einer Lookup Table im Hauptprogramm hinzugefügt werden. So kann der Ladezustand besser abgebildet werden.

4.3 Ausblicke

Ein kritischer Punkt, um die Beständigkeit des Geräts zu gewährleisten wäre die Herstellung eines geeigneten Gehäuses. Das Gehäuse wird sicherlich nach Projektabschluss noch entwickelt. Das Gerät wird auch weiterentwickelt und verbessert. Es ist nicht auszuschliessen, dass bei genügender Nachfrage das Gerät auch kommerzialisiert werden könnte.

5 Glossar

WORT	BESCHREIBUNG
EINPLATINEN-COMPUTER	Ein Einplatinencomputer ist ein Computersystem, bei dem sämtliche zum Betrieb nötigen elektronischen Komponenten auf einer einzigen Leiterplatte zusammengefasst sind.
GITHUB	GitHub ist ein netzbasierter Dienst zur Versionsverwaltung für Software-Entwicklungsprojekte.
ENTPRELLUNG	Durch die prellenden Schließvorgänge würde ohne Entprellung ein Tastenanschlag fehlerhafterweise als mehrfacher Anschlag registriert werden.
FREILAUFDIODE	Freilaufdioden sind Bauteile, welche zum Schutz vor Überspannung oder unerlaubten Spannungen eingesetzt werden.
PULL-UP	Elektrische Trennung zweier Stromkreise.
BASH	Bash, die Bourne-again shell, ist eine freie Unix-Shell unter GPL.
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.
ACCESS POINT	Ein Wireless Access (englisch für drahtloser Zugangspunkt), auch Access Point (AP) oder Basisstation genannt, ist ein elektronisches Gerät, das als Schnittstelle für kabellose Kommunikationsgeräte fungiert.
NETZWERKTOPOLOGIE	Die Topologie eines Rechnernetzes beschreibt die spezifische Anordnung der Geräte und Leitungen, die ein Rechnernetz bilden, über das die Computer untereinander verbunden sind und Daten austauschen.
FOOTPRINT	Ein Footprint ist die Anordnung von Pads (in der Oberflächenmontagetechnik) oder Durchgangslöchern (in der Durchstecktechnik), die zur physischen Befestigung und elektrischen Verbindung eines Bauteils mit einer Leiterplatte verwendet wird.
LOOKUP TABLE	Lookup-Tabellen bzw. Umsetzungstabellen werden in der Informatik und in der Digitaltechnik verwendet, um Informationen statisch zu definieren und diese zur Laufzeit des Programms – zur Vermeidung aufwändiger Berechnungen oder hohen Speicherverbrauchs – zu benutzen.

6 Literatur- und Quellenverzeichnis

Auf den folgenden Seiten werden alle verwendeten Quellen angegeben.

PowerBoost 1000C

Autor: lady ada

Link: <https://www.adafruit.com/product/2465> (Stand:13.12.2020)

OLED Display

Autor: Unbekannt

Link: https://github.com/adafruit/Adafruit_CircuitPython_SSD1306 (Stand: 13.12.2020)

Raspberry Pi 4 Schematics

Autor: James Adams

Link: https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/rpi_SCH_4b_4p0_reduced.pdf (Stand: 13.12.2020)

Introduction to USB Type-C™

Autor: Andrew Rogers

Link: <http://ww1.microchip.com/downloads/en/appnotes/00001953a.pdf>
(Stand: 13.12.2020)

Command to display Memory usage, Disk Usage and CPU Load

Autor: terdon

Link: <https://unix.stackexchange.com/questions/119126/command-to-display-memory-usage-disk-usage-and-cpu-load> (Stand: 13.12.2020)


Raspberry Pi Automatic Hotspot and Static Hotspot Installer

Autor: Unbekannt

Link: <https://www.raspberryconnect.com/projects/65-raspberrypi-hotspot-accesspoints/183-raspberry-pi-automatic-hotspot-and-static-hotspot-installer>
(Stand:03.04.2018)

7 Anhang

7.1 Zeitplan

Zeitplan																	
Nr.	Bezeichnung		19.08.20	26.08.20	02.09.20	09.09.20	16.09.20	14.10.20	21.10.20	28.10.20	04.11.20	11.11.20	18.11.20	25.11.20	02.12.20	09.12.20	16.12.20
1	Mindstorming	Soll															
		Ist															
2	Zeitplan erstellen	Soll															
		Ist															
3	Entwicklung Hardware Prototyp 1	Soll															
		Ist															
4	Entwicklung Linux Software	Soll															
		Ist															
5	Bestückung Hardware Prototyp 1	Soll															
		Ist															
6	Testing Prototyp 1	Soll															
		Ist															
7	Entwicklung Hardware Prototyp 2	Soll															
		Ist															
8	Entwicklung Hauptprogramm	Soll															
		Ist															
9	Bestückung Hardware Prototyp 2	Soll															
		Ist															
10	Testing Prototyp 2	Soll															
		Ist															
11	Dokumentation	Soll															
		Ist															
12	Präsentation	Soll															
		Ist															
13		Soll															
		Ist															
14		Soll															
		Ist															
15		Soll															
		Ist															

7.2 Aufgabestellung

berry_case

Zweck

Ich arbeite gerne im ÖV mit meinem Raspberry Pi 4 . Es ist aber sehr umständlich: eine Powerbank ist nötig und ein Netzwerk aufzusetzen ohne Bildschirm ist praktisch nicht möglich. Um die UX zu verbessern soll ein Gehäuse mit integriertem Akku für den Raspberry entwickelt werden.

Beschreibung / Funktion

Das Gehäuse muss klein gehalten werden. Eine UPS Funktion muss implementiert werden um im minimum zwei Stunden im Akkubetrieb arbeiten zu 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.

Aufbau

Alle Komponenten sollen auf einer einzigen Leiterplatte platziert werden so dass die Herstellungskosten tief gehalten werden können und der Zusammenbau selbsterklärend ist. Der Raspberry wird direkt über die GPIO gespiesen, gleichzeitig kommuniziert er über die gleiche HW Interface mit dem uC auf dem PCB (über USART oder I2C). Die Software besteht aus dem Programm für dem uC(C) und ein Scrip(python) die auf dem Raspberry als Daemon ausgeführt wird.

Optionale Anforderungen

Optional soll mithilfe eines CAD Tools ein Kunststoff Gehäuse gezeichnet und mit einem 3D Drucker ausgedruckt werden.

Umfang der IPA

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

Zusätzlich muss die Firmware für den Mikrokontroller und die Linux Software geschrieben werden. Folgende Arbeiten gehören zur IPA:

- Grobplanung
- Vorstudie
- Konzept erarbeiten
- Hardware Entwicklung
- Firmware / Software Entwicklung
- Testing / Bugfixing
- Dokumentation

Planung

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.

Dokumentation

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

Folgende Unterlagen werden am Abgabetermin zugestellt:

- Zeitplan
- Arbeitsjournal
- Dokumentation
- Entwicklungsunterlagen

Termine

Starttermin: 19.08.2020

Abgabetermin: 16.12.2020

Präsentationstermin: 13.01.2020, 20.01.2020 oder am 27.01.2020