Pflichtenheft

Würfel, Philipp

Lüders, Marius

Recktenwald, Gregor

*Smart Mini Camper*

Autor: Philipp Würfel , Marius Lüders, Gregor Recktenwald

Letzte Änderung: 17. Mai 2019

Dateiname: smart\_mini\_camper\_pflichtenheft.docx

Version: 1.2

***Inhaltsverzeichnis***

[1 Vorhandene Dokumente 4](#_Toc9020547)

[2 Überblick 5](#_Toc9020548)

[3 Hauptziele 6](#_Toc9020549)

[4 Annahmen und Abgrenzungen 6](#_Toc9020550)

[5 Workflow Smart Mini Camper 7](#_Toc9020551)

[6 Funktionalität 8](#_Toc9020552)

[6.1 Überblick 8](#_Toc9020553)

[6.2 Batteriestatus, Ladeleistung und Verbrauchsleistung bestimmen 9](#_Toc9020554)

[6.3 Messdaten in einer SQL-Datenbank hinterlegen 9](#_Toc9020555)

[6.4 Verbindung mit dem Pi herstellen 10](#_Toc9020556)

[6.5 Batteriestatus, Ladeleistung und Verbrauchsleistung anzeigen (aktuelle Daten) 10](#_Toc9020557)

[6.6 Diagramme anhand der Messdaten zeichnen 12](#_Toc9020558)

[6.7 Verbraucher an Batterie anschließen 14](#_Toc9020559)

[7 Benötigte Hardware 15](#_Toc9020560)

[8 Lizenzmodell 15](#_Toc9020561)

[9 Backlog 15](#_Toc9020562)

***Abbildungsverzeichnis***

[Abbildung 1 Kontextdiagramm Smart Mini Camper 5](#_Toc9020541)

[Abbildung 2 Workflow Smart Mini Camper 7](#_Toc9020542)

[Abbildung 3 Use-Case Diagramm Smart Mini Camper 8](#_Toc9020543)

[Abbildung 4 Mock-Up und Workflow zur Anzeige von Batteriestatus, Ladeleistung und Verbrauchsleistung 11](https://d.docs.live.net/86d4e8e7e265e328/Dokumente/Uni/Smart_Mini_Camper_Pflichtenheft.docx#_Toc9020544)

[Abbildung 5 Mock-Up und Workflow zum Zeichnen von Diagrammen anhand der Messdaten 13](https://d.docs.live.net/86d4e8e7e265e328/Dokumente/Uni/Smart_Mini_Camper_Pflichtenheft.docx#_Toc9020545)

***Copyright***

© SmartMiniCamper

Die Weitergabe, Vervielfältigung oder anderweitige Nutzung dieses Dokumentes oder Teile davon ist unabhängig vom Zweck oder in welcher Form untersagt, es sei denn, die Rechteinhaber/In hat ihre ausdrückliche schriftliche Genehmigung erteilt.

***Version Historie***

|  |  |  |  |
| --- | --- | --- | --- |
| *Version:* | *Datum:* | *Verantwortlich* | *Änderung* |
| 0.1 | 05.05.2019 | Gregor Recktenwald | Initiale Dokumenterstellung, Workflow,Kapitel 6, Kapitel 8 |
| 0.2 | 06.05.2019 | Philipp Würfel | Kontextdiagramm, UseCase Diagramm |
| 0.3 | 08.05.2019 | Marius Lüders, Gregor Recktenwald | Überarbeitung Texte, Kapitel 6 etc. |
| 0.4 | 12.05.2019 | Philipp Würfel | Überarbeitung UseCase Diagramm |
| 0.5 | 15.05.2019 | Marius Lüders | Kapitel 6 |
| 1.0 | 15.05.2019 | Philipp Würfel | Kapitel 7 und Kapitel 9, Entwurf Projektplan |
| 1.1 | 17.05.2019 | Marius Lüders | Formatierung, Einfügen vom Projektplan |
| 1.2 | 17.05.2019 | Philipp Würfel, Gregor Recktenwald | Anpassung Workflow, Bearbeitung Backlog, Finalisierung |

# Vorhandene Dokumente

Alle für die vorliegende Spezifikation ergänzenden Unterlagen müssen hier aufgeführt werden

|  |  |  |
| --- | --- | --- |
| Dokument | Autor | Datum |
| Lastenheft (Lastenheft.pdf) | Gregor Recktenwald, Marius Lüders,  Philipp Würfel | 26.04.2019 |
| Projektplan (ProjPlanSmartMiniCamper.mpp) | Gregor Recktenwald, Marius Lüders,  Philipp Würfel | 17.05.2019 |

# Überblick

An eine 12V Batterie werden elektrische Verbraucher angeschlossen. Zusätzlich wird die Batterie über einen Laderegler mit einem Solar-Panel geladen. Mittwoch einem Raspberry Pi und daran angeschlossenen Messgeräten soll die Lade- und Verbrauchsleistung, sowie der Batteriestatus bestimmt werden. In einer SQL-Datenbank werden diese Daten hinterlegt. Mittwoch einer Android App sollen die Daten vom Raspberry Pi über einen Web-Service übertragen und in entsprechenden Diagrammen veranschaulicht werden. Die Kommunikation erfolgt, wenn sich beide im selben Netzwerk befinden.

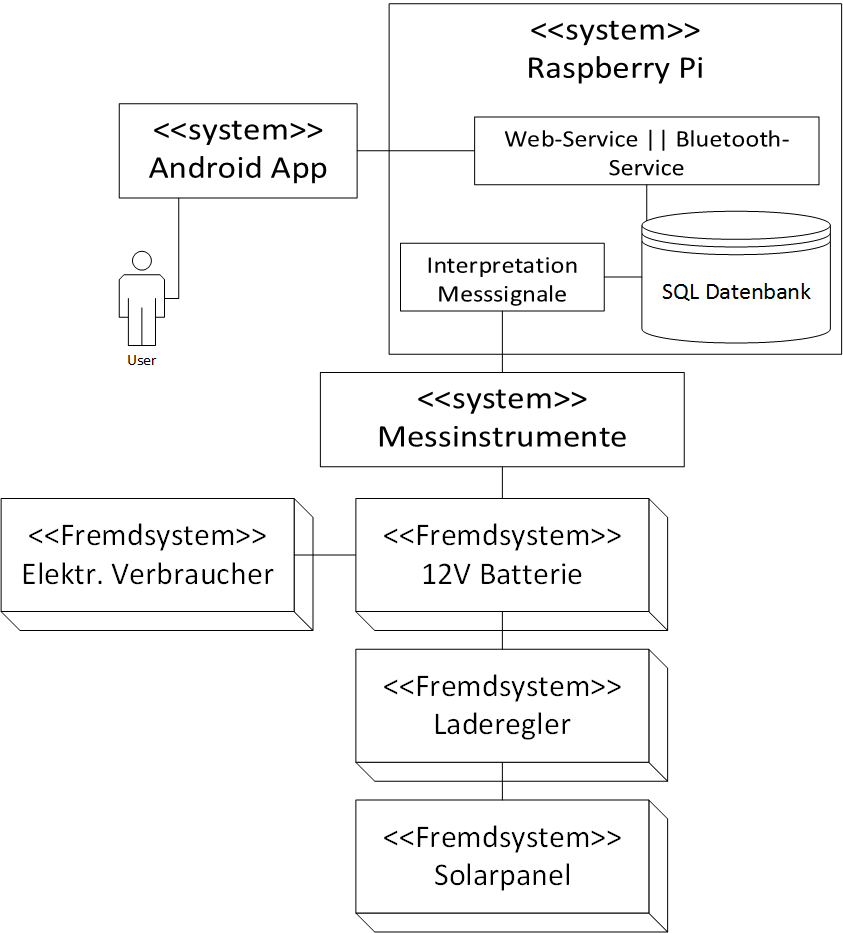


Abbildung 1 Kontextdiagramm Smart Mini Camper

# Hauptziele

| # | Ziel | Beschreibung der Implementation |
| --- | --- | --- |
| 1 | Nachvollziehbarkeit der Ladung und Entladung einer Batterie | GUI-functionality |
| 2 | Effektivere Nutzung des Solarpanels | GUI-functionality |
| 3 | Optimierung der Kommunikation | System |

# Annahmen und Abgrenzungen

| # | Annahmen (fachliche und technische Annahmen) |
| --- | --- |
| 1 | Anwender besitzen ein Smartphone |
| 2 | Anwender nutzen das Android Betriebssystem 6.0 (API 23) oder höher |
| 3 | Stromversorgung mit 12 V Autobatterie (minimale Stromstärke Input/Output : 30A) |

| # | Abgrenzungen (Was ist in dieser Lösung nicht enthalten bzw. abgedeckt) |
| --- | --- |
| 1 | Verbindung zum Pi durch etwas anderes als WLAN |
| 2 | Darstellung der Live-Daten vom Raspberry Pi (bspw. über LCD-Display) |
| 3 | Analyse der Daten auf Fehler bzw. Sinnhaftigkeit |

# Workflow Smart Mini Camper

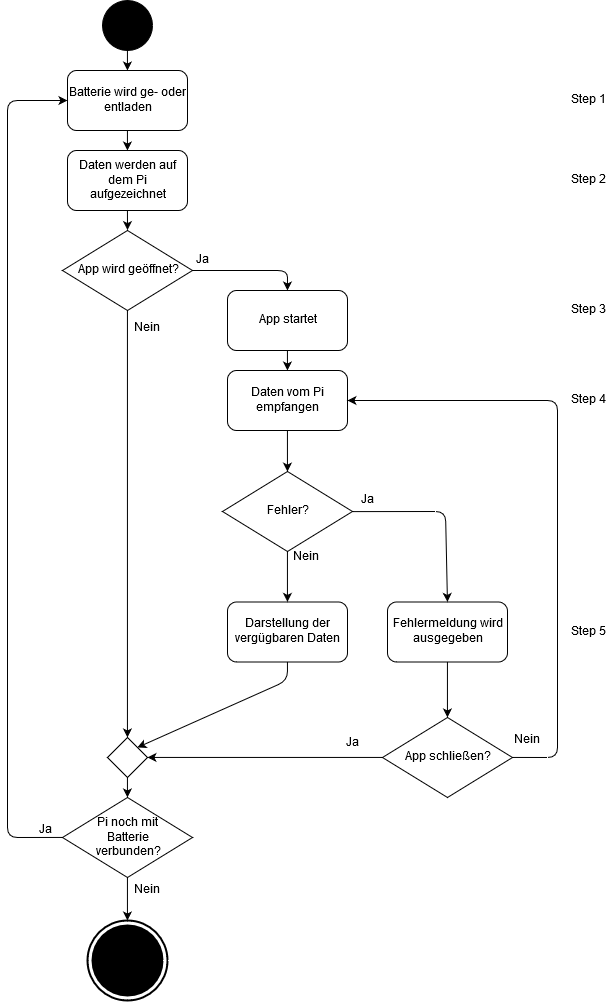


Abbildung 2 Workflow Smart Mini Camper

# Funktionalität

## Überblick



Abbildung 3 Use-Case Diagramm Smart Mini Camper

|  |  |
| --- | --- |
| Übersicht Hauptfunktionen | |
| 1 | Batteriestatus, Ladeleistung und Verbrauchsleistung bestimmen |
| 2 | Messdaten in SQL-Datenbank hinterlegen |
| 3 | Batteriestatus, Ladeleistung und Verbrauchsleistung anzeigen (aktuelle Daten) |
| 4 | Diagramme der Daten zeichnen |
| 5 | Verbraucher über Batterie laden |

## Batteriestatus, Ladeleistung und Verbrauchsleistung bestimmen

|  |  |
| --- | --- |
| Zweck/Ziel | Der Raspberry Pi soll durch die angeschlossenen Messinstrumente den Batteriestatus, Ladeleistung und Verbauchsleistung aufzeichnen |
| Akteur/Auslöser | Raspberry Pi (Messinstrumente) |
| Vorbedingung | * Messinstrumente sind an die Batterie und den Pi angeschlossen * Messinstrumente liefern Daten |
| Dateninput | Messdaten der Sensoren |
| Verarbeitungsschritte | 1. Messinstrumente zeichnen Daten auf (Spannung und Ampere) |
| Ergebnis | Verschiedene Messwerte werden aufgezeichnet, die vom Pi verarbeitet werden können. |
| Fehlerbehandlung | Wenn der Pi keine Messdaten gibt wartet er 10sec, ob noch Daten kommen und schreibt dann einen Fehlercode in die Datenbank, anstatt der Messdaten |
| Anforderung | Jedes Auf- oder Entladen soll aufgezeichnet werden |
| Test Cases | * Mittwoch verschiedenen Verbrauchern und mit und ohne Sonneneinstrahlung die Messinstrumente testen * Dummy Daten dem Pi geben um das aufzeichnen zu testen |

## Messdaten in einer SQL-Datenbank hinterlegen

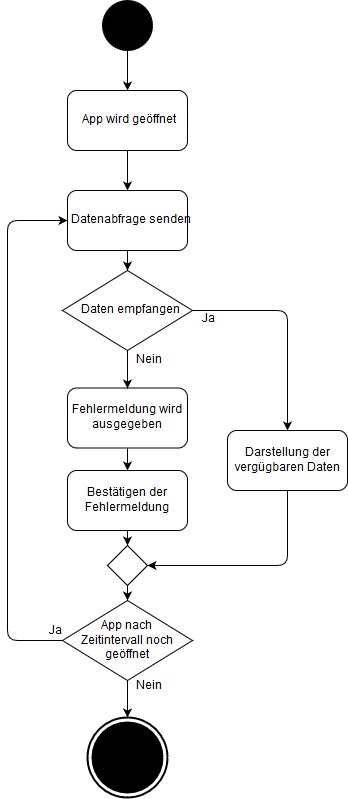
|  |  |
| --- | --- |
| Zweck/Ziel | Der Pi schreibt die aufgezeichneten Daten in eine SQL-Datenbank |
| Akteur/Auslöser | Raspberry Pi |
| Vorbedingung | * Messinstrumente sind an die Batterie und den Pi angeschlossen * Messinstrumente liefern Daten * Der Pi hat die Daten aufgezeichnet |
| Dateninput | Messdaten der Sensoren |
| Verarbeitungsschritte | Der Pi sendet der Datenbank die Daten und diese verarbeitet die Daten weiter |
| Ergebnis | Die Messdaten werden in einer Datenbank hinterlegt für einen einfacheren und sauberen Zugriff |
| Fehlerbehandlung | Wenn der Pi keine Verbindung mit der Datenbank aufnehmen kann, schreibt er sie in ein Textdokument, damit nichts verloren geht |
| Anforderung | Datenbank verarbeitet die Daten korrekt |
| Test Cases | Die Datenbank mit Dummy Daten füttern und überprüfen ob sie richtig weiterverarbeitet werden |

## Verbindung mit dem Pi herstellen

|  |  |
| --- | --- |
| Zweck/Ziel | Die App stellt eine Verbindung mit dem Pi her |
| Akteur/Auslöser | Smart Home App |
| Vorbedingung | * Das Handy ist mit dem Pi im selben Wlan * Der Webservice läuft auf dem Pi |
| Dateninput | Verbindungsanfrage der App |
| Verarbeitungsschritte | Die App sendet eine Verbindungsanfrage an den Pi und dieser bestätigt die Verbindung |
| Ergebnis | Eine Verbindung für den Datenaustausch besteht |
| Fehlerbehandlung | Sollte die Verbindung zum Pi fehlschlagen versucht die App es noch 2mal. Sollte dann immer noch keine Verbindung zustande kommen, wird eine Fehlermeldung gezeigt |
| Anforderung | Verbindung mit dem Pi kommt zustande |
| Test Cases | - |

## Batteriestatus, Ladeleistung und Verbrauchsleistung anzeigen (aktuelle Daten)

|  |  |
| --- | --- |
| Zweck/Ziel | Die App soll dem User live Daten zur Batterie zur Verfügung stellen (wird sie gerade geladen, entladen oder beides) |
| Akteur/Auslöser | User (durch das Öffnen der App) |
| Vorbedingung | * Verbindung zum Raspberry Pi * Öffnen der App |
| Daten-Input | Request/Anfrage von der App an den Pi und die Daten des Pis die von der App verarbeitet werden |
| Verarbeitungsschritte | 1. Beim Öffnen der App wird ein http Request an den Pi gesendet 2. Der Pi erhält den Request und holt die Daten aus der DB und sendet diese an das Smartphone 3. Darstellung der Daten |
| Ergebnis | Eine aktuelle Darstellung der Batteriedaten. Der User ist so in der Lage die Batterie zu überwachen und ein Gefühl für laden und entladen zu bekommen. |
| Fehlerhandling | Wenn die App keine Daten vom Pi empfängt, wird die Anfrage noch 2x mal wiederholt, sollte weiterhin keine Antwort kommen wird eine Fehlermeldung ausgegeben (Popup mit Text keine Verbindung) |
| Anforderung | Die Daten werden in Echtzeit dargestellt |
| Test Cases | Eine Minute lang mit verschiedenen Verbrauchern und mit und ohne Sonneneinstrahlung die Plausibilität der Werte überprüfen |



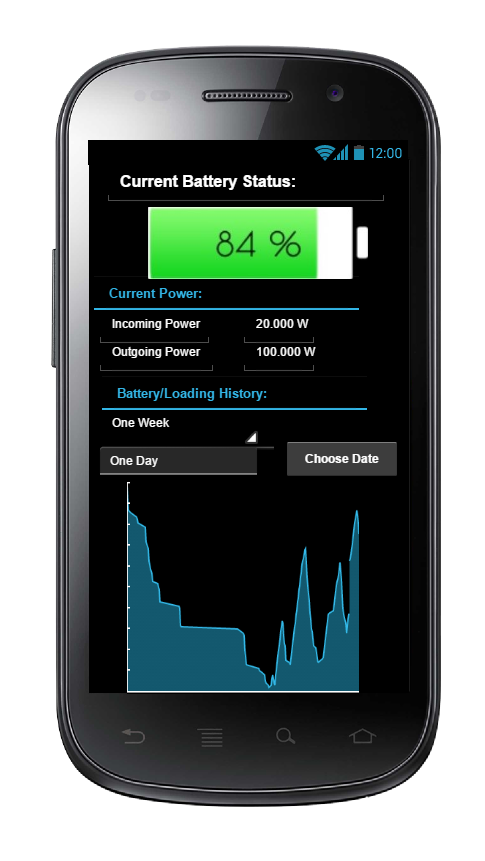


Abbildung 4 Mock-Up und Workflow zur Anzeige von Batteriestatus, Ladeleistung und Verbrauchsleistung

## Diagramme anhand der Messdaten zeichnen

|  |  |
| --- | --- |
| Zweck/Ziel | Die App soll dem User einen Überblick verschaffen wie viel Strom in dem von ihm ausgewählten Zeitraum verbraucht und gespeichert wurde |
| Akteur/Auslöser | User |
| Vorbedingung | * Aufzeichnung der Daten vom Raspberry Pi * Verbindung zum Raspberry Pi * Öffnen der App * Empfangen der Daten vom Raspberry Pi |
| Daten-Input | Die Information für welchen Zeitraum die Daten benötigt werden |
| Verarbeitungsschritte | 1. Öffnen der App 2. Auswählen des gewünschten Zeitraums 3. mit http Request nach gewünschtem Zeitraum fragen ( im Header den Zeitraum angeben) 4. Pi empfängt Zeitraum und holt die Daten für den ausgewählten Zeitraum aus der DB und sendet diese an das Smartphone 5. Smartphone stellt die Daten in Diagrammen dar |
| Ergebnis | Graphische Darstellung der Ladungs- und Entladungsdaten für den gewünschten Zeitraum. |
| Fehlerhandling | Wenn die App keine Daten vom Pi empfängt, wird die Anfrage noch 2x mal wiederholt, sollte weiterhin keine Antwort kommen wird eine Fehlermeldung ausgegeben (Popup mit Text keine Verbindung) |
| Anforderung | * Es kann auf die Daten verschiedener Zeiträume zugegriffen werden * Diese Daten können abgebildet werden |
| Test Cases | Mittwoch verschiedenen Testdaten Diagramme zeichnen lassen |

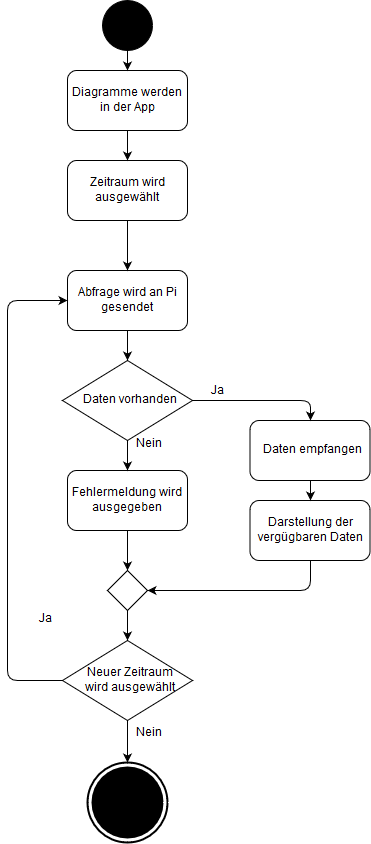




Abbildung 5 Mock-Up und Workflow zum Zeichnen von Diagrammen anhand der Messdaten

## Verbraucher an Batterie anschließen

|  |  |
| --- | --- |
| Zweck/Ziel | Laptop und Smartphones sollen über die App geladen werden können |
| Akteur/Auslöser | Anschluss von Verbrauchern an Batterie |
| Vorbedingung | * Batterie ist geladen * Wechselrichter mit entsprechender Leistung wurde angeschlossen |
| Daten-Input |  |
| Verarbeitungsschritte | 1. Batteriestatus über App überprüfen 2. Verbraucher an Wechselrichter anschließen |
| Ergebnis | Verbraucher werden von der Batterie über den Wechselrichter mit 230V Wechselspannung versorgt (Modifizierte |
| Fehlerhandling | Tiefentladeschutz vom Wechselrichter schaltet Stromzufuhr bei zu niedrigem Ladezustand der Batterie ab |
| Anforderung | Geräte werden geladen |
| Test Cases | * Verbraucher über Wechselrichter anschließen und überprüfen, ob diese genügend mit Strom versorgt werden |

# Benötigte Hardware

| # | Name | Beschreibung | Menge |
| --- | --- | --- | --- |
| 1 | Solarpanel WS120SF | 120 W Solarpanel zur Stromerzeugung | 1 |
| 2 | Solar Laderegler PEKO10 | 10A PWM-Laderegler für 12V Systeme | 1 |
| 3 | AGM Batterie | 12V AGM Batterie 20 Ah (max. Entladestrom 300A) | 1 |
| 4 | Wechselrichter | 150W Wechselrichter mit 12V Zigarettensteckdose | 1 |
| 5 | Stromstärkesensor ACS712 | 0-30A Input, Passive Stromstärkemessung mit dem Hall-Effekt, analoges Output-Signal | 2 |
| 6 | Spannungssenor | 0-25V Input, passiv gekühlte Spannungsmessung (Spannungsteiler), analoges Output-Signal | 1 |
| 7 | Analog-Digital-Wandler (ADC) MCP3008 | 8-Channel, 10-Bit ADC mit SPI Interface | 1 |
| 8 | Raspberry Pi 3 | Einplatinencomputer für die Messung, Kommunikation und Datenbank | 1 |

# Lizenzmodell

Das Thema Nachhaltigkeit und erneuerbare Energien wird in der Zukunft immer wichtiger. Daher wird der Quellcode für dieses Projekt wird mit der MIT-Lizenz zur Verfügung gestellt. So kann er von jedem für zukünftige Projekte frei genutzt und verändert werden.

# Backlog

|  |  |  |
| --- | --- | --- |
| Vorgangsnummer | Vorgangsname | Anforderungsreferenz (Lastenheft) |
|  | **Sprint 2** |  |
|  | **Technische Spezifikation** |  |
| 1 | Dokumentation Messsystem Raspberry Pi |  |
| 2 | Dokumentation Kommunikation Raspberry Pi -- App |  |
| 3 | Dokumentation App |  |
|  | **Messystem Raspberry Pi** |  |
| 4 | Raspberry Pi einrichten | 5.5 |
| 5 | Verbindung/Verschaltung der Hardwarekomponenten | 4.4, 4.5, 4.6, 4.7 |
| 6 | Entwicklung Messroutine auf Raspberry Pi | 4.1 |
| 7 | Probemessungen Ladung Batterie | 4.1 |
| 8 | Probemessungen Batteriestatus | 4.1 |
| 9 | Probemessungen Verbrauch | 4.1 |
| 10 | Test Messung im Gesamtsystem | 4.1 |
|  | **Kommunikation Raspberry Pi <--> App** |  |
| 11 | Webservice Planung | 4.2 |
| 12 | Webservice einrichten | 4.2 |
| 13 | Lokales WLAN einrichten | 4.2, 5.5 |
| 14 | Zugriff von App auf Webservice implementieren | 4.2, 4.3 |
|  | **App Entwicklung** |  |
| 15 | Design der App | 5.1, 5.3, 5.6 |
| 16 | Menüansichten erstellen | 5.3, 5.6 |
| 17 | In GUI Daten (ggf. Dummy) von Webservice empfangen | 5.4, 5.6 |
| 18 | Empfangene Daten von Webservice anzeigen | 5.6 |
|  | **Sprint 3** |  |
|  | **Technische Spezifikation** |  |
| 19 | Dokumentation Raspberry Pi |  |
| 20 | Dokumentation App |  |
| 21 | Datenbankmodell entwickeln | 4.1 |
|  | **Messystem Raspberry Pi** |  |
| 22 | Datenbank auf Raspberry Pi einrichten | 4.1 |
| 23 | Optimierung der Messungen | 4.1 |
| 24 | Optimierung der Messroutine | 4.1 |
| 25 | Übertragung Messdaten an Datenbank | 4.1, 5.2 |
| 26 | Test Messung im Gesamtsystem |  |
|  | **Kommunikation Raspberry Pi <--> App** |  |
| 27 | Implementierung JSON als Kommunikationsformat | 4.2, 4.3 |
| 28 | Zugriff von Webservice auf Datenbank implementieren | 4.1, 4.2, 4.3 |
|  | **App Entwicklung** |  |
| 29 | Weiterentwicklung App Design | 5.1, 5.3, 5.6 |
| 30 | In GUI Daten von Webservice empfangen | 4.3, 5.2, 5.4, 5.7 |
| 31 | JSON-Parsing der empfangenen Daten | 4.3 |
| 32 | Darstellung der Daten | 4.8 |
| 33 | Verschiedene Darstellungsvarianten entwickeln | 4.8 |
| 34 | App Testen |  |
|  | **Sprint 4** |  |
|  | **Technische Spezifikation** |  |
| 35 | Dokumentation Raspberry Pi |  |
| 36 | Dokumentation App |  |
|  | **Messystem Raspberry Pi** |  |
| 37 | Optimierung und Finalisierung Messsystem | 4.1 |
| 38 | Testfälle entwickeln und durcharbeiten |  |
|  | **Kommunikation Raspberry Pi <--> App** |  |
| 39 | Optimierung und Finalisierung Kommunikation | 4.2, 4.3, 5.5 |
| 40 | Testfälle entwickeln und durcharbeiten |  |
|  | **App Entwicklung** |  |
| 41 | Optimierung und Finalisierung App | 4.8, 5.1, 5.2, 5.3, 5.4, 5.6, 5.7 |
| 42 | Testfälle entwickeln und durcharbeiten |  |
|  | **Projektmanagement** |  |
| 43 | Projektplan 1 |  |
| 44 | Projektplan 2 |  |