Qualitätssicherung

HTW-Berlin

Smart Mini Camper

Autor: Gregor Recktenwald, Philipp Würfel, Marius Lüders

Letzte Änderung: 7. Juni 2019

Dateiname: Qualitätssicherung\_Smart\_Mini\_Camper.docx

Version: 1.0

***Inhaltsverzeichnis***

[1 Testfälle 4](#_Toc10836814)

[1.1 Serveranbindung 4](#_Toc10836815)

[1.2 Datenbankanbindung 5](#_Toc10836816)

[1.3 App JSON-Daten lesen 6](#_Toc10836817)

[1.4 App Darstellung von JSON-Daten 7](#_Toc10836818)

[1.5 App Anbindung zum Webservice 8](#_Toc10836819)

[1.6 Spannung mit Sensoren messen 9](#_Toc10836820)

[1.7 Batterie mit Solarpanel laden 9](#_Toc10836821)

[1.8 Verbraucher anschließen 10](#_Toc10836822)

[1.9 Messung Ladestrom 10](#_Toc10836823)

[1.10 Messung Laststrom (Verbraucherseite) 11](#_Toc10836824)

***Version Historie***

|  |  |  |  |
| --- | --- | --- | --- |
| *Version:* | *Datum:* | *Verantwortlich* | *Änderung* |
| 0.1 | 06.06.2019 | Marius Lüders | Initiale Dokumenterstellung, Überarbeitung der Vorlage, Testfälle |
| 1.0 | 07.06.2019 | Gregor Recktenwald, Philipp Würfel, Marius Lüders | Überarbeitung der Testfälle, Erstellung weitere Testfälle |

***Vorhandene Dokumente***

Alle für die vorliegende Spezifikation ergänzenden Unterlagen müssen hier aufgeführt werden

|  |  |  |
| --- | --- | --- |
| Dokument | Autor | Datum |
| Smart\_Mini\_Camper\_Lastenheft.doc | Gregor Rechtenwald, Philipp Würfel,  Marius Lüders | 25.04.2019 |
| Smart\_Mini\_Camper\_Pflichtenheft.doc | Gregor Rechtenwald, Philipp Würfel,  Marius Lüders | 17.05.2019 |

# 

# Testfälle

## Serveranbindung

| **Testfall** | **Beschreibung** |
| --- | --- |
| Testfall-Nummer | 1 |
| Testart | Funktionstest |
| Zu testender Geschäftsprozess/  Zu testende Funktionsgruppe | Testen ob der Server ansprechbar ist und über das Netz erreichbar ist |
| Testziel | Der Server sendet Daten an den Browser |
| Testvoraussetzungen | * RESTFul Service auf dem Raspberry Pi muss aktiv sein * Testgerät muss mit dem Wlan des Pi´s verbunden sein |
| Testfalldaten | * Der Laptop verbindet sich mit dem Wlan des Pis * Internetbrowser starten * IP des Pi mit der zugehörigen Adresse eingeben   JSON:  [  {  "currentBat": 11.8,  "currentIn": 23.11,  "currentOut": 22.2,  "date": "2019-06-03 10:22:09",  "id": 1  },  {  "currentBat": 12,  "currentIn": 21,  "currentOut": 22,  "date": "2019-06-03 12:44:47",  "id": 2  }  ] |
| Erwartetes Verhalten | * Der Raspberry Pi sendet die Json-Datei an den Laptop |

## Datenbankanbindung

| **Testfall** | **Beschreibung** |
| --- | --- |
| Testfall-Nummer | 2 |
| Testart | Funktionstest |
| Zu testender Geschäftsprozess/  Zu testende Funktionsgruppe | Testen ob der Server Daten aus der Datenbank abrufen kann und versenden kann |
| Testziel | Der Server sendet Daten an den Browser |
| Testvoraussetzungen | * RESTFul Service auf dem Raspberry Pi muss aktiv sein * Testgerät muss mit dem Wlan des Pi´s verbunden sein * Die Datenbank muss im Verzeichnis vom RESTFul sein (SQLite3) |
| Testfalldaten | * Der Laptop verbindet sich mit dem Wlan des Pis * Internetbrowser starten * IP des Pi mit der zugehörigen Adresse eingeben   JSON:  [  {  "currentBat": 11.8,  "currentIn": 23.11,  "currentOut": 22.2,  "date": "2019-06-03 10:22:09",  "id": 1  },  {  "currentBat": 12,  "currentIn": 21,  "currentOut": 22,  "date": "2019-06-03 12:44:47",  "id": 2  }  ] |
| Erwartetes Verhalten | * Der Pi sendet die Json-Datei an den Laptop |

## App JSON-Daten lesen

| **Testfall** | **Beschreibung** |
| --- | --- |
| Testfall-Nummer | 3 |
| Testart | Funktionstest |
| Zu testender Geschäftsprozess/  Zu testende Funktionsgruppe | Testen ob eine lokale JSON Datei von der App verarbeitet werden kann |
| Testziel | App kann JSON Daten verarbeiten |
| Testvoraussetzungen | * Es muss eine lokale JSON Datei erstellt werden * Die App muss lesezugriff auf die JSON Datei haben |
| Testfalldaten | * Die App liest die JSON Daten ein * Die App zeigt die JSON Daten an |
| Erwartetes Verhalten | * Die App liest die JSON Datei ein   JSON:  [  {  "currentBat": 11.8,  "currentIn": 23.11,  "currentOut": 22.2,  "date": "2019-06-03 10:22:09",  "id": 1  },  {  "currentBat": 12,  "currentIn": 21,  "currentOut": 22,  "date": "2019-06-03 12:44:47",  "id": 2  },  {  "currentBat": 12,  "currentIn": 48,  "currentOut": 22.4,  "date": "2019-06-03 14:54:37",  "id": 3  },  {  "currentBat": 12,  "currentIn": 28,  "currentOut": 22.1,  "date": "2019-06-03 14:55:01",  "id": 4  },  {  "currentBat": 12,  "currentIn": 28.12,  "currentOut": 25.1,  "date": "2019-06-03 14:55:17",  "id": 5  },  {  "currentBat": 12,  "currentIn": 32,  "currentOut": 28,  "date": "2019-06-03 14:55:30",  "id": 6  },  {  "currentBat": 12,  "currentIn": 35,  "currentOut": 33,  "date": "2019-06-03 14:55:38",  "id": 7  },  {  "currentBat": 12,  "currentIn": 38,  "currentOut": 44,  "date": "2019-06-03 14:55:46",  "id": 8  },  {  "currentBat": 12,  "currentIn": 43,  "currentOut": 49,  "date": "2019-06-03 14:56:10",  "id": 9  },  {  "currentBat": 12,  "currentIn": 48,  "currentOut": 52,  "date": "2019-06-03 14:56:19",  "id": 10  }  ]   * Die Daten werden auf dem Smartphone entsprechend im Logcat oder auf dem Bildschirm angezeigt |

## App Darstellung von JSON-Daten

| **Testfall** | **Beschreibung** |
| --- | --- |
| Testfall-Nummer | 4 |
| Testart | Funktionstest |
| Zu testender Geschäftsprozess/  Zu testende Funktionsgruppe | Testen ob verschiedene Daten in einer JSON Datei als Diagramm dargestellt werden können |
| Testziel | Die App stellt die Messdaten als x-Werte über den Datumsangaben als y-Werte dar |
| Testvoraussetzungen | Die App muss Leserechte auf die JSON Datei haben und in der JSON Datei müssen verschiedene Daten sein mit jeweils verschiedenen Datumsangaben |
| Testfalldaten | * Die App liest die Daten der JSON * Die App zeichnet ein Diagramm mit den Daten auf der X-Achse |
| Erwartetes Verhalten | * Die JSON wird von der App gelesen und die Daten als Diagramm dargestellt |

## App Anbindung zum Webservice

| **Testfall** | **Beschreibung** |
| --- | --- |
| Testfall-Nummer | 5 |
| Testart | Funktionstest |
| Zu testender Geschäftsprozess/  Zu testende Funktionsgruppe | Testen ob die App Daten vom Server aus der Datenbank empfangen kann |
| Testziel | Der Server sendet Daten an die App, welche diese im Logcat oder per Toast anzeigt |
| Testvoraussetzungen | * Pi muss angeschaltet sehn * RESTFul Service muss aktiv sein * Die Datenbank muss im Verzeichnis vom RESTFul sein (SQLite3) * App muss über die IP eine Verbindung Webservice auf dem Raspberry Pi aufbauen |
| Testfalldaten | * Eingabe der IP in App mit der zugehörigen Adresse   IP:5000/smartminicamper/db |
| Erwartetes Verhalten | * Der Pi sendet die Json-Datei an die App * Ausgabe in Logcat oder bei Bedarf per Toast auf App-Bildschirm   JSON:  [  {  "currentBat": 11.8,  "currentIn": 23.11,  "currentOut": 22.2,  "date": "2019-06-03 10:22:09",  "id": 1  },  {  "currentBat": 12,  "currentIn": 21,  "currentOut": 22,  "date": "2019-06-03 12:44:47",  "id": 2  }  ] |

## Spannung mit Sensoren messen

| **Testfall** | **Beschreibung** |
| --- | --- |
| Testfall-Nummer | 6 |
| Testart | Funktionstest |
| Zu testender Geschäftsprozess/  Zu testende Funktionsgruppe | Spannung messen |
| Testziel | Es wird die Spannung einer Batterie gemessen |
| Testvoraussetzungen | Sensoren müssen parallel an die Batterie angeschlossen sein, die die Spannung der Batterie messen.  Es wird eine Batterie angeschlossen, welche eine Spannung von 12,5 V besitzt. |
| Testfalldaten | * Sensoren werden an die Batterie angeschlossen * Die Sensoren messen die Spannung der Batterie * Der gemessene Spannungswert wird angezeigt |
| Erwartetes Verhalten | * Der gemessen Spannungswert der Batterie beträgt 12,5 V   Python Console:   * Output Channel 3: ca. 12,5 V |

## Batterie mit Solarpanel laden

| **Testfall** | **Beschreibung** |
| --- | --- |
| Testfall-Nummer | 7 |
| Testart | Funktionstest |
| Zu testender Geschäftsprozess/  Zu testende Funktionsgruppe | Laden der Batterie mit einem Solarpanel |
| Testziel | Die Batterie wird mit Hilfe des Solarpanels geladen |
| Testvoraussetzungen | Das Solarpanel ist mit der Batterie verbunden.  Die Sonne scheint auf das Solarpanel  Die Batterie ist nicht vollständig geladen |
| Testfalldaten | * Vor dem Laden wird die Spannung der Batterie gemessen * Solarpanel wird bei Sonnenschein an die Batterie angeschlossen * Nach mindestens 30 min wird die Spannung der Batterie gemessen |
| Erwartetes Verhalten | * Die gemessene Spannung ist nach dem Laden größer als davor |

## Verbraucher anschließen

| **Testfall** | **Beschreibung** |
| --- | --- |
| Testfall-Nummer | 8 |
| Testart | Funktionstest |
| Zu testender Geschäftsprozess/  Zu testende Funktionsgruppe | Verbraucher mit Hilfe der Batterie mit Strom versorgen |
| Testziel | Einen Verbraucher (bspw. Laptop) an die Batterie anschließen |
| Testvoraussetzungen | Die Batterie muss ausreichend geladen sein.  Der Verbraucher (Laptop) muss an den Power Converter angeschlossen sein  Der Power Converter muss mit der Batterie verbunden und eingeschalten sein. |
| Testfalldaten | * Power Converter wird ein­­­geschalten * Batterie wird entladen * Verbraucher werden über einen Zeitraum von 15 min geladen |
| Erwartetes Verhalten | * Der Verbraucher wird mit Strom versorgt |

## Messung Ladestrom

| **Testfall** | **Beschreibung** |
| --- | --- |
| Testfall-Nummer | 9 |
| Testart | Funktionstest |
| Zu testender Geschäftsprozess/  Zu testende Funktionsgruppe | Messungen des eingehenden Stroms vom Solarpanel |
| Testziel | Anhand der Messdaten prüfen ob die Daten des eingehenden Stroms plausibel sind |
| Testvoraussetzungen | Das Solarpanel ist mir der Batterie und Messinstrumenten verbunden.  Testdaten sind noch ungenau, da noch nicht fein genug kalibriert. |
| Testfalldaten | Es werden Messdaten für den eingehenden Strom angezeigt (Nullpunkt liegt bei ca. 512, die Kalibrierung und verlässliche Umrechnung in Ampere erfolgt im nächsten Sprint)   * Fall 1: Es wird bei guter Sonneneinstrahlung gemessen (Solarpanel sollte mindestens 30 W liefern) * Fall 2: Es wird ohne Sonneneinstrahlung gemessen (Solarpanel liefert 0 W) |
| Erwartetes Verhalten | Python Console:   * Fall 1: Output Channel 2, mit Stromfluss:   Digital Value > 512 || Digital Value > 512   * Fall 2: Output Channel 2, ohne Stromfluss:   Digital Value ca. 512 |

## Messung Laststrom (Verbraucherseite)

| **Testfall** | **Beschreibung** |
| --- | --- |
| Testfall-Nummer | 10 |
| Testart | Funktionstest |
| Zu testender Geschäftsprozess/  Zu testende Funktionsgruppe | Messungen des ausgehenden Stroms auf der Verbraucherseite |
| Testziel | Anhand der Messdaten prüfen ob die Daten des ausgehenden Stroms plausibel sind |
| Testvoraussetzungen | Power Converter ist mit Batterie und Messinstrumenten verbunden.  An dem Power Converter wird ein Verbraucher angeschlossen (bspw. Laptop)  Testdaten sind noch ungenau, da noch nicht fein genug kalibriert. |
| Testfalldaten | Es werden Messdaten für den ausgehenden Strom angezeigt (Nullpunkt liegt bei ca. 512, die Kalibrierung und verlässliche Umrechnung in Ampere erfolgt im nächsten Sprint)   * Fall 1: Power Converter ist eingeschalten und Verbraucher sind angeschlossen (Verbraucher sollte mindestens 30 W beziehen) * Fall 2: Power Converter ausgeschalten (Last = 0 W) |
| Erwartetes Verhalten | Python Console:   * Fall 1: Output Channel 2, mit Last:   Digital Value > 512 || Digital Value > 512   * Fall 2: Output Channel 2, ohne Stromfluss:   Digital Value ca. 512 |