

**Wetterstation mit Raspberry Pi**

Version: 0.5

Datum: 15.11.14

Projektteam: Andreas Hasler / David Daniel

Inhaltsverzeichnis

[1Informationen zum Dokument 4](#__RefHeading__1170_183045321)

[1.1Zweck des Dokuments 4](#__RefHeading__1172_183045321)

[1.2Versionskontrolle 4](#__RefHeading__1174_183045321)

[1.3Referenzierte Dokumente 4](#__RefHeading__1176_183045321)

[2Projektdefinition 5](#__RefHeading__1178_183045321)

[3Anforderungen 5](#__RefHeading__1180_183045321)

[3.1Funktionale Anforderungen 5](#__RefHeading__1182_183045321)

[3.2Nicht funktionale Anforderungen 6](#__RefHeading__1184_183045321)

[4Kontextdiagramm 6](#__RefHeading__1186_183045321)

[5Terminplan 7](#__RefHeading__1188_183045321)

[6Use Cases 8](#__RefHeading__1190_183045321)

[6.1Diagramm 8](#__RefHeading__1192_183045321)

[6.2Beschreibungen 9](#__RefHeading__1194_183045321)

[7Grobentwurf 13](#__RefHeading__1196_183045321)

[7.1Hardware 13](#__RefHeading__1198_183045321)

[7.1.1Direkte Anbindung 13](#__RefHeading__1200_183045321)

[7.1.2Indirekte Anbindung 13](#__RefHeading__1202_183045321)

[7.2Steuerung / Online-Schnittstelle 14](#__RefHeading__1204_183045321)

[7.2.1Webseite 14](#__RefHeading__1206_183045321)

[7.2.2Webservice 15](#__RefHeading__1208_183045321)

[7.2.3Smartphone-App mit Webservice-Zugriff 15](#__RefHeading__1210_183045321)

[7.3 Lösungsfindung 15](#__RefHeading__1212_183045321)

[7.3.1Hardware 15](#__RefHeading__1214_183045321)

[7.3.2Software 16](#__RefHeading__1216_183045321)

[8Detailentwurf 17](#__RefHeading__1218_183045321)

[8.1Hardware / Schaltung 17](#__RefHeading__1220_183045321)

[8.2Steuerung 17](#__RefHeading__1222_183045321)

[Analyse 17](#__RefHeading__1224_183045321)

[Synthese 21](#__RefHeading__1226_183045321)

[Struktur 21](#__RefHeading__1228_183045321)

[8.3Datenbank 23](#__RefHeading__1230_183045321)

[8.4Webseite 24](#__RefHeading__1232_183045321)

[9Qualitätssicherung 25](#__RefHeading__1234_183045321)

[Testing 26](#__RefHeading__1236_183045321)

[TC1 - Ermitteln der Werte 26](#__RefHeading__1238_183045321)

[TC2 – Speicherung 26](#__RefHeading__1240_183045321)

[TC3 - Aktuelle Daten über die Web Applikation 26](#__RefHeading__1242_183045321)

[TC4 - Gespeicherte Daten über die Web Applikation 26](#__RefHeading__1244_183045321)

[TC5 - Ausfall aller Sensoren 26](#__RefHeading__1246_183045321)

[TC6 – Speicherplatz 26](#__RefHeading__1248_183045321)

1. Informationen zum Dokument
   1. Zweck des Dokuments

Dieses Dokument beinhaltet die Projektdokumentation zum Projekt *Wetterstation*, welches im Zuge des 9. Semesters im Fach Embedded Systems und Hardware Hacking an der FFHS umgesetzt wurde.

* 1. Versionskontrolle

|  |  |  |  |
| --- | --- | --- | --- |
| Ausgabe | Datum | Autor | Bemerkungen |
| 0.1 | 27.09.2014 | Andreas Hasler | Initialversion |
| 0.2 | 28.09.2014 | Andreas Hasler | Anpassungen Anforderungen und Terminplan |
| 0.3 | 29.09.2014 | Andreas Hasler | Anpassungen Anforderungen |
| 0.4 | 10.10.2014 | Andreas Hasler | Use-Cases hinzugefügt |
| 0.5 | 25.10.2014 | Andreas Hasler | Grobentwurf / Lösungsfindung |

* 1. Referenzierte Dokumente

|  |
| --- |
| Dokument / Bemerkungen |
| Präsenz Block 2 (27.09.2014) mit der Aufgabenstellung auf Seite 11 |

1. Projektdefinition

Mit dem Raspberry Pi soll eine Wetterstation erstellt werden, welche Wetterdaten (Luftdruck, Temperatur, Feuchtigkeit und Lichtstärke) ermittelt und auf einem Display alternierend darstellt. Zusätzlich sollen die Wetterdaten auf dem Raspberry Pi in einer Datenbank persistent abgespeichert werden, so dass die aktuellsten Daten Online eingesehen werden können.

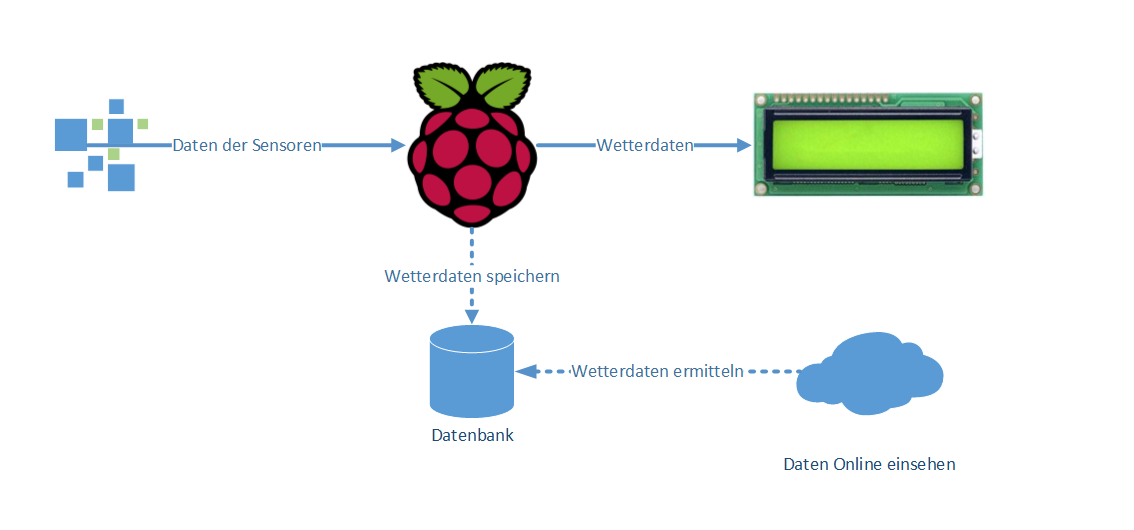


Abbildung: Projektidee (Skizze)

1. Anforderungen

Nachfolgend werden die funktionalen sowie die nicht funktionalen Anforderungen an das System beschrieben. Bei den funktionalen Anforderungen handelt es sich ausschliesslich um Muss-Anforderungen.

* 1. Funktionale Anforderungen
* Die Wetterdaten (Luftdruck, Temperatur, Feuchtigkeit und Lichtstärke) sind mittels Sensoren zu ermitteln.
* Die ermittelten Wetterdaten sind persistent in einer Datenbank abzuspeichern.
* Des Weiteren sind die ermittelten Wetterdaten auf einem Display auszugeben.
* Die aktuellsten Wetterdaten müssen Online eingesehen werden können.
  1. Nicht funktionale Anforderungen
* Die Projekt muss am 14.12.2014 (inkl. Dokumentation) abgeschlossen sein
* Das Projekt muss mittels Präsentation am 20.12.2014 anlässlich der 5. Präsenz vorgestellt werden.
* Die Signal- und Datenverarbeitung hat auf dem Raspberry Pi zu erfolgen.

1. Kontextdiagramm

Nachfolgend wird das Kontextdiagramm des Projekts (inkl. den Kann-Zielen) darstellt:

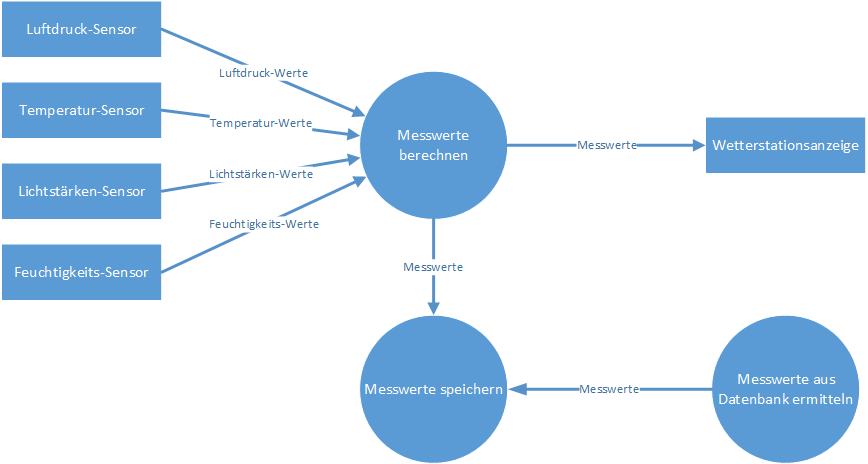


Abbildung: Kontextdiagramm des Projekts

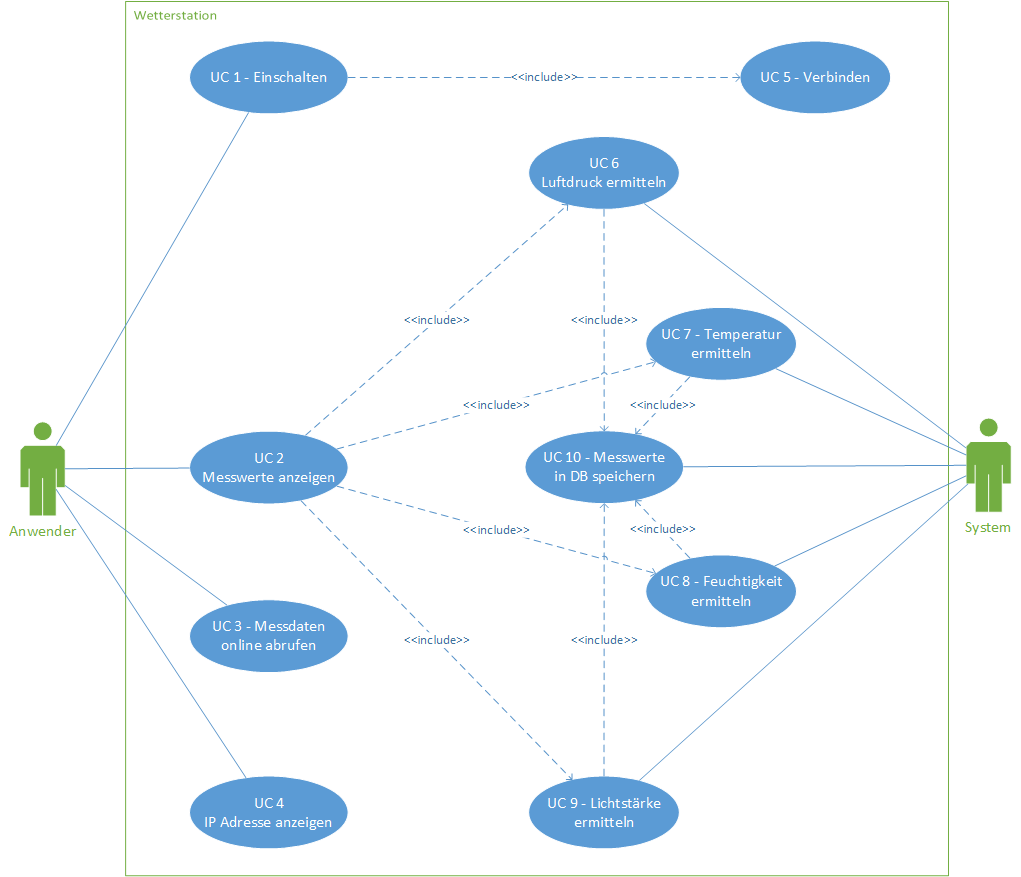
Die Werte der einzelnen Sensoren werden ermittelt und in einer zentral berechnet / umgerechnet. Anschliessend werden die Daten an die Anzeige weitergeleitet. Zudem werden die Messwerte nach der Berechnung persistent abgespeichert, damit die Daten Online abgefragt werden können.

1. Terminplan

|  |  |
| --- | --- |
| Bezeichnung | Termin |
| **Projektskizze erstellt** | **05.10.2014** |
| **Anforderungen / Kontextdiagramm / Terminplan** | **12.10.2014** |
| Use-Cases erstellen / verifizieren | 19.10.2014 |
| Lösungsentwürfe erstellen (Grobentwurf) / Lösungsfindung | 26.10.2014 |
| **Schaltungsentwurf / Softwareentwurf / Testkonzept** | **16.11.2014** |
| Schaltung / Hardware umsetzen | 22.11.2014 |
| Software implementieren (Ermittlung Messwerte, Weitergabe der Messwerte an den LCD-Bildschirm) | 07.12.2014 |
| Applikationstest und Abnahme | 14.12.2014 |
| **Projektdokumentation finalisieren** | **14.12.2014** |
| **Präsentation anlässlich Präsenz 5** | **20.12.2014** |

Die Meilensteine (Abgaben in moodle) sind Fett markiert und sind zwingend einzuhalten.

1. Use Cases
   1. Diagramm



* 1. Beschreibungen

|  |  |
| --- | --- |
| **UC 1 - Einschalten** | |
| Beschreibung | Die Wetterstation einschalten |
| Stakeholder | Anwender |
| Uses | *UC 5 - Verbinden* |
| Vorbedingungen | Die Wetterstation ist noch nicht eingeschaltet |
| Nachbedingungen | Die Wetterstation ist eingeschaltet |
| Ablauf | Verbindung Wetterstation / 230V Steckdose mit Netzteil |

|  |  |
| --- | --- |
| **UC 2 - Messwerte anzeigen** | |
| Beschreibung | Die unterschiedlichen Messwerte sollen auf dem LCD Display der Wetterstation angezeigt werden |
| Stakeholder | Anwender |
| Uses | *UC 6 - Luftdruck ermitteln, UC 7 - Temperatur ermitteln, UC 8 - Feuchtigkeit ermitteln, UC 9 - Lichtstärke ermitteln* |
| Vorbedingungen | * Wetterstation eingeschaltet * Messwerte durch die Sensoren ermittelt |
| Nachbedingungen | Messwerte werden auf dem LCD-Display angezeigt |
| Ablauf | 1. Anwender betätigt den Schalter 1 am LCD-Display (gilt nur wenn der Schalter 2, 3 oder 4 zuvor betätigt wurde, ansonsten werden die Messwerte standardmässig anzeigt) 2. Messwerte werden auf dem LCD-Display angezeigt (Pro LCD-Reihe ein Messwert) |

|  |  |
| --- | --- |
| **UC 3 – Messdaten online abrufen** | |
| Beschreibung | Die Messdaten dem Anwender Online zur Verfügung stellen |
| Stakeholder | Anwender |
| Vorbedingungen | * Messdaten in der Datenbank vorhanden * IP-Adresse des Raspberry Pi bekannt |
| Nachbedingungen | Die Messwerte konnten Online ermittelt werden |
| Ablauf | 1. Anwender verbindet sich mittels der bekannten IP-Adresse mit der Schnittstelle auf dem Raspberry Pi 2. Messwerte werden aus der Schnittstelle auf Grund der Angabe des Datumbereichs (resp. des aktuellen Wertes) ausgelesen |

|  |  |
| --- | --- |
| **UC 4 - IP Adresse anzeigen** | |
| Beschreibung | Die IP-Adresse des Raspberry Pi wird auf dem LCD-Display angezeigt (für Fernwartung oder Zugriffe auf die Online-Schnittstelle) |
| Stakeholder | Anwender |
| Vorbedingungen | Wetterstation eingeschaltet |
| Nachbedingungen | IP-Adresse des Raspberry Pi wird auf dem LCD-Display dargestellt |
| Ablauf | 1. Anwender betätigt den Schalter 2 am LCD- 2. IP-Adresse wird auf dem LCD-Display dargestellt |

|  |  |
| --- | --- |
| **UC 5 - Verbinden** | |
| Beschreibung | Das Raspberry Pi verbindet sich beim Systemstart mit der Hardware zur Ermittlung der Messwerte. |
| Stakeholder | *System (Raspberry Pi)* |
| Vorbedingungen | * Wetterstation eingeschaltet * Hardware zur Ermittlung der Messwerte an das Raspberry Pi angeschlossen und bereit |
| Nachbedingungen | Das System ist mit der Hardware zur Ermittlung der Messwerte verbunden |
| Ablauf | 1. Verbindung mit der Hardware zur Ermittlung der Messwerte aufbauen (IP-Verbindung).    1. Bei einem Kommunikationsfehler soll dieser auf dem LCD Display ausgeben werden.    2. Kann die Verbindung hergestellt werden, kann mit der Ermittlung der Messwerte begonnen werden. |

|  |  |
| --- | --- |
| **UC 6 - Luftdruck ermitteln** | |
| Beschreibung | Der aktuelle Luftdruck wird von der Hardware mittels einem Sensor ermittelt |
| Stakeholder | *System (Raspberry Pi)* |
| Uses | *UC 10 - Messwerte in DB speichern* |
| Vorbedingungen | * Wetterstation eingeschaltet * Verbindung zwischen dem Raspberry Pi und der Hardware hergestellt |
| Nachbedingungen | Messwert wird auf LCD-Display dargestellt oder aber es wird eine entsprechende Fehlermeldung beim Messwert angezeigt. |
| Ablauf | 1. Prüfen ob der Sensor verfügbar ist    1. Sensor nicht verfügbar: Fehlermeldung an LCD-Display ausgeben. Abbruch des Use Cases 2. Ermitteln des aktuellen Messwertes 3. Validieren des ermittelten Messwertes    1. Messwert nicht plausibel: Fehlermeldung an LCD-Display ausgeben. Abbruch des Use Cases 4. Ausgabe des Messwertes auf dem LCD-Display (UC 2) 5. Speicherung des Messwertes in der DB (UC 9) |

|  |  |
| --- | --- |
| **UC 7 - Temperatur ermitteln** | |
| Beschreibung | Der aktuelle Temperatur wird von der Hardware mittels einem Sensor ermittelt |
| Stakeholder | *System (Raspberry Pi)* |
| Uses | *UC 10 - Messwerte in DB speichern* |
| Vorbedingungen | * Wetterstation eingeschaltet * Verbindung zwischen dem Raspberry Pi und der Hardware hergestellt |
| Nachbedingungen | Messwert wird auf LCD-Display dargestellt oder aber es wird eine entsprechende Fehlermeldung beim Messwert angezeigt. |
| Ablauf | 1. Prüfen ob der Sensor verfügbar ist    1. Sensor nicht verfügbar: Fehlermeldung an LCD-Display ausgeben. Abbruch des Use Cases 2. Ermitteln des aktuellen Messwertes 3. Validieren des ermittelten Messwertes    1. Messwert nicht plausibel: Fehlermeldung an LCD-Display ausgeben. Abbruch des Use Cases 4. Ausgabe des Messwertes auf dem LCD-Display (UC 2) 5. Speicherung des Messwertes in der DB (UC 9) |

|  |  |
| --- | --- |
| **UC 8 - Feuchtigkeit ermitteln** | |
| Beschreibung | Die aktuelle Feuchtigkeit wird von der Hardware mittels einem Sensor ermittelt |
| Stakeholder | *System (Raspberry Pi)* |
| Uses | *UC 10 - Messwerte in DB speichern* |
| Vorbedingungen | * Wetterstation eingeschaltet * Verbindung zwischen dem Raspberry Pi und der Hardware hergestellt |
| Nachbedingungen | Messwert wird auf LCD-Display dargestellt oder aber es wird eine entsprechende Fehlermeldung beim Messwert angezeigt. |
| Ablauf | 1. Prüfen ob der Sensor verfügbar ist    1. Sensor nicht verfügbar: Fehlermeldung an LCD-Display ausgeben. Abbruch des Use Cases 2. Ermitteln des aktuellen Messwertes 3. Validieren des ermittelten Messwertes    1. Messwert nicht plausibel: Fehlermeldung an LCD-Display ausgeben. Abbruch des Use Cases 4. Ausgabe des Messwertes auf dem LCD-Display (UC 2) 5. Speicherung des Messwertes in der DB (UC 9) |

|  |  |
| --- | --- |
| **UC 9 - Lichtstärke ermitteln** | |
| Beschreibung | Die aktuelle Lichtstärke wird von der Hardware mittels einem Sensor ermittelt |
| Stakeholder | *System (Raspberry Pi)* |
| Uses | *UC 10 - Messwerte in DB speichern* |
| Vorbedingungen | * Wetterstation eingeschaltet * Verbindung zwischen dem Raspberry Pi und der Hardware hergestellt |
| Nachbedingungen | Messwert wird auf LCD-Display dargestellt oder aber es wird eine entsprechende Fehlermeldung beim Messwert angezeigt. |
| Ablauf | 1. Prüfen ob der Sensor verfügbar ist    1. Sensor nicht verfügbar: Fehlermeldung an LCD-Display ausgeben. Abbruch des Use Cases 2. Ermitteln des aktuellen Messwertes 3. Validieren des ermittelten Messwertes    1. Messwert nicht plausibel: Fehlermeldung an LCD-Display ausgeben. Abbruch des Use Cases 4. Ausgabe des Messwertes auf dem LCD-Display (UC 2) 5. Speicherung des Messwertes in der DB (UC 9) |

|  |  |
| --- | --- |
| **UC 10 - Messwerte in DB speichern** | |
| Beschreibung | Die ermittelten Messwerte in die Datenbank speichern |
| Stakeholder | *System (Raspberry Pi)* |
| Vorbedingungen | * Messwerte wurden von den entsprechenden Sensoren ermittelt * Datenbank auf dem Raspberry Pi verfügbar |
| Nachbedingungen | Messwerte wurden in der Datenbank hinterlegt |
| Ablauf | Messwerte werden in den Datenbank abgespeichert (bei einem allfälligen Zugriffsfehler wird der Fehler nicht nach aussen populiert). |

1. Grobentwurf
   1. Hardware

Bei der Umsetzung der Hardware-Schaltung gibt es 2 mögliche Varianten, wie die elektronischen Bauteile (Sensoren und LCD-Display) mit dem Raspberry Pi (und damit mit der Steuerung) verbunden werden können:

* Direkte Anbindung
* Indirekte Anbindung über einen Master-Baustein
  + 1. Direkte Anbindung

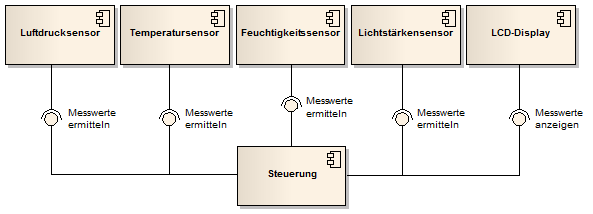


Abbildung: Direkte Anbindung der elektronischen Bauteile an das Raspberry Pi / die Steuerung

Bei der direkten Anbindung werden alle Sensoren sowie der LCD-Display direkt mit dem Raspberry Pi verbunden. Jedes elektronische Bauteil benötigt aus diesem Grunde eine separate Daten- sowie Stromzuleitung.

* + 1. Indirekte Anbindung

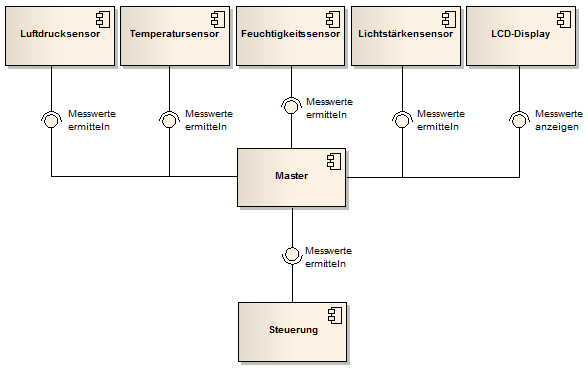


Abbildung: Indirekte Anbindung der elektronischen Bauteile an das Raspberry Pi / die Steuerung

Bei der indirekten Anbindung werden die Sensoren wie das LCD-Display an einen Master angeschlossen, welcher wiederum mit dem Raspberry Pi verbunden ist. Die Stromversorgung erhalten die Bauteile durch den Master. Die Daten werden vom/zum Raspberry Pi über den Master gesandt.

* 1. Steuerung / Online-Schnittstelle

Die Steuerung ermittelt die Messwerte von den Sensoren und gibt diese innerhalb eines bestimmten Intervalls an das LCD-Display sowie an die Datenbank weiter. Die Steuerung wird nach dem Startvorgang des Raspberry Pi automatisch gestartet (kein manueller Eingriff nötig), so dass die Messwerte umgehend ermittelt und gespeichert werden. Damit die Daten von anderen Personen eingesehen werden können, wird eine Online-Schnittstelle definiert, welche die Daten gegen aussen zur Verfügung stellt.

Für die Umsetzung der Steuerung haben wir uns für C++ als Sprache entschieden. Von den auf Raspberry Pi verfügbaren Sprachen ist bei C++ die grösste Erfahrung vorhanden. Bei der Datenbank zur persistenten Speicherung der Messwerte haben wir uns für SQLite entschieden.

Die Online-Schnittstelle kann auf 3 unterschiedliche Arten umgesetzt werden:

* Webseite (PHP)
* RESTful Webservice (PHP)
* Smartphone-App (Windows Phone 8) im Zusammenspiel mit einem RESTful Webservice (PHP)
  + 1. Webseite

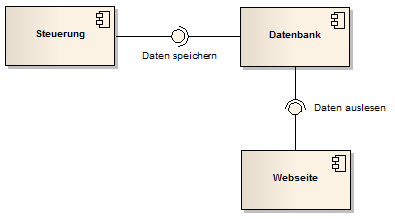


Abbildung: Webseite mit PHP mit Zugriff auf die Datenbank

Die Webseite greift auf die Messdaten, welche von der Steuerung in der Datenbank gespeichert wurden, zu. Die Daten werden anschliessend über ein mit PHP entwickeltes Web-UI zur Verfügung gestellt.

* + 1. Webservice

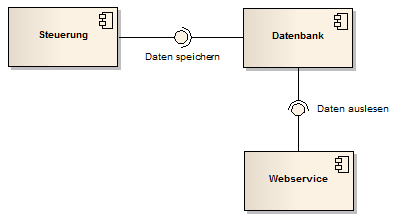


Abbildung: Webservice mit PHP mit Zugriff auf die Datenbank

Der Webservice soll als RESTful Service die Daten über eine Schnittstelle öffentlich zur Verfügung stellen. Die Umsetzung ist mit PHP geplant.

* + 1. Smartphone-App mit Webservice-Zugriff

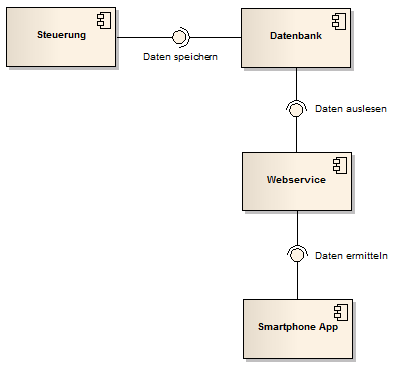


Abbildung: Webservice mit PHP mit Zugriff auf die Datenbank mit einer zusätzlichen Smartphone App (Windows Phone 8)

Bei dieser Variante soll ein RESTful Webservice mit PHP erstellt werden, welcher die Messdaten öffentlich zur Verfügung stellt. Zusätzlich zur Variante nur mit einem Webservice soll zusätzlich eine Smartphone-App (Windows Phone 8) erstellt werden, welche die Daten entsprechend konsumiert.

* 1. Lösungsfindung
     1. Hardware

Beide Varianten lassen sich insbesondere durch einen Unterschied voneinander unterscheiden: Bei der direkten Anbindung führt jeder Sensor sowie der LCD-Bildschirm die Verbindung direkt auf das Raspberry Pi und damit auf die Steuerung. Bei der indirekten Anbindung werden die Verbindungen der elektronischen Bauteile zuerst auf einem Master-Baustein zusammengeführt und erst anschliessend auf die Steuerung gebracht.

Dies führt automatisch zu je einem Vor- wie auch Nachteil der beiden Varianten. So funktioniert bei einem Ausfall eines elektronischen Bauteils bei der direkten Anbindung die Wetterstation immer noch, wenn auch nur eingeschränkt. Allerdings ist die Schnittstelle zwischen den Bauteilen und der Steuerung komplexer, da für jedes Bauteil eine eigene Ansteuerung erstellt werden muss. Bei der indirekten Ansteuerung führt andererseits ein Ausfall des Master-Bausteins dazu, dass die Wetterstation nicht mehr funktioniert, da alle Verbindungen über diesen geführt werden. Hingegen ist die Anbindung an die einzelnen Bauteile einfacher, da nur eine Verbindung unterhalten werden muss.

Erfahrungsgemäss führen mehrere Schnittstellen gegenüber nur einer Schnittstelle in einer Steuerung eher zu mehr Problemen (Threads, Asynchron, Synchronisation). Hingegen kann die Möglichkeit eines Ausfalls eines Master-Bausteins bei sachgemässem Einsatz als sehr gering angenommen werden. **Aus diesem Grunde entscheiden wir uns für die indirekte Ansteuerung der elektronischen Bauteile**.

* + 1. Software

Die Variante mit dem reinen Webservice scheidet aus, da diese Variante nur Benutzern mit Programmierkenntnissen einen Mehrwert bringt (und diese eine eigene Anwendung entwickeln müssten). Die Verbindung zwischen Webservice und Smartphone wäre zwar reizvoll (insbesondere weil diese Variante auf 2 unterschiedlichen Technologien aufsetzen würde), allerdings müsste selbst bei hybriden Apps pro Smartphone-Technologie (Android, iPhone, Windows Phone, Blackberry) eine App erstellt werden, zudem wären die Desktop-Benutzer ausgeschlossen. Aus diesem Grunde **bietet sich die Lösung mit der Webseite mit PHP an**, da diese sowohl von mobilen Benutzern (mit unterschiedlichen Technologien) als auch von Benutzern mit stationären Computern genutzt werden kann.

1. Detailentwurf
   1. Hardware / Schaltung

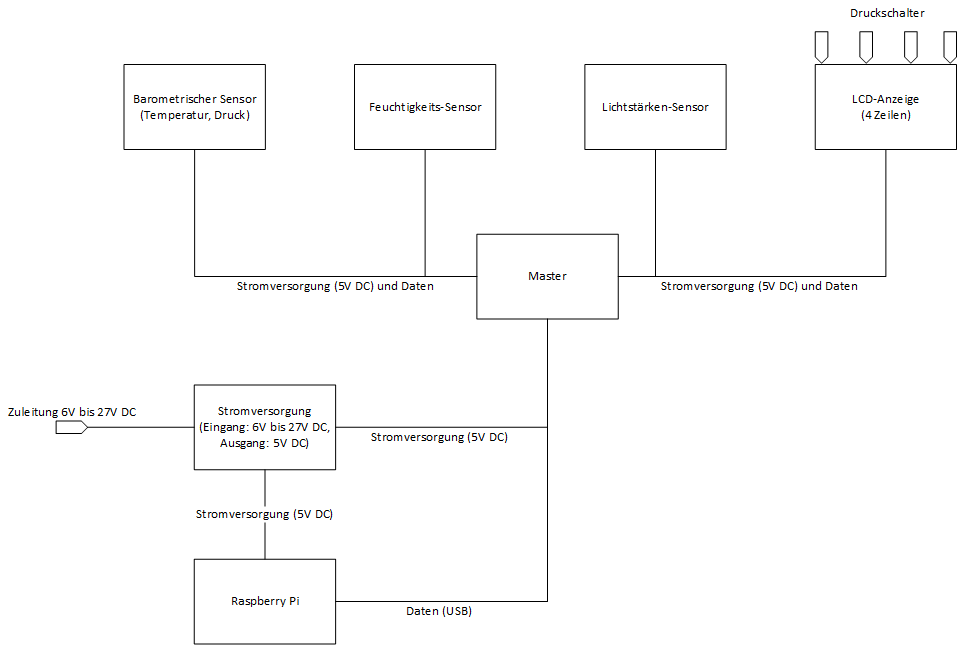


Abbildung: Entwurf der Schaltung der Sensoren, der LCD-Anzeige, sowie des Raspberry Pi

* 1. Steuerung

Analyse

Auf dem Raspberry Pi wird eine Software betrieben, welche zusammenfassend die folgenden Funktionalitäten bietet:

* Nach Einschalten des Gerätes werden die aktuellen Messwerte wie Temperatur und Luftdruck auf dem Display angezeigt.
* Die Messwerte werden in periodischen Abständen in der Datenbank gespeichert.
* Auf Knopfdruck kann der Anwender die IP Adresse anzeigen lassen, resp. es ist möglich, mittels Knopfdruck die Anzeige zwischen den Messwerten und der IP Adresse zu wechseln.

Somit handelt es sich um die folgenden Komponenten, welche miteinander kommunizieren:

* LCD
* Sensoren
* Datenbank
* Schalter

Da die Anwendung nach dem Einschalten des Gerätes nicht mehr gestoppt wird, bis das Gerät ausgeschaltet wird, handelt es sich um einen Prozess, welcher im Hintergrund läuft. Dieser Prozess ermittelt die Messwerte von den Sensoren, speichert diese in der Datenbank und stellt die Werte auf dem LCD dar. Somit benötigt der Prozess während der gesamten Laufzeit Zugriff auf die genannten Komponenten.

Sensoren

Die Sensoren werden periodisch von der Anwendung abgefragt. Zudem soll das Display die Anzeige aktualisieren, sobald sich ein Wert ändert. Es wird daher eine Möglichkeit benötigt, auf Ereignisse bezüglich der gemessenen Werte eines Sensors reagieren zu können. Ein Sensor benötigt daher eine Beobachter Schnittstelle und die Möglichkeit, die aktuellen Messwerte zu liefern.

Datenbank

Eine Sammlung von Messwerten wird schliesslich in der Datenbank abgelegt, zu welcher folglich ein entsprechendes Schema folgende Felder umfasst (ohne Primärschlüssel):

* Temperatur
* Luftdruck
* Lichtstärke
* Feuchtigkeit
* Datum, Uhrzeit

Das Auslesen der Messwerte aus der Datenbank wird durch eine andere Anwendung durchgeführt (Web-Anwendung), daher muss die Abstraktion, welche hier eingesetzt wird, dies vorerst nicht unterstützen.

Die Daten werden in einer SQLite Datenbank abgelegt, auf diese Weise muss kein separater Datenbankserver betrieben werden und eine externe Anwendung (Web Anwendung) kann ebenfalls mittels einem standardisierten Verfahren (SQL) auf die Daten zugreifen.

LCD-Anzeige

Das LCD wird zur Darstellung von verschiedenen Werten verwendet, darin werden sowohl Messwerte mit unterschiedlichen Einheiten (Grad, Pascal oder hPa, Prozent etc.) sowie die IP Adresse und evtl. auch die Zeit dargestellt. Aufgrund des beschränkten Platzes auf dem Display wäre es schwierig, dem Anwender des Displays freizustellen, wie der anzuzeigende Inhalt auf dem Display dargestellt werden soll. Das Gerät muss lediglich die gewünschten Werte auf dem Display darstellen, es reicht daher, wenn die Abstraktion des Displays dies entsprechend ermöglicht. Somit muss das Display eine Möglichkeit bieten, einen Text wie die IP Adresse, eine Temperatur etc. oder alle aktuellen Messwerte zugleich anzuzeigen. Durch die Notwendigkeit, dass das Display die Anzeige aktualisiert, was das Beobachten der Sensoren bedingt, bietet es sich an, einen Model-View Ansatz zu verfolgen. Die Sensoren bilden so das Model, währenddem das Display als View auf Änderungen am Model lauscht.

Schalter

Der oder die Schalter fungieren als Botschafter, welche die Anwendung benachrichtigen, sobald ein Schalter gedrückt wurde. Hierzu bietet sich daher ebenfalls ein klassisches Beobachter Modell an.

Ablauf

Die Anwendung ermittelt in periodischen Zeitabständen die aktuellen Messwerte und speichert diese in der Datenbank. Auf Anfrage über einen Schalter wechselt die Anwendung die Anzeige und stellt den gewünschten Wert dar. Das periodische Speichern der Werte beeinflusst jedoch nicht die Anzeige, für die Anzeige werden stets aktuelle Werte ermittelt. Sobald die Anzeige wechselt, wird der entsprechende Wert ermittelt und angezeigt. Anschliessend werden Änderungen des aktuell dargestellten Wertes registriert und bei jeder Änderung wird die Anzeige aktualisiert.

Der Ablauf ist im Sequenz Diagramm in Abbildung Abbildung ersichtlich. In Abbildung Abbildung ist der Programm Fluss der periodischen Speicherung der Messwerte visualisiert. Dieser Vorgang läuft unter einem separaten Thread kontinuierlich ab.

Abbildung 1: Die Anwendung reagiert auf Änderungen der Messwerte und auf Tastendruck

Abbildung 2: Die Messwerte werden periodisch mittels Polling gespeichert

Synthese

Abstraktionen

Somit ergeben sich die folgenden Abstraktionen mit ihren Eigenschaften und Methoden:

* Sensoren
  + Alle Sensoren bieten die Möglichkeit, den aktuellen Messwert zu ermitteln und einen Beobachter zu benachrichtigen, sobald der aktuelle Wert signifikant geändert hat. Folgende Sensoren werden benötigt:
    - Temperatur-Sensor
    - Feuchtigkeit-Sensor
    - Luftdruck-Sensor
    - Licht-Sensor
* Datenbank
  + Die Datenbank muss eine Menge von Messwerten (Temperatur, Luftdruck etc.) mit dem aktuellen Zeit Stempel speichern.
* LCD
  + Das Display muss die folgenden Elemente darstellen können:
    - Die aktuellen Messwerte
    - Die IP Adresse des Gerätes
    - Eine Fehlermeldung
  + Schalter
    - Der Schalter bietet die Möglichkeit, einen Beobachter zu registrieren, welcher nach dem Betätigen des Schalters benachrichtigt wird.

Struktur

Statt einen monolithischen Ansatz zu verfolgen und dem LCD die Rolle der View zu zuordnen, sollte eher angestrebt werden, einen modularen Aufbau zu erzielen. Da andernfalls das Hinzufügen von weiteren Sensoren und die Implementierung der View schwierig sind. Eine einfache Indirektion löst dieses Problem: Statt dass die View auf alle Sensoren lauscht, existiert zu jedem Sensor eine kleine View, welche nur auf einen einzelnen Sensor lauscht und das LCD als Zeichen-Oberfläche erhält. Auf diese Weise können die einzelnen Sensoren leichter ausgetauscht werden und die Anwendung weist eine schwächere Kopplung auf.

Zusammenfassend ergibt sich die Klassenstruktur wie sie in Abbildung Abbildung dargestellt ist.

Der Grund, weswegen keine Schnittstelle Observer definiert wurde, liegt darin, dass mit C++ gearbeitet wird und statt einer Schnittstelle für den Observer lediglich ein Funktionsobjekt verwendet werden kann. Da der Funktor vorgängig mit Argumenten gebunden werden kann, muss dem Observer beim Aufruf kein spezifisches Argument mitgegeben werden. Generell interessiert den Beobachter lediglich, dass sich der Wert des beobachteten Sensors geändert hat. Eine Schnittstelle des Beobachters, welche beispielsweise den Sensor als Argument erhält, wäre nutzlos, da es sich bei dem Sensor, welcher der Observer erhalten würde, um eine Schnittstelle handeln würde, um den konkreten Sensor zu erhalten, wäre daher ein casting notwendig.

Dies entfällt durch das vorgängige Binden des Funktors, beispielsweise kann eine Instanz einer std::function<void()> mittels einer Closure oder einer gebundenen Member Funktion erzeugt werden.

Aus Platzgründen wurden nicht alle Attribute und Methoden der Klasse Application notiert, generell handelt es sich um eine grobe Sicht auf die Architektur. Eine zentrale Rolle spielt der Konstruktor der Klasse, darin werden die Views bei den Sensoren als Beobachter registriert und die Anzeige mit den aktuellen Werten versehen. Bei einem Wechsel der Anzeige können schliesslich die registrierten Beobachter deregistriert werden um die IP Adresse oder im Fehlerfall die Fehlermeldung anzuzeigen. Eine Alternative zu dem erwähnten Modell, in welchem die Views auf Änderungen am Modell lauschen, besteht darin, dass nicht die Views direkt auf Änderungen am Display lauschen sondern Application. Dies kann realisiert werden, indem wie angedeutet, eine Methode onValueChanged als Grundlage dient, einen Observer zu erstellen. Die Methode kann mit this und der zugehörigen View gebunden werden, um einen Funktor zu erzeugen, welcher als Beobachter eines Sensores dient. Analog dazu muss schliesslich noch dasselbe für den Thread dbWriter getan werden, indem statt bloss den Observer mit dem Sensor zu speichern, zusätzlich noch ein Funktor gespeichert wird, welcher den zugehörigen Wert in Values setzt. Schliesslich kann der Thread die map durchwandern, jeden zugehörigen Funktor mit einem Value aufrufen, den Zeit Stempel setzen und die so assemblierten Werte mittels der Datenbank speichern. Im Falle eines Wechsels der Anzeige werden jeweils die Beobachter bei den Sensoren an- resp. Abgemeldet.

Abbildung 3: Klassen Diagramm

* 1. Datenbank

Die periodisch ermittelten Messwerte werden für spätere Auswertungen und für den Zugriff über die Web Applikation gespeichert. Dass kein zusätzlicher Datenbank Server betrieben werden muss und die Daten dennoch möglichst übergreifend verfügbar sind und zudem eine standardisierte Abfragesprache wie SQL verwendet werden kann, wird SQLite eingesetzt. Für die Messwerte wird bloss eine einzige Tabelle benötigt, worin sämtliche Messwerte mit entsprechendem Zeitstempel abgelegt werden. Die Tabelle Tabelle beinhaltet eine Auflistung der Spalten der Datenbank Tabelle.

|  |  |  |
| --- | --- | --- |
| **Name** | **Typ** | **Bemerkungen** |
| id | INTEGER | Ein stetig wachsender integraler Wert, der die Zeile eindeutig identifiziert. |
| created\_at | TEXT | Ein Zeitstempel in ISO Darstellung ("YYYY-MM-DD HH:MM:SS.SSSZ"), der den Erstellungs Zeitpunkt der Messwerte repräsentiert |
| temperature | REAL | Die Temperatur |
| humidity | REAL | Die Feuchtigkeit |
| pressure | REAL | Der Luftdruck |
| illumination | REAL | Die Lichtstärke |

Tabelle 1: Die Messwerte in der Datenbank

* 1. Webseite

1. Qualitätssicherung

Um den notwendigen Qualität Standard zu ermitteln, werden die funktionalen Anforderungen herbeigezogen um daraus entsprechende Test Szenarien abzuleiten.

* Luftdruck, Temperatur, Feuchtigkeit und Lichtstärke müssen korrekt ermittelt, dargestellt und gespeichert werden.
  + Mittels manuellen Tests und einem Referenz Gerät kann verifiziert werden, ob die Werte in einem gültigen Toleranzbereich liegen. Falls dies über das LCD verifiziert werden kann, ist auch sichergestellt, dass die Werte entsprechend ermittelt werden.
  + Mittels manueller Überprüfung kann verifiziert werden, ob der entsprechende Inhalt in der Datenbank abgelegt wurde. Beispielsweise mittels einer SQLite Shell.
* Sämtliche Informationen müssen über eine Web Applikation eingesehen werden können.
  + Auch dies kann mittels manueller Überprüfung auf dieselbe Weise verifiziert werden: Sämtliche online dargestellten Werte müssen im Vergleich mit den Werten eines Referenz Gerätes in einem gültigen Toleranzbereich liegen. Zudem kann überprüft werden, ob zuvor gespeicherte Daten abrufbar sind.

Zudem müssen weitere Risiken berücksichtigt werden, wobei externe Fehlerquellen wie ein Ausfall der Stromzufuhr ignoriert werden, solche Fehler können nicht adressiert werden. Ein Ausfall des LCD oder ein festsitzender Schalter können ebenfalls nicht behandelt werden. Es sollte jedoch sichergestellt werden, dass keine falschen Werte gespeichert werden, falls ein Sensor ausfällt, sollte dies von der Anwendung erkannt werden.

Zudem ist es möglich, dass irgendwann der Speicher ausgeht. Es wäre auch denkbar, dass eine sehr grosse Datenbank Probleme verursacht.

Es bleiben zusätzlich die folgenden möglichen Fehler, welche behandelt werden müssen:

* Ausfall von Sensoren
  + Im Falle von ausgefallenen Sensoren sollen die entsprechenden Werte nicht gespeichert werden. Auf dem LCD soll in diesem Fall eine entsprechende Fehlermeldung angezeigt werden.
* Kein Speicherplatz mehr verfügbar
  + Falls das Speichern der Messwerte fehlschlägt, weil kein Speicherplatz mehr verfügbar ist, soll auf dem LCD ebenfalls eine entsprechende Fehlermeldung angezeigt werden.
* Kein Netzwerk verfügbar
  + In diesem Fall soll bei einem Wechsel der Anzeige zur IP Adresse eine Fehlermeldung angezeigt werden.

Testing

Um den korrekten Betrieb des Systems zu verifizieren, werden Testfälle definiert, welche im Anschluss an die Implementierung und auch währenddessen durchlaufen werden. Die benötigten Testfälle können anhand der zuvor ermittelten Kriterien und Risiken erstellt werden.

TC1 - Ermitteln der Werte

|  |  |
| --- | --- |
| **TC1 – Ermitteln der Werte** | |
| *Beschreibung* | Das korrekte Ermitteln der Werte der Sensoren wird manuell überprüft. Ein Referenz Gerät liefert die Werte, mit welchen die ermittelten Werte verglichen werden. Die Differenz muss anschliessend im gültigen Toleranzbereich liegen. |
| *Vorbedingungen* | * Ein passendes Referenz Gerät   + Ein beliebiges, handelsübliches Gerät * Definition der Toleranzbereiche   + Feuchtigkeit: 2%   + Temperatur: 0.7°   + Lichtstärke: 0.4lx   + Luftdruck: 1.4hPa |
| *Ablauf* | Die Messwerte werden vom Display des Gerätes abgelesen und mit den Werten des Referenz Gerätes verglichen. Die Differenz der beiden Messdaten wird für jeden gemessenen Werte berechnet. |
| *Auswertung* | Die berechnete Differenz eines jeden Messwertes muss im Toleranzbereich liegen. |

TC2 – Speicherung

|  |  |
| --- | --- |
| **TC2 – Speicherung** | |
| *Beschreibung* | Mittels einer SQLite Shell wird geprüft, ob der periodisch gespeicherte Inhalt in der Datenbank vorhanden ist. |
| *Vorbedingungen* | Die Anwendung befindet sich in laufendem Betrieb und es wurden gemäss der bereits verstrichenen Zeit Datensätze angelegt. |
| *Ablauf* | Nachdem die Anwendung bereits so lange läuft, dass mind. 3 Datensätze angelegt wurden, wird mittels Shell eine Verbindung zu der SQLite Datenbank hergestellt. Es wird geprüft, dass die erwarteten Datensätze vorhanden sind. |
| *Auswertung* | Die erwarteten Datensätze müssen mit den korrekten Zeitstempeln existieren. |

TC3 - Aktuelle Daten über die Web Applikation

|  |  |
| --- | --- |
| **TC3 – Aktuelle Daten über die Web Applikation** | |
| *Beschreibung* | Mittels manuellem Testing wird verifiziert, dass die von der Web Applikation dargestellten Werte im gültigen Toleranzbereich liegen. |
| *Vorbedingungen* | Die Web Applikation ist über den Browser erreichbar. |
| *Ablauf* | Die Web Anwendung wird mit dem Browser geöffnet. Es werden die aktuellen Werte über die Web Oberfläche mit den Werten des Referenz Gerätes verglichen. Zu den Werten des Referenz Gerätes wird zu jeder gemessenen Grösse die Differenz berechnet. |
| *Auswertung* | Die angezeigten Werte müssen im Toleranzbereich von TC1 liegen. |

TC4 - Gespeicherte Daten über die Web Applikation

|  |  |
| --- | --- |
| **TC4 – Gespeicherte Daten über die Web Applikation** | |
| *Beschreibung* | Mittels manuellem Testing wird verifiziert, dass die zuvor gespeicherten Werte auch über die Web Oberfläche verfügbar sind. |
| *Vorbedingungen* | Die Web Applikation ist über den Browser erreichbar. |
| *Ablauf* | Die Web Anwendung wird mit dem Browser geöffnet. Es werden die gespeicherten Werte über die Web Oberfläche abgefragt. Parallel dazu werden die gespeicherten Werte über eine SQLite Shell ermittelt. |
| *Auswertung* | Die angezeigten Werte müssen mit denjenigen in der SQLite Datenbank übereinstimmen. |

TC5 - Ausfall aller Sensoren

|  |  |
| --- | --- |
| **TC5 – Ausfall aller Sensoren** | |
| *Beschreibung* | Während des Betriebs wird das Verbindungskabel zwischen dem Raspberry Pi und der Wetterstation getrennt. Auf der Anzeige der Werte der Sensoren muss eine entsprechende Fehlermeldung angezeigt werden. |
| *Vorbedingungen* | Der Dämon Prozess wurde gestartet und das LCD zeigt die gemessenen Werte an. |
| *Ablauf* | Das Verbindungskabel zum Bricklet wird entfernt. Es wird überprüft, ob auf dem Display eine entsprechende Fehlermeldung angezeigt wird. |
| *Auswertung* | Es wird geprüft, ob eine entsprechende Fehlermeldung angezeigt wird. |

TC6 – Speicherplatz

|  |  |
| --- | --- |
| **TC5 – Ausfall aller Sensoren** | |
| *Beschreibung* | Es wird eine ausreichend grosse Datei auf eine separate, für diesen Test präparierte SD Karte gelegt. Die Datei muss so gross sein, dass bloss noch wenige Bytes an Speicher auf der Karte vorhanden sind. Eine solche Datei kann mittels dd erstellt werden. Anschliessend wird verifiziert, dass die Anwendung versucht, die Datensätze abzulegen. Sobald dies fehlschlägt, muss eine entsprechende Fehlermeldung angezeigt werden. |
| *Vorbedingungen* | Für den Test wird eine übergrosse Datei auf dem Dateisystem abgelegt. Die Datei weist eine Grösse auf, welche bis auf wenige Bytes dem noch verfügbaren Speicher der Karte entspricht. Somit weist die verwendete SD Karte bloss noch wenig Speicherplatz auf. |
| *Ablauf* | Die Anwendung wird gestartet und es wird beobachtet, wie der verfügbare Speicher auf der Karte abnimmt. Sobald kein verfügbarer Speicher mehr vorhanden ist, wird geprüft, ob auf dem LCD eine entsprechende Fehlermeldung angezeigt wird. |
| *Auswertung* | Es wird geprüft, ob eine entsprechende Fehlermeldung angezeigt wird. |