ES-HH - Software Design

Andreas Hasler
andreas.hasler@students.ffhs.ch

David Daniel
david.daniel@students.ffhs.ch

13. November 2014

Dieses Dokument erläutert die Architektur Überlegungen zum Software Design für ras-weather. Die hier diskutierte Architektur bezieht sich auf die Software, welche auf dem Raspberry Pi betrieben wird. Externe Software wie die Web-Applikation oder die Smartphone Applikation werden in diesem Dokument nicht beschrieben.

Inhaltsverzeichnis

1 Analyse

Auf dem Raspberry Pi wird eine Software betrieben, welche zusammenfassend (Hasler & Daniel, 2014) 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.

1.1 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.

1.2 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 [1]

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.

[1] Datum und Uhrzeit der Erstellung der Messwerte

1.3 LCD

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.

1.4 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.

1.5 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 1 ersichtlich. In Abbildung 2 ist der Programm Fluss der periodischen Speicherung der Messwerte visualisiert. Dieser Vorgang läuft unter einem separaten Thread kontinuierlich ab.

2 Synthese

2.1 Abstraktionen

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

- Sensoren

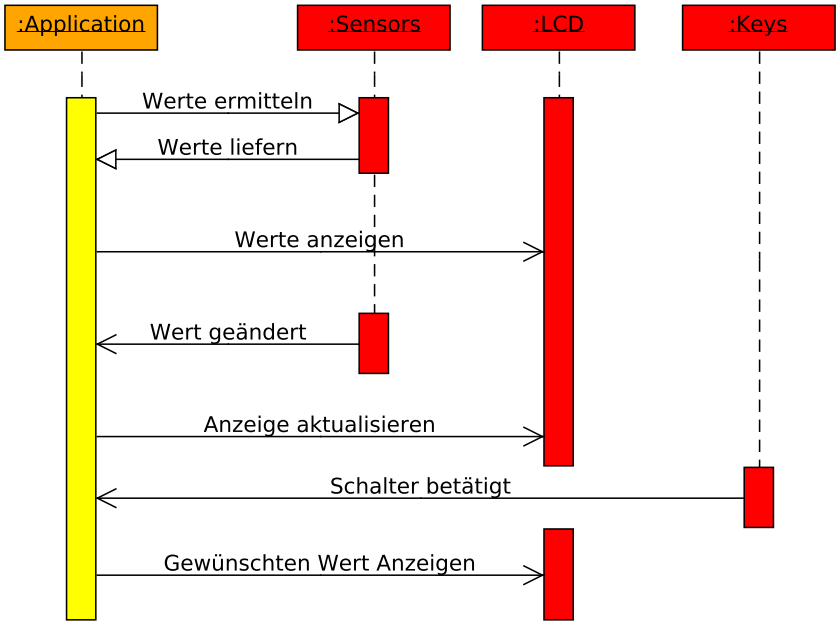

Abbildung 1: Die Anwendung reagiert auf Änderungen der Messwerte und auf Tastendruck

  - Alle Sensoren bieten die Möglichkeit, den aktuellen Messwert zu ermitteln und einen Beobachter zu benachrichtigen, sobald der aktuelle Wert signifikant [2]. 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

[2] Mit signifikant ist gemeint, dass sich der Wert mind. soweit geändert hat, als dass die Änderung auf dem Display erkennbar ist.

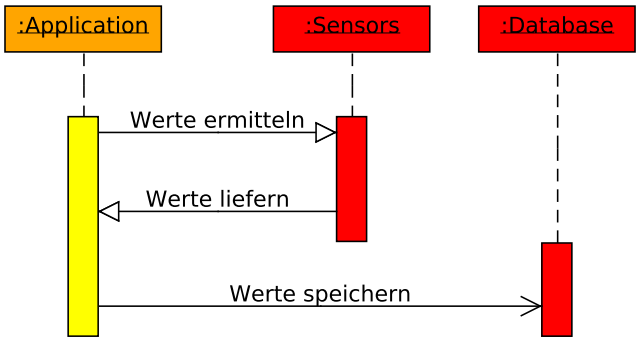

Abbildung 2: Die Messwerte werden periodisch mittels Polling gespeichert

    - 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.

2.2 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 3 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.

3 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.

4 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.

4.1 TC1 - Ermitteln der Werte

Das korrekte Ermitteln der Werte der Sensoren wird wie beschrieben manuell überprüft. Ein Referenz Gerät liefert die Werte, mit welchen die ermittelten Werte verglichen werden. Die Differenz muss anschliessend im gültigen Toleranzbereich liegen.

4.2 TC2 - Speicherung

Mittels einer SQLite Shell wird geprüft, ob der periodisch gespeicherte Inhalt in der Datenbank vorhanden ist.

4.3 TC3 - Aktuelle Daten über die Web Applikation

Mittels manuellem Testing wird verifiziert, dass die von der Web Applikation dargestellten Werte im gültigen Toleranzbereich liegen.

4.4 TC4 - Gespeicherte Daten über die Web Applikation

Mittels manuellem Testing wird verifiziert, dass die zuvor gespeicherten Werte verfügbar sind.

4.5 TC5 - Ausfall aller Sensoren

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.

4.6 TC6 - Speicherplatz

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.

Abbildungsverzeichnis

Literatur

Hasler, A. & Daniel, D. (2014). Projektdokumentation wetterstation.

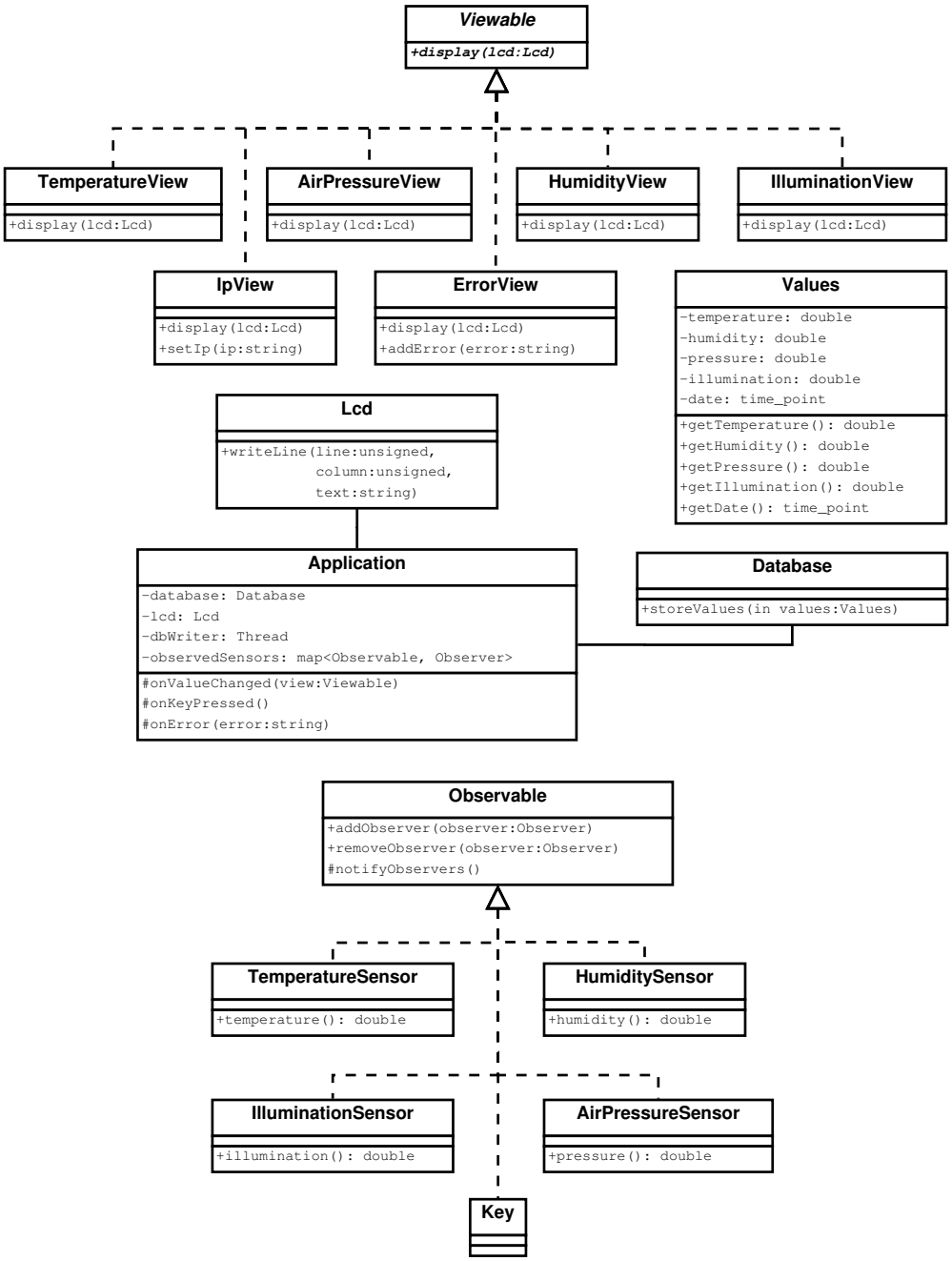

Abbildung 3: Klassen Diagramm