AUTOMATISIERTE LICHTMESSUNG MIT INDOOR-LOKALISATIONSSYSTEM

TECHNISCHER BERICHT

Bachelorarbeit

Autoren
Patrick Scherler
Esteban Luchsinger

Betreuer
Prof. Dr. Farhad Mehta

Industriepartner
Tobias Hofer
HSi Elektronik AG

HOCHSCHULE FÜR TECHNIK RAPPERSWIL
Oberseestrasse 10, 8640 Rapperswil, Switzerland

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

1 Abstract

2 Management Summary

2.1 Ausgangslage

2.2 Vorgehen, Technologien

2.3 Ergebnisse

2.4 Ausblick

3 Rechtliche Hinweise

4 Ausgangslage und Zielsetzung

Dieses Kapitel befasst sich mit der beim Projektstart bestehenden Ausgangslage beim Auftraggeber und der daraus entstandenen Motivation für diese Arbeit.

4.1 Motivation

Die Firma HSi Elektronik AG (Auftraggeber) führt unter anderem Lichtinstallationen bei Kunden im öffentlichen Bereich durch. Um die geforderten Helligkeitswerte der geltenden Standards garantieren zu können, werden entsprechende Lichtmessungen durchgeführt. Dabei setzt der Auftraggeber auf innovative Technologien, wie die Simulation der Lichtverhältnisse vor und nach der Installation. Die effektiven Messungen werden allerdings noch immer manuell mit einem Luxmeter durchgeführt und dokumentiert.

Gemäss den Standards müssten für die Überprüfung eines einzelnen Raumes Messungen in vordefinierten Abständen und Höhen ausgeführt werden. Manuelle Messungen verursachen hohe Zeitaufwände und sind durch die menschliche Ungenauigkeit fehleranfällig. Aus diesem Grund werden in der Praxis meist stichprobenartig Messungen erstellt, um die Daten der Simulation zu bestätigen.

Eine automatisierte und zuverlässige Lösung für die Lichtmessung würde diesen Prozess nicht nur verkürzen, sondern auch die Qualität und Aussagekraft der Messungen verbessern. Dies könnte in Zukunft auch eine regelkonforme Prüfung der geltenden Standards ermöglichen.

4.2 Zweck und Ziel

Ziel dieser Bachelorarbeit ist das Entwickeln einer praxistauglichen Lösung zur lokationsbasierten Ausführung von Lichtmessungen innerhalb eines Raumes. Die Lösung soll weitestgehend automatisiert sein und zwar vor allem dort, wo menschliche Fehler geschehen können. Dem Vermesser, der die Luxmessungen durchführt sollen möglichst alle benötigten Informationen über eine einfache Benutzerschnittstelle angezeigt werden.

Eine detaillierte Beschreibung mit den einzuhaltenden Punkten und einer Abgrenzung ist in der Aufgabenstellung ausgeführt. Nachfolgend sind als Ergänzung dazu alle Abweichungen der Aufgabenstellung aufgelistet, die zu Beginn der Arbeit so definiert wurden:

Anstatt *Beim Erreichen der Punkte im vordefinierten Raster sollen automatisch Lux-Messungen ausgeführt werden.*

Folgendes *Es werden permanent Lux-Messungen ausgeführt und mit den entsprechenden Positionsdaten verknüpft. Die Messwerte können anschliessend im User Interface gefiltert oder kombiniert werden.*

Grund Es sollen so viele Messwerte wie möglich gesammelt und erst anschliessend bei der Analyse im User Interface gefiltert werden.

4.3 Lieferumfang

Der Lieferumfang dieser Arbeit umfasst folgende Punkte:

- System zur lokationsbasierten Ausführung von Lichtmessungen innerhalb eines Raumes. (Siehe Aufgabenstellung)
- Technischer Bericht mit Architekturdokumentation.
- Source Code Repository und Code Dokumentation.

- Zwischen- und Abschlusspräsentation inkl. Live-Demonstration der Applikation.

4.4 Annahmen und Abgrenzungen

Zusätzlich zur Abgrenzung in der Aufgabenstellung wurden zu Beginn der Arbeit mit dem Auftraggeber und dem Betreuer folgende Punkte definiert:

Ansteuerung Die Ansteuerung des Sensors kann notfalls vom Auftraggeber erledigt oder simuliert werden. In dieser Arbeit soll hauptsächlich das Zusammenspiel der verschiedenen Komponenten, die Visualisierung der Sensordaten und die Positionsbestimmung behandelt werden.

Standards Die in der Motivation erwähnten Standards dienen lediglich als Referenz und müssen von der entwickelten Software nicht zwingend eingehalten werden. Eine entsprechende Lösung zur Einhaltung der Standards wäre in Zukunft allerdings denkbar und könnte vom Auftraggeber an seine Kunden als weiteren Service angeboten werden.

4.5 Einschränkungen

Für dieses Projekt gibt es vor allem Einschränkungen im Bereich der Grösse und Verfügbarkeit des Systems.

- Das komplette System soll mobil sein, damit es ein Vermesser jederzeit zum Kunden mitnehmen kann.
- Die Sensorkomponente muss möglichst portabel sein, damit diese später auf ein Hilfsmittel, wie beispielsweise eine Drohne, passt.
- Eine Messung muss komplett offline durchgeführt werden können und nicht auf Services aus dem Internet angewiesen sein.

4.6 Abgrenzung zu anderen bestehenden Lösungen

- Simulationssoftware, die von der HSi eingesetzt wird. (Relux (CH), Dialux (DE))
- Evtl. Abgrenzung zu IoT?

4.7 Stand der Technik

Eine vergleichbare Lösung zur automatisierten Lichtmessung wurde auf dem Markt nicht gefunden. Die Problemstellung scheint zu spezifisch zu sein oder die Anwendungsfälle dafür zu gering. Eine Zerlegung in Einzelprobleme bringt jedoch die nachfolgenden Lösungen hervor.

4.7.1 Lichtmessung

Für Lichtmessungen gibt es bereits diverse Lösungen auf dem Markt. Im professionellen Bereich werden Beleuchtungsstärkemessgeräte (Luxmeter) mit Genauigkeitsklassen und rückführbarer Kalibrierung eingesetzt. Zur Ansteuerung per Software, wie es auch in dieser Arbeit benötigt wird, können Sensoren eingesetzt werden.

Zusätzlich gibt es seit einiger Zeit diverse Luxmeter-Apps, mit denen ein Smartphone als Luxmeter verwendet werden kann. Eine weitere Recherche auf diesem Gebiet scheint allerdings zwecklos, da die Mobiltelefone bereits aufgrund der verbauten Hardware nicht an die Genauigkeit von professionellen Geräten herankommen[1].

[1] https://www.dial.de/de/article/luxmeter-app-vs-messgeraetsind-smartphones-zum-messen-geeignet/

4.7.2 Positionsbestimmung

Für die Positionsbestimmung in geschlossenen Räumen wird Pozyx verwendet. Dieses verspricht laut Hersteller zentimetergenaue Positionsbestimmung per UWB. Die Entscheidung für diese Technologie wurde bereits vor Beginn der Arbeit gefällt.

GPS kommt für die Positionsbestimmung nicht in Frage, da der Empfang in Gebäuden generell stark reduziert bis unmöglich ist.

4.7.3 Kartierung

Die Kartierung umfasst das Verknüpfen der Helligkeitswerte mit den Positionsdaten und der Darstellung im Raum. Dies ist sicherlich der Kern dieser Arbeit.

- Google Maps Indoor
- Pozyx Beta
- elastic.co und GeoHash grid Aggregation (Nachteil: Kosten für Hosting / grosser Ressourcen-Verbrauch)
- HeatmapJS (klein/schlank)

5 Anforderungsanalyse

5.1 Systemkontext

Das Systemkontextdiagramm[2] ist ein guter Ausgangspunkt, um das Softwaresystem als Gesamtbild darzustellen. Das zu entwickelnde System, der Flux-Coordinator[3], befindet sich als Blackbox in der Mitte, umgeben von seinen Benutzern und den Sensoren, mit denen es interagiert.

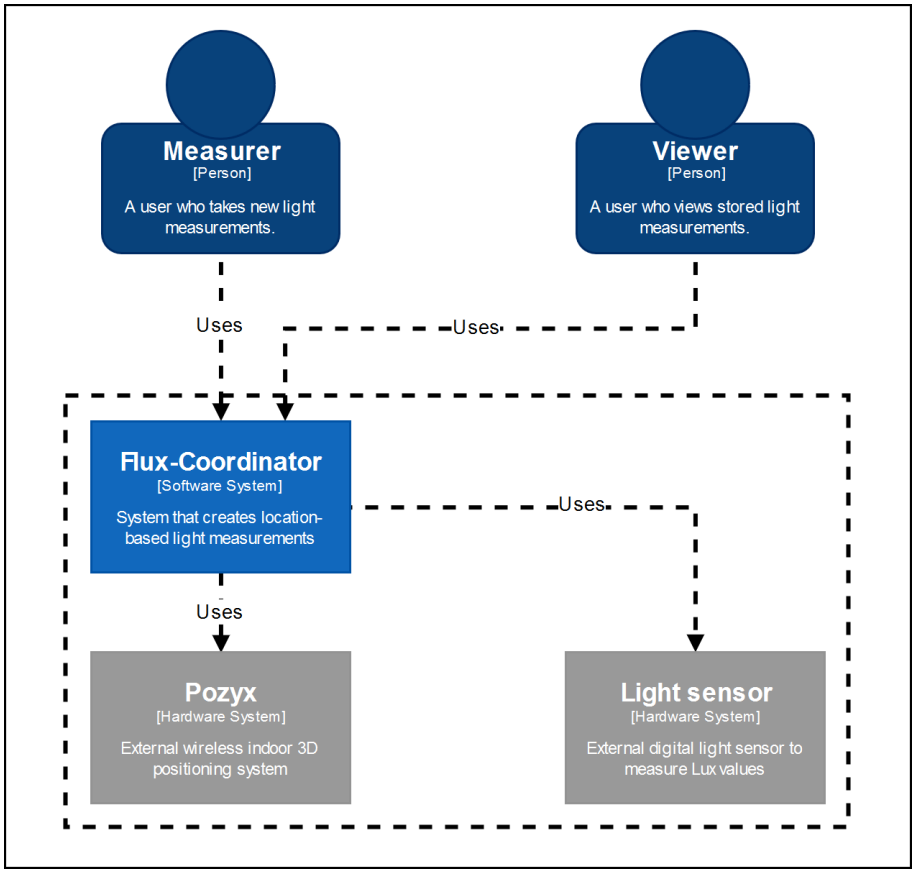


Abbildung 1 Systemkontextdiagramm des Flux-Coordinators

5.2 Rollen

Folgende Rollen wurden während der Anforderungsanalyse identifiziert:

- Benutzer: Greift auf die Anwendung zu, um bereits erstellte Aufzeichnungen anzuzeigen, zu importieren oder zu exportieren.
- Vermesser: Kalibriert die Sensoren, startet die Messungen und überwacht diese während der Durchführung. Der Vermesser ist ein Benutzer mit erweiterten Anforderungen.

5.3 Personas

5.4 Szenario

[2] https://c4model.com/
[3] Der Präfix Flux ist eine Kombination aus Flux (lateinisch Fluss) und Lux (SI-Einheit der Beleuchtungsstärke) und soll den physikalischen Hintergrund der Arbeit symbolisieren. Der Name Flux-Coordinator ist eine Anlehnung an den «Flux Capacitor» aus der Filmreihe «Back to the Future».

5.5 Funktionale Anforderungen

Die Funktionen, die die Anwendung unterstützen soll:

#	Anforderung	Beschreibung	Rolle
1	Messung visualisieren	Als Benutzer möchte ich alle Messwerte einer Messung als Visualisierung anzeigen können, um deren Zusammenhänge zu verstehen.	Benutzer
2	Messwerte filtern	Als Benutzer möchte ich in der Visualisierung einen Filter setzen können, um die Messwerte nach ihren Abständen zu gruppieren.	Benutzer
3	Messung speichern	Als Benutzer möchte ich eine Messung speichern können, um die gesammelten Daten zu einem späteren Zeitpunkt zu betrachten.	Benutzer
4	Messung starten	Als Vermesser möchte ich eine Messung über die Benutzerschnittstelle starten können, damit die Messwerte aufgezeichnet werden.	Vermesser
5	Messung abschliessen	Als Vermesser möchte ich eine laufende Messung über die Benutzerschnittstelle abschliessen können, um die Aufzeichnung weiterer Messwerte zu beenden.	Vermesser
6	Messung fortfahren	Als Vermesser möchte ich eine bereits abgeschlossene Messung über die Benutzerschnittstelle fortfahren können, um sie durch weitere Messwerte zu erweitern.	Vermesser
7	Messung exportieren	Als Benutzer möchte ich im System vorhandene Messungen in einem textbasierten Format exportieren können, um sie extern zu sichern.	Benutzer
8	Messung importieren	Als Benutzer möchte ich eine nicht mehr im System vorhandene Messung importieren können, um sie wieder anzuzeigen.	Benutzer
9	Positionierungssystem konfigurieren	Als Vermesser möchte ich das Positionierungssystem über die Benutzerschnittstelle konfigurieren können, um die Messung schneller aufzusetzen.	Vermesser
10	Lichtsensor kalibrieren	Als Vermesser möchte ich den Lichtsensor über die Benutzerschnittstelle kalibrieren können, um ungenaue Messungen zu vermeiden und Korrekturen vorzunehmen.	Vermesser
11	Anmeldung durchführen	Als Benutzer möchte ich mich anmelden können, um auf die Daten meiner Installationen zugreifen zu können.	Benutzer
12	Abmeldung durchführen	Als Benutzer möchte ich mich nach der Anmeldung abmelden können, um die Daten meiner Installationen für unbefugte unzugänglich zu machen.	Benutzer
13	Instrumente testen	Als Vermesser möchte ich die Konfiguration und Kalibrierung der Sensoren vor Beginn der Messung testen können, um fehlerhafte Messungen zu vermeiden.	Vermesser

Tabelle 1 Funktionale Anforderungen des Flux-Coordinators

Das Formulieren der CRUD-Methoden (Create, Read, Update, Delete) wurde der Einfachheit halber weggelassen. Diese werden bei Bedarf implementiert.

Die Priorisierung der Anforderungen entspricht der Reihenfolge in der Tabelle. Es gibt keine optionalen Anforderungen, sondern lediglich die Priorisierung und eine zeitlich begrenzte Entwicklungszeit.

5.6 Nicht funktionale Anforderungen

Die folgenden nichtfunktionalen Anforderungen sind nach ISO 9126 [1] gruppiert.

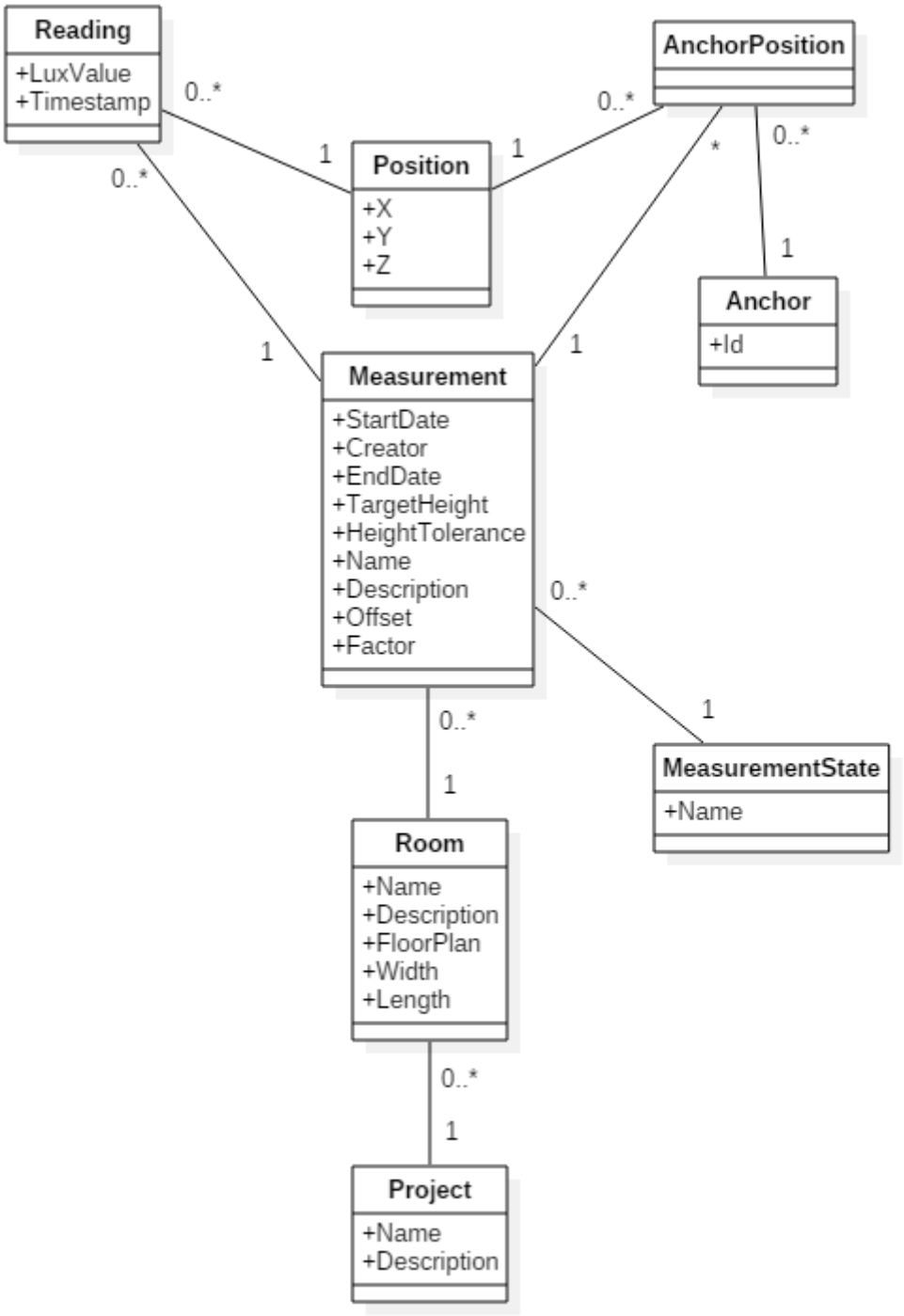
#	Anforderung	Kategorie
1	Sensoren lassen sich austauschen, ohne dafür die Serverkomponente anpassen zu müssen.	Maintainability
2	Ein Benutzer muss auf das User Interface zugreifen können, ohne zusätzliche Software auf seinem Client Gerät installieren zu müssen.	Usability
3	Die Messwerte müssen bei einer Round-Trip-Time von unter 25 ms in mindestens 90% der Anfragen in weniger als 1 Sekunde angezeigt werden.	Efficiency
4	Die Server-Komponente muss auf einer von den Sensoren getrennten Plattform ausgeführt werden können.	Portability
5	Das User Interface soll auf den gängigen Plattformen (PC, Laptop, Tablet) verwendet werden können.	Portability
6	Die Funktionalität des Systems muss vor nicht-authentifizierten Zugriffen geschützt sein.	Security
7	Die aufgezeichneten Messwerte müssen unverändert persistiert werden. Eventuelle Faktoren oder Filter müssen zusätzlich zu den Rohdaten abgelegt werden.	Functionality

Tabelle 2 Nichtfunktionale Anforderungen des Flux-Coordinators

6 Analyse

6.1 Domainanalyse

Das nachfolgende Domain-Modell zeigt die wesentlichen konzeptionellen Klassen und ihre Zusammenhänge. Es soll einen Überblick über die Problemdomäne schaffen.



Nachfolgend sind einige Klassen und Beziehungen genauer beschrieben:

Auf den ersten Blick ist das Attribut *TargetHeight* der Klasse *Measurement* identisch mit dem *Z*-Attribut der Klasse *Position*. Die *TargetHeight* muss jedoch zu Beginn einer Messung definiert werden und beschreibt somit die Soll-Höhe der auszuführenden Messung. Der *Z*-Wert beschreibt hingegen den effektiv von der Positionierungssoftware gemessenen Wert.

Das Attribut *HeightTolerance* definiert die Toleranz für Abweichungen von der gewünschten Messhöhe (*TargetHeight*). Dieser Wert muss bei unterschiedlichen Traversierungs-Hilfsmitteln gegebenenfalls angepasst werden, da nicht immer dieselbe Präzision erreicht werden kann. Eine zu kleine Toleranz bedeutet das Wegwerfen vieler Messwerte und würde die Kartierung des Raumes verlangsamen. Eine zu grosse Toleranz ergibt Messwerte, die sich in der aufgenommenen Höhe stark unterscheiden. Dies könnte die Aussagekraft der Messung verschlechtern.

Die Beziehung zwischen *Measurement* und *AnchorPosition* definiert für eine Messung beliebig viele Antennen des Positionierungssystems. Laut dem Hersteller des Positionierungssystems Pozyx sind mit entsprechender Programmierung unbegrenzt viele Antennen möglich[4].

6.1.1 Ubiquitous Language

Nachfolgend sind die wichtigsten Begriffe der Problemdomäne für den Kontext dieser Arbeit definiert. Dies ist ein bewährtes Vorgehen im Domain-Driven Design (DDD) [1].

Besonders hervorzuheben sind die Begriffe *Measurement* und *Reading*, da sie schnell verwechselt werden und doch den Kern dieser Arbeit bilden.

Begriff	**Erklärung**
Project	Logische Gruppierung von Messungen (Measurement) und deren Räumen (Room) die zu einem gemeinsamen Auftrag gehören
Room	Zu vermessender Raum oder Bereich eines Raumes innerhalb eines Gebäudes
Measurement	Einzelne Messdurchführung innerhalb eines Raumes (Room) mit beliebig vielen Messwerten (Readings)
Reading	Einzelner Messwert als Kombination aus Helligkeitswert in Lux und relativer Position innerhalb des Raumes als kartesische Koordinaten
Anchor	Referenzpunkt des Positionierungssystems innerhalb des Raumes
Kartierung	Das Vermessen eines Raumes heisst Kartierung.
Floor Plan	Der Grundriss des zu vermessenden Raumes.

Tabelle 3 Definition der Ubiquitous Language

6.2 Entwicklungsumgebung

6.3 Lichtmessungen durchführen

6.4 Human Error (Menschliche Fehler)

https://en.wikipedia.org/wiki/Human_error

6.5 Positionserkennung

6.6 Integration der Sensordaten

6.7 User Experience

[4] https://www.pozyx.io/Documentation/where_to_place_the_anchors (Kommentar von Pozyx Labs)

7 Realisierung

Die meisten Architekturdiagramme sind dem C4 Model von Simon Brown [3] nachempfunden. Dieses soll es Softwareentwicklern vereinfachen, die Funktionsweise eines Softwaresystems zu beschreiben und minimiert die Lücke zwischen der Software Architektur und dem Programmcode.

7.1 Architektur

Das System soll möglichst so aufgebaut werden, dass der Auftraggeber es in allen möglichen Situationen verwenden kann. Aus diesem Grund wurde das System als verteiltes Softwaresystem konzipiert. Durch die Aufteilung in verschiedene Subsysteme lassen sich diese auch problemlos parallel entwickeln.

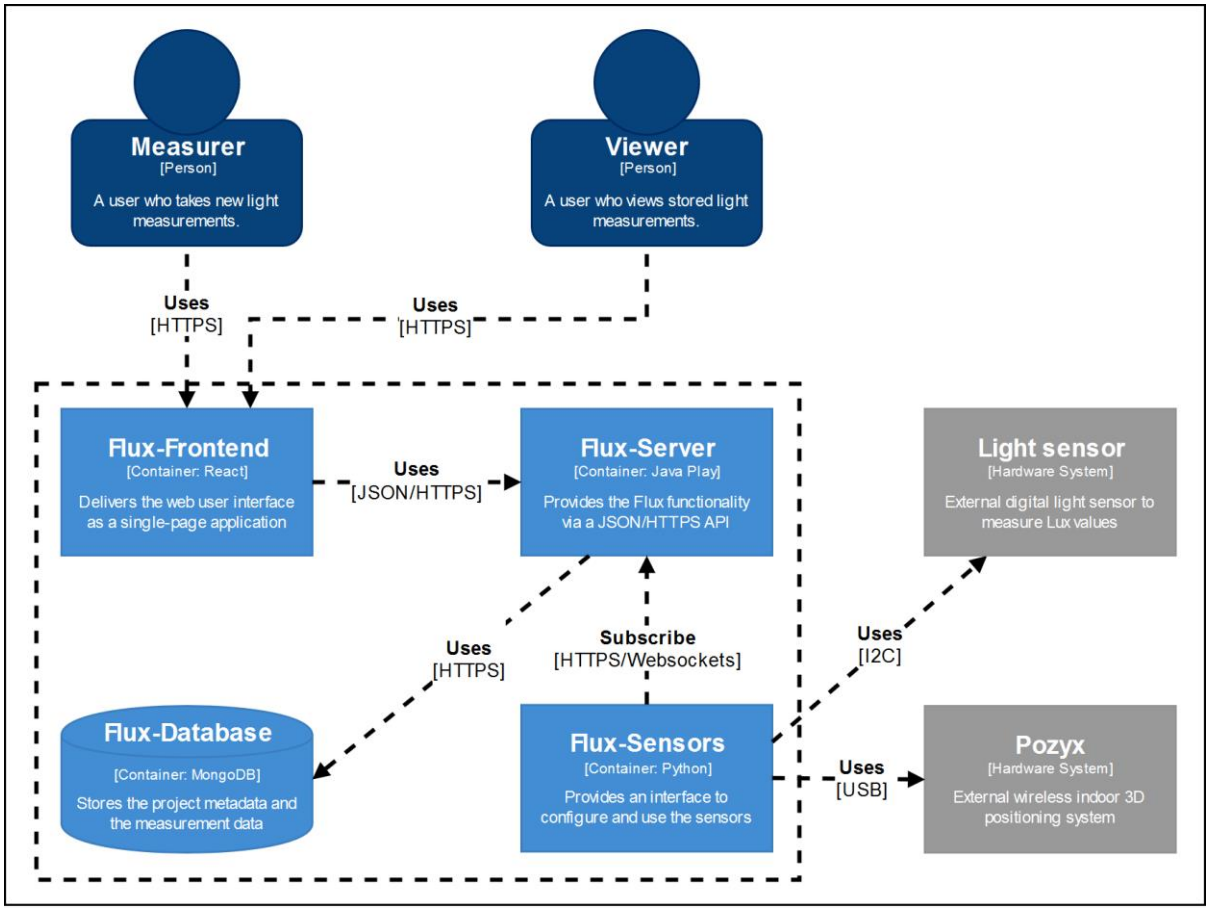


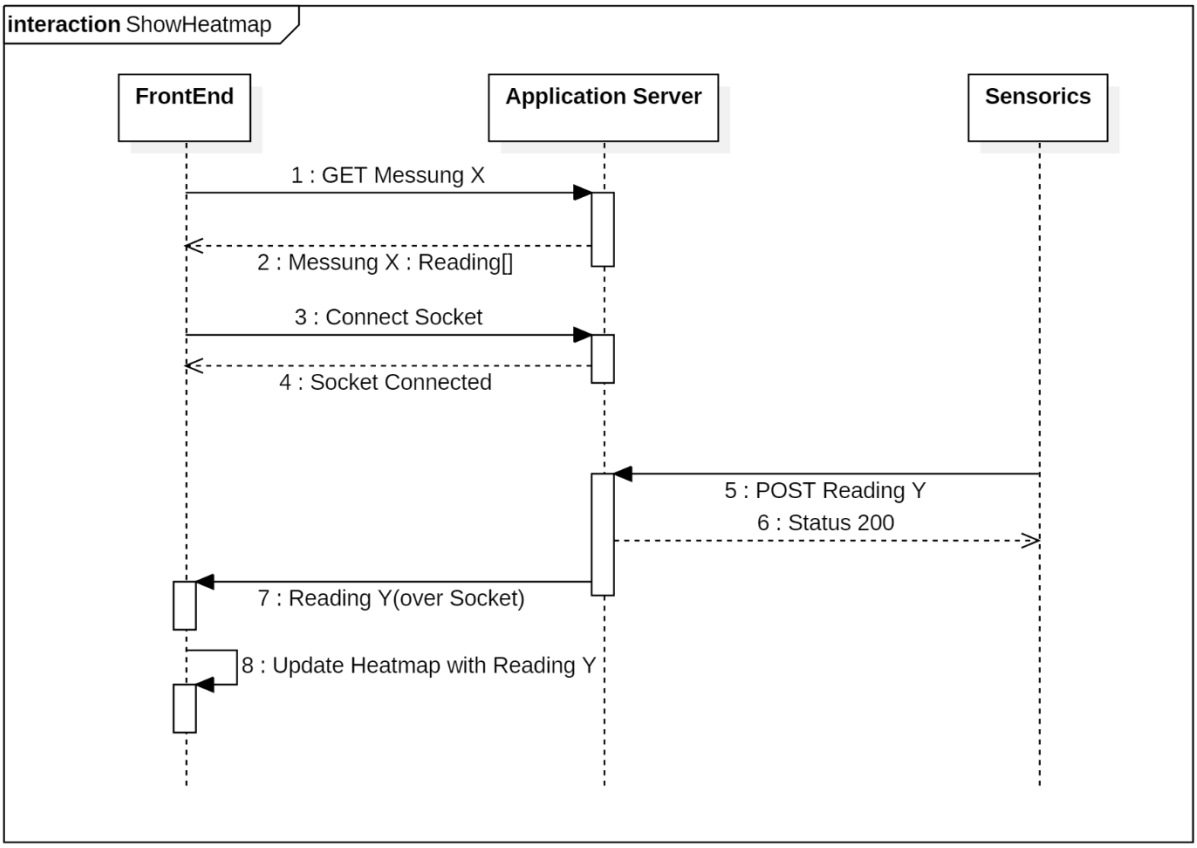
Abbildung 2 Container Diagramm des Flux-Coordinators

7.1.1 Frontend

Das Frontend ist für das Rendern der von der API gelieferten Daten zuständig. Dafür wird die Frontend Library ReactJS verwendet. Durch das Lokale Rendern auf dem Client, lässt sich die Reaktionszeit des UIs vermindern, was zu einer besseren Nutzererfahrung führt.

Anzeigen der Heatmap

Die Heatmap lädt am Anfang die schon auf der Datenbank verfügbaren Messungen von der Schnittstelle. Anschliessend öffnet der Client eine Socketverbindung zum Application Server. Diese Socketverbindung wird gebraucht, damit die neuen Messungen der Sensoren direkt und fast in Echtzeit auf das Frontend übertragen werden. Der Socket wird wieder geschlossen, sobald der Benutzer die Kartenansicht verlässt.



7.1.2 Application Server

Als Knotenpunkt für alle verteilten Komponenten dient eine Applikationsschnittstelle (API). Die Schnittstelle wird auf einem Application Server gehostet. Die Schnittstelle wird basierend auf dem REST Prinzip aufgebaut, sodass die Kommunikation zu und vom Application Server über http Anfragen stattfindet. Als Datenformat soll das JSON Format dienen, weil dieser sich mittlerweile in der Webentwicklung durchzusetzen scheint. XML wäre für dieses Projekt unnötig kompliziert, da die erweiterte Funktionalität, wie zum Beispiel XLink Attribute für diese Arbeit keine Vorteile bringen. Die einfache Objektserialisierung von JavaScript in JSON und die Verwendung einer dokumentbasierten Datenbank, welche die Daten im BSON Format speichert [2] (und in JSON ausgibt) ergeben weitere Vorteile. Das Entwicklerteam hat wenig Erfahrung mit YAML und die Vorteile von YAML sind nicht so offensichtlich, wie bei JSON.

7.1.3 Sensoren

Die Sensoren werden von einem Raspberry Pi gesteuert. Dieser Client übermittelt die Daten an die Schnittstelle über http POST Requests, sobald sie bereitstehen. Wichtig ist, dass die Sensordaten des Luxmeters und des Positionierungssystems lokal auf dem Client zusammen verschmolzen und mit einem Timestamp versehen werden.

Hinweis: Der Grund, weshalb POST und nicht PUT Anfragen verwendet werden ist, dass PUT Anfragen idempotent sein sollen. Dabei ist es «good practice», wenn der Client dem Server bei einer PUT Anfrage auch den Namen gibt, unter welchem die Ressource zugreifbar sein soll. Im Fall der Messwerte, behält aber ultimativ der Application Server die Hoheit über die Datenhaltung.

7.2 Technologieentscheidungen

7.2.1 Datenbanktechnologie

Die Datenbank speichert die ermittelten Messwerte und enthält auch die Metadaten über Projekte, Räume und Messdurchführungen. Aufgrund des spezifischen Datenmodells und der im Projekt evaluierten Anforderungen wurde der Einsatz eines nicht-relationalen DBMS diskutiert. Der Entscheid fiel schliesslich auf ein dokumentbasiertes DBMS. Als Implementierung wurde aufgrund der grossen Verbreitung und der aktiven Community MongoDB gewählt. Die nachfolgenden Gründe waren für die Entscheidung relevant.

Einfachere Bereitstellung

Eine Anforderung an das Projekt ist eine einfache Bereitstellungsstrategie. Relationale DBMS (RDBMS) sind erfahrungsgemäss schwieriger bereitzustellen, als dokumentbasierte DBMS. Der Application Server bietet eine Schnittstelle im JSON Format an, weshalb sich eine Datenhaltung in JSON zusätzlich lohnt.

Flexibles Schema

Die Austauschbarkeit der Sensoren zählt zu den nichtfunktionalen Anforderungen und Ideen für diverse mögliche Erweiterungen wurden bereits diskutiert. Das flexible Schema einer NoSQL Datenbank würde sich dafür optimal eignen.

Synergien durch verwendete Technologien

Dadurch, dass JSON sowohl für die Datenhaltung, wie auch beim Application Server und im Frontend verwendet wird, erübrigt sich ein objektrelationales Mapping. Dies ist eine zusätzliche Entlastung für die Wartung des Projektes und steigert die Effizienz der Entwicklung.

7.3 Datenbankschema

Zwar entfällt bei einer dokumentbasierten Datenbank das Erstellen eines Schemas im traditionellen Sinn, doch die optimale Aufteilung der Daten spielt dennoch eine grosse Rolle. In MongoDB werden die Daten in Datenbank, Collections und Documents aufgeteilt. Diese drei Kategorien lassen sich grob in Datenbank, Tabellen und Zeilen für RDBMS übersetzen. Für dokumentbasierte DBMS muss jedoch gegenüber traditionellen RDBMS ein Umdenken stattfinden. Zum Beispiel sind in dokumentbasierten DBMS denormalisierte Daten öfter erwünscht, als in RDBMS.

7.3.1 References vs. Embedded Documents

Ein Document in MongoDB kann eingebettete Dokumente mit verwandten Daten enthalten. Generell ist es besser, Dokumente in andere Dokumente einzubetten, anstatt eine Referenz zu einem anderen Dokument (möglicherweise in einer anderen Collection) zu speichern. Es gibt jedoch Einschränkungen bei eingebetteten Dokumenten zu beachten. So ist die Dokumentgrösse auf 16 MB begrenzt, was bei mehreren Messungen mit je einer grossen Anzahl Messwerte längerfristig ein Problem darstellen könnte.

Referenzen bieten jedoch eine grössere Anpassbarkeit, als eingebettete Dokumente. Sie helfen auch, Dokumente klein zu halten und so die Performance von Abfragen zu verbessern.

Das Mapping der Datenbank mit den Workflowobjekten sieht wie folgt aus:

Database Die Datenbank beinhaltet die gesamten Daten des Systems.

Collection Eine Collection (analog zu Tabelle in RDBMS) teilt die verschiedenen Dokumente in Kategorien auf. Bei dokumentbasierten Datenbanken muss eine Balance zwischen dem Hinzufügen aller Werte in einem einzelnen Document oder dem Aufteilen der Werte in verschiedene Collections. Ein Document wird nämlich immer vollständig herausgegeben. Ein «herauspicken» von Daten innerhalb eines Documents gestaltet sich als schwierig.

Es gibt folgende Collections: projects und readings.

Document Ein Document beinhaltet einen konkreten Wert (analog zu Zeilen in RDBMS). Ein Beispiel für ein Dokument ist ein konkretes Projekt oder eine Messung.

7.3.2 Vorgehen bei der Modellierung

Bei der Modellierung einer dokumentbasierten Datenbank stellen sich gegenüber klassischen relationalen Datenbanken grundsätzlich drei Fragen:

- Soll ein Objekt referenziert oder eingebettet werden?
- Wie soll die Referenz implementiert werden? (Falls das Objekt nicht eingebettet wird)
- Sollen gewisse Daten denormalisiert werden? (Diese Frage wird z.T. auch in relationalen Datenbanken gestellt.)

Die Frage nach Referenz oder Einbettung darf nicht verwechselt werden mit der Frage nach Attribut oder eigener Tabelle in relationalen Datenbanken. Dort wird nämlich eher die Komplexität der Daten untersucht, wobei die Überlegung in dokumentbasierten Datenbanken eine andere ist. Folgende Überlegungen sind für die obigen drei Fragen entscheidend:

- Use Cases und häufigste Queries: Was wird häufig zusammen abgefragt?
- Kardinalität der Beziehungen
- Volatilität der Daten

Use Cases

Für die Visualisierung einer Messung müssen stets alle zugehörigen Messwerte geladen werden. Für die Darstellung der Navigation reichen jeweils der Name und die Beschreibung der einzelnen Projekte, Räume und Messungen aus.

Kardinalität

Im Gegensatz zu relationalen Datenbanken zählt hier nicht nur die Unterscheidung zwischen 1:1, 1:n oder n:m, sondern sind vielmehr auch konkrete Zahlen von Bedeutung.

Ein Projekt hat in den meisten Fällen wahrscheinlich nicht mehr als eine Hand voll Räume mit je ebenso vielen Messungen. Eine Messung hingegen kann aus tausenden Messwerten bestehen.

Volatilität

Mit Volatilität ist die Langlebigkeit bzw. Flüchtigkeit der gespeicherten Daten gemeint. Optimalerweise sollten im selben Dokument vorwiegend Daten mit ähnlicher Volatilität beherbergt werden.

Eine abgeschlossene Messung mit ihren Messwerten wird im Nachhinein nicht mehr verändert. Name und Beschreibung von Projekt, Raum und Messung können sich ab und zu ändern. Sich häufig ändernde Daten sind in dieser Problemdomäne keine vorhanden.

Dies sind optimale Voraussetzungen für eine Denormalisierung. Das macht erst Sinn bei einem Verhältnis von vielen Lesezugriffen auf wenige Schreibzugriffe, denn der Nachteil einer Denormalisierung liegt in der erschwerten Aufrechterhaltung der Konsistenzbedingungen.

7.3.3 Ergebnis der Modellierung

Als Ergebnis wird ein einzelnes Dokument für sämtliche Projekte und Räume mit den Metadaten der Messungen erstellt. Zusätzlich wird pro durchgeführte Messung für alle Messwerte ein weiteres Dokument erstellt.

Für eine einfachere Verständlichkeit sind die aufgeführten Schemas mit Beispieldaten ausgefüllt. Eine ausführliche Dokumentation als JSON-Schema[5] ist im Anhang abgelegt.

[5] http://json-schema.org/

```
{
  "projects":[
    {
      "projectId":0,
      "name":"HSR 2018",
      "description":"Messungen an der HSR im 2018",
      "rooms":[
        {
          "roomId":0,
          "name":"Zimmer 1.262",
          "description":"SA/BA Labor",
          "floorPlan":"91837645918723641982466345 6.png",
          "width":17300,
          "length":5600,
          "floorSpace":76.25,
          "measurements":[
            {
              "measurementId":0,
              "name":"Messung 1",
              "description":"Erste Probemessung",
              "creator":"Paul Schmidt",
              "startDate":"2018-03-23 13:01:23",
              "endDate":"2018-03-23 13:17:56",
              "state":"done",
              "targetHeight":800,
              "heightTolerance":50,
              "offset":125,
              "factor":1.5,
              "anchorPositions":[
                {
                  "anchorId":0,
                  "name":489,
                  "xPosition":0,
                  "yPosition":0,
                  "zPosition":2000
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```


Abbildung 3 Datenbankschema der Metadaten

```
{
  "measurementId":0,
  "readings":[
  {
    "readingId":0,
    "luxValue":489,
    "xPosition":14849,
    "yPosition":2316,
    "zPosition":822,
    "timestamp":"2018-03-23 13:01:36"
  }
  ]
}
```


Abbildung 4 Datenbankschema einer Messdurchführung

7.4 Deployment

Bei der bisherigen Planung des Systems wurde stets auf eine tiefe Kopplung der einzelnen Container wertgelegt, um den Einschränkungen und Anforderungen des Projekts gerecht zu werden. Nachfolgend sind zwei Deployment-Szenarien abgebildet, in denen der Flux-Coordinator am ehesten zum Einsatz kommen wird. Durch die Unabhängigkeit der einzelnen Container sind jedoch diverse Variationen möglich. So lässt sich das System in Zukunft optimal erweitern.

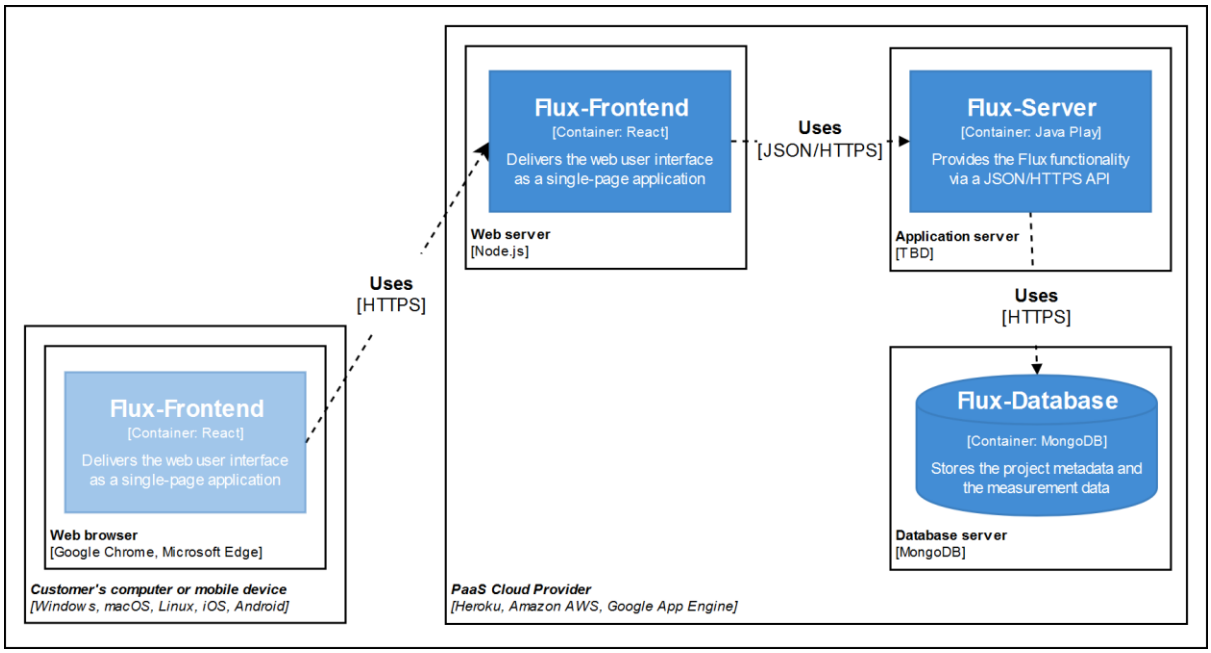


Abbildung 5 Deployment Diagramm mit Paas Cloud Provider

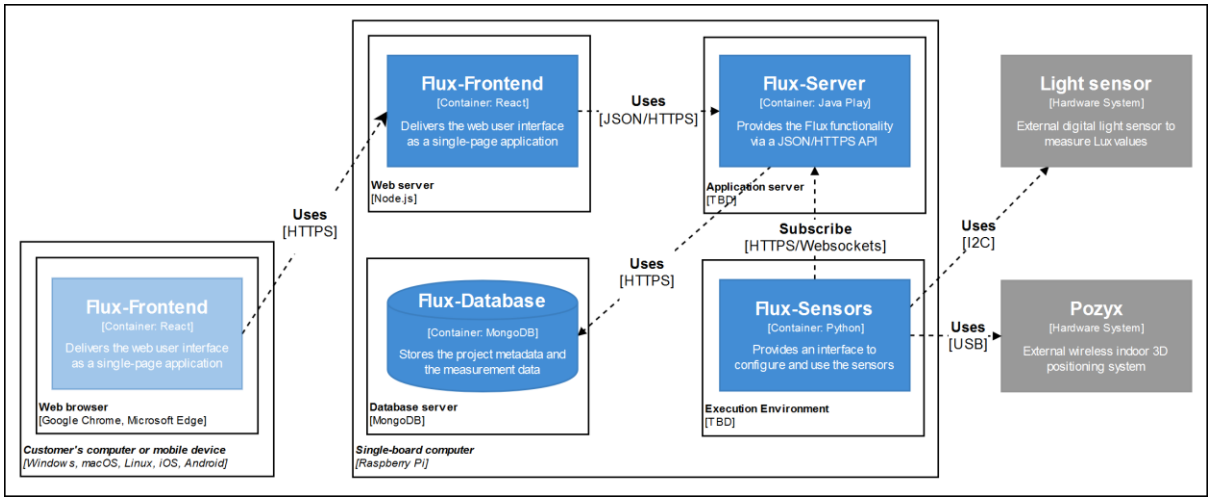


Abbildung 6 Deployment Diagramm mit Single-board Computer

7.5 Probleme & Lösungsansätze

7.5.1 Verfügbarkeit der verteilten Systeme

Das auszuliefernde System besteht aus mehreren voneinander gekapselten Komponenten. Einige der Komponenten sind abhängig von der Funktionsfähigkeit der anderen Komponenten. Um effizient entwickeln zu können, muss zum Beispiel der Application Server verfügbar sein. Wenn der Application Server nicht verfügbar ist, können die Sensoren keine Daten an das Frontend übermitteln. Der Application Server seinerseits ist von der Datenbank abhängig, sonst kann er keine Daten abspeichern.

Um diese Abhängigkeiten möglichst einfach zu erfüllen, wurde entschieden, auf Continuous Delivery zu setzen. Nach dem Durchlaufen und Bestehen der Tests einer Komponente, wird diese automatisch auf einem Heroku Dyno bereitgestellt und ist somit für die anderen Komponenten über das Internet verfügbar. Durch diese ständige Verfügbarkeit kann sich der Entwickler stets darauf verlassen, dass die für die Entwicklung benötigten Komponenten erreichbar sind und muss nicht manuell eigene Instanzen deployen. Somit kann sich der Entwickler besser auf die eigentliche Arbeit konzentrieren.

7.5.2 Async IO für Datenbank

Das System hat nur begrenzte Resourcen zur Verfügung. Wenn Blocking – I/O bei den Datenbank Abfragen verwendet würden, würde wertvolle Rechenzeit verschwendet. Deshalb wird ein asynchroner MongoDB Treiber eingesetzt: «MongoDB RxJava Driver», welcher Reactive Extensions unterstützung hinzufügt, um einfach und zuverlässig asynchrone I/Os auf die Datenbank unterstützen zu können.

7.5.3 Ergebnis

8 Schlussfolgerung und Ausblick

8.1 Beurteilung der Ergebnisse

8.2 Mehr Titel...

9 Literaturverzeichnis

[1] International Organization for Standardization, «Software engineering -- Product quality -- Part 1: Quality model,» 2001.

[2] E. Evans, Domain-Driven Design. Tackling Complexity in the Heart of Software, Addison-Wesley, 2003.

[3] MongoDB Inc, "JSON and BSON," MongoDB Inc, [Online]. Available: https://www.mongodb.com/json-and-bson. [Accessed 11 03 2018].

Anhang

A. Beispielanhang

Json-Schema, Swagger API, Mockups, usw.

B. Glossar

C. Installation Raspberry Pi

Für die Installation des Raspberry Pi sind nachfolgend eine manuelle und eine automatisierte Variante beschrieben. Um Fehler zu vermeiden wird die automatisierte Installation empfohlen. Die manuelle Installation kann als Checkliste zur Fehlersuche oder bei der Verwendung einer anderen Hardware verwendet werden. Die Resultate der beiden Vorgehensweisen sind identisch und wurden mit folgenden Raspberry Pi Modellen getestet:

- Raspberry Pi Zero W[6]

Manuelle Installation

Als erstes muss das Betriebssystem des Raspberry Pi heruntergeladen werden. Die Installation wurde getestet mit dem aktuell empfohlenen Raspbian Stretch in der Version Lite.

Das Image kann unter folgender Quelle bezogen werden: https://www.raspberrypi.org/downloads/raspbian/

Anschliessend muss das Image auf die SD-Karte geladen werden. Dieser Vorgang ist abhängig vom lokal installierten Betriebssystem. Hierfür kann die offizielle Anleitung befolgt werden: https://www.raspberrypi.org/documentation/installation/installing-images/.

Bevor die SD-Karte in den Raspberry Pi eingefügt wird, müssen noch zwei Dateien hinzugefügt werden, um das System im «Headless Mode» ohne Monitor und Tastatur betreiben zu können.

Folgende Dateien müssen ins Root-Verzeichnis der SD-Karte hinzugefügt werden:

- Leere Datei mit dem Namen ***ssh***
- Konfigurationsdatei mit dem Namen ***wpa_supplicant.conf*** und folgendem Inhalt:

```
1  country=CH
2  ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
3  update_config=1
4
5  network={
6      ssid="flux-network"
7      scan_ssid=1
8      psk="fcwFCKLKVa5C7j6R3Jrt"
9      key_mgmt=WPA-PSK
10 }
```

Hinweis: Dateiname, -endung und Inhalt müssen genau der Beschreibung oben übereinstimmen, damit sie vom Raspberry Pi korrekt geladen und verarbeitet werden. Die Datei *ssh* hat keine Dateiendung.

Nun kann die SD-Karte ausgeworfen und in den Raspberry Pi eingefügt werden. Nach Abschluss des Boot-Vorgangs kann eine SSH-Verbindung aufgebaut werden:

```
$ ssh pi@192.168.1.147
```

[6] https://www.raspberrypi.org/products/raspberry-pi-zero-w/

Hinweise:

- Der verwendete Computer und der Raspberry Pi müssen sich im selben WLAN-Netzwerk mit der oben beschriebenen Konfiguration befinden (Datei *wpa_supplicant.conf*).
- Die IP-Adresse des SSH-Befehls muss mit der Adresse des Raspberry Pi übereinstimmen. Diese kann beispielsweise im WLAN-Router ausgelesen werden.
- Das Default Passwort des Benutzers *pi* ist *raspberry*.
- Unter Windows kann eine SSH Client Software, wie z.B. PuTTY verwendet werden.

Nach der Anmeldung kann der Raspberry Pi über folgenden Befehl konfiguriert werden:

```
sudo raspi-config
```

Es müssen folgende Einstellungen vorgenommen werden:

- Dateisystem erweitern
- Hostname: fluxpi
- User-Passwort: fluxPiUser
- I2C aktivieren
- Zeitzone einstellen

Nach diesen Einstellungen muss der Raspberry Pi neugestartet werden und anschliessend erneut eine SSH-Verbindung aufgebaut werden.

Zuletzt müssen alle Flux-Komponenten installiert werden.

Automatisierte Installation

Für die automatisierte Installation des Raspberry Pi wird die Software PiBakery[7] benötigt. Nach der lokalen Installation und Ausführung von PiBakery kann unter *Import* die Konfigurationsdatei \Ressourcen\RaspberryPi\PiBakery\fluxpi.xml geladen werden.

[7] http://www.pibakery.org/

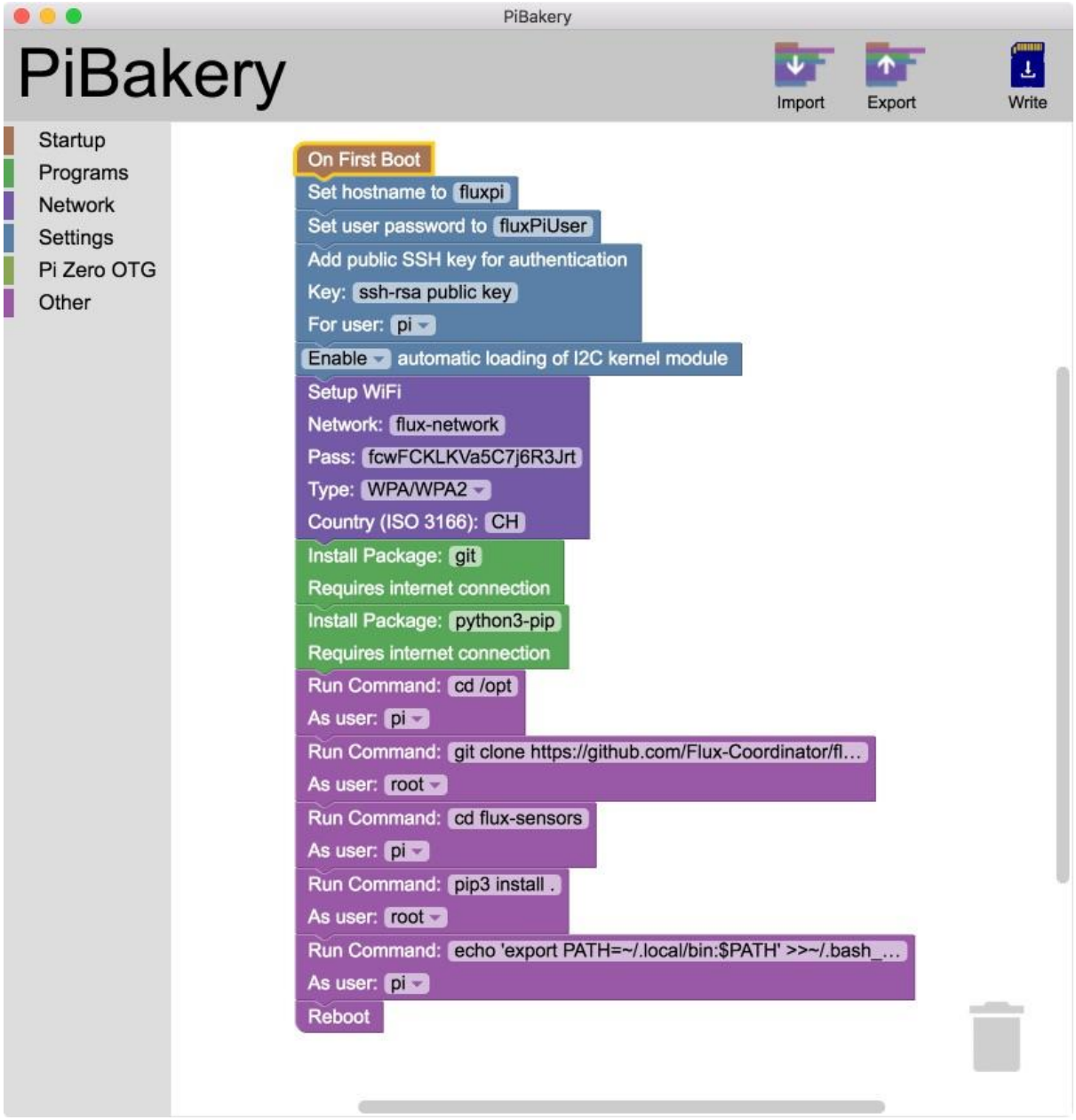


Abbildung 7 PiBakery mit importierter Konfiguration

Sofern die SD-Karte am Computer angeschlossen ist, kann nun die Konfiguration unter *Write* auf die SD-Karte geschrieben werden.

> **Warnung:** Das gewählte Laufwerk wird gelöscht und überschrieben, das heisst alle Daten, die darauf gespeichert sind, werden dauerhaft gelöscht.

Nach Abschluss des Schreibvorgangs kann die SD-Karte ausgeworfen und in den Raspberry Pi eingefügt werden. Direkt nach dem Einschalten des Raspberry Pi werden die Scripts ausgeführt und das System vorbereitet. Dieser Vorgang kann einige Minuten dauern.

Anschliessend kann analog zur manuellen Installation per SSH auf den Raspberry Pi verbunden werden. Nach der Anmeldung müssen über den Raspbian Wizard noch einige letzte Einstellungen manuell konfiguriert werden:

```
sudo raspi-config
```

Es müssen folgende Einstellungen vorgenommen werden:

- Dateisystem erweitern
- Zeitzone einstellen