HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

DFLD.de

Studienarbeit, Abteilung Informatik

Fluglärmmessung mit Raspberry Pi

Hochschule für Technik Rapperswil

Frühlingssemester 2013
31. Mai 2013

Autoren:	Pascal Langenstein & Renato Stähli
Betreuer:	Prof. Dr. Peter Heinzmann
Projektpartner:	DFLD
Arbeitsperiode:	18.02.2013 - 31.05.2013
Arbeitsumfang:	240 Stunden, 8 ECTS pro Student
Link:	`http://atmng.cnlab.ch`

1 Abstract

Abteilung	Informatik
Namen der Studierenden	Pascal Langenstein Renato Stähli
Studienjahr	FS 2013
Titel der Studienarbeit	Fluglärmmessung mit Raspberry Pi
Examinator	Prof. Dr. Peter Heinzmann
Themengebiet	Internet-Technologien und -Anwendungen / Raspberry Pi
Projektpartner	DFLD
Institut	ITA

Im Rahmen dieser Studienarbeit wurde ein System zur Fluglärmmessung, bestehend aus Raspberry Pi Messstationen, Daten- und Webserver entwickelt. Es unterscheidet sich von vergleichbaren Lösungen durch niedrigere Kosten, einfachere Installation und höhere Flexibilität.
Zur Realisierung der Messstationen wurde der Standard Raspberry Pi Bausatz (Gehäuse, SD-Karte, Netzteil) um einen Analog-Digital-Wandler mit Mikrofonbuchse (RJ11) erweitert. So kann das bewährte „modul bus“ Mikrofon weiterhin verwendet werden. Die entwickelte Python-Applikation tastet den Lärmpegel des Mikrofons mit 3Hz ab und rechnet die 18-Bit Delta-Sigma quantisierten Werte in dB(A) Lärmpegelwerte um. Die Kommunikation zum Datenserver erfolgt über HTTP (Standard Port 80) im JSON Format. Die so realisierte Messstation kostet 200 CHF, d.h. nur 40% der bestehenden Lösung. Beim Datenserver nimmt die mit dem Spring Framework realisierte Tomcat-Applikation die Lärmpegelwerte entgegen und speichert sie in der Open-Source Datenbank MySQL ab. Die realisierte Webanwendung zeigt dem Benutzer eine Karte mit allen registrierten Messstationen und Lärmkurven zu den Stationen. Die Darstellung der Lärmkurven erfolgt über das Framework Highcharts, welches neben diversen Zoomfunktionen auch die gleichzeitige Darstellung der Kurven von mehreren Stationen ermöglicht.
Insgesamt wurden drei Messstationen in Betrieb genommen. Der Vergleich der Messresultate mit denjenigen der geeichten GFS-Messstation zeigt, dass unser System noch weiter optimiert werden kann. Während der Entwicklung des Datenservers wurde automatisiert getestet, dabei kamen die Frameworks Junit und REST-assured zum Einsatz. Der entwickelte Code steht als Open-Source Projekt auf Github zur Verfügung. Die Applikation wird durch die HSR verwaltet und eventuell weiterentwickelt. Ebenso die drei Messstationen die in Betrieb genommen wurden.

2 Aufgabenstellung

Studienarbeit Fluglärmmessung

Studiengang:	Informatik
Institut:	ITA : Internet-Tech. und Anwendungen
Gruppe:	Langenstein, Pascal und Stähli, Renato
Betreuer:	Heinzmann, Peter
Industriepartner:	DFLD

Ausgangslage

Die “Air Traffic Monitoring“ Anwendung atmng.cnlab.ch erfasst Lärm-, Bild- und Bewegungsdaten von Flugzeugen im Raum Zürich. Nun soll das Lärmerfassungssystem und die Webanwendung erneuert werden. Insbesondere sollen die “modul bus“-Ethernet-Lärmmessstationen durch eine Lösung mit Raspberry Pi ersetzt werden.

Ziel

Als Lärmerfassungssystem sollen Raspberry Pi eingesetzt werden können. Es soll eine neue Webanwendung zur Anzeige der Lärmmesswerte und Auswahl der Stationen verfügbar sein. Das neue Lärmaufnahmesystem soll bei mindestens drei Messstationen im Einsatz sein und Messwerte zu atmng.cnlab.ch liefern.

Teilaufgaben

- Technologiestudie / Einarbeitung
  - Einarbeitung in Air Traffic Monitoring Anwendung
  - Welche Lärmmess-Möglichkeiten gibt es für Raspberry Pi? Pegel-/Eventerfassung, Sound Replay, . . .
  - Welche Möglichkeiten zur Erfassung der Geschwindigkeit von Fahrzeugen gibt es für Raspberry Pi?
- Vorstudie Datenauswertungsmöglichkeiten
  - Analyse/Vergleich verschiedener Lärmpegelmess-Systeme (Messung absoluter Pegelwerte, db(A)-Bestimmung)
  - Messung Ton (Audiosignal) und daraus die Bestimmung von db(A)
  - Analyse/Vergleich verschiedener Systeme zur Fahrzeugzählung/Geschwindigkeits- und Lärmmessung
- Entwurf neues Lärmmess-System
  - Praktische Aspekte (Installation Mikrofon, Kabeldurchführung, Stromversorgung)
  - Datenlieferung zu verschiedenen Stellen
  - Analyse/Beschreibung der Lärmpegel (Parametrisierung)
  - Kalibrierungsmöglichkeiten
  - Systemmanagement (Alarmierung bei Systemausfällen, Konfiguration, Administration)
    - Modul-ID, Seriennummern
    - Modul-Typ (z.B. Wetterstation WS2000, Mobotix-Kamera mit Audio, ...)
    - Sensoren: Sensor1, Sensor2, ...
    - Besitzer
    - Standort
    - Interface
    - Installationsdaten (Inbetriebsetzungsdatum)
    - Verfügbarkeitsdaten
  - Realisierung der S-Anwendung
    - Design/Spezifikation
    - Realisierung
    - Inbetriebnahme von drei Messstationen

Technologien

- Realisierung als Stand-alone-Anwendung (Java) mit Export zur ATM-Plattform

Referenzen, Beispiele

1. Hinweise zur Durchführung von Studienarbeiten: http://www.cnlab.ch/kurse/SADA/
2. P. Heinzmann, Präsentation Fluglärmmessung, PowerPoint-File AirTrafficMonitor-V4.0_hei, 2013.
3. Pressemitteillungen zum Thema “private Fluglärmmessungen“

Rapperswil, den .....

.....
Prof. Dr. Peter Heinzmann
(Betreuer)

3 Erklärung zur Urheberschaft

*D*iese Erklärung basiert auf der Muster-Erklärung in den Richtlinien der HSR zur Durchführung von Projekt-, Studien-, Diplom- oder Bachelorarbeiten vom 16. Februar 2009. Die vorliegende Arbeit basiert auf Ideen, Arbeitsleistungen, Hilfestellung und Beiträgen gemäss folgender Aufstellung:

Gegenstand, Leistung	Person	Funktion
Alle arbeiten gemeinsam	Pascal Langenstein	Autor der Arbeit
Alle arbeiten gemeinsam	Renato Stähli	Autor der Arbeit
Korrekturlesen, Hinweise zur sprachlichen Überarbeitung		
Korrekturlesen, Hinweise zur sprachlichen Überarbeitung		
Hilfestellungen Architekturfragen		Assistent HSR
Hilfestellungen bei Software- und Architekturfragen		
Idee, Aufgabenstellung, Unterstützung , Betreuung während der Arbeit	Prof. Dr. Peter Heinzmann , cnlab AG	Verantwortlicher Professor
LaTex-Template		

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit gemäss obiger Zusammenstellung selber und ohne weitere fremde Hilfe durchgeführt habe,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe,
- dass in der Arbeit verwendete urheberrechtlich geschützte (Copyright) Inhalte (insbesondere Fotografien und Grafiken) klar gekennzeichnet und mit Quellenhinweis versehen sind,

- dass Inhalte die unter Creative-Commons-Lizenz veröffentlicht wurden, klar gekennzeichnet sind.

Rapperswil, den 31.05.2013

Pascal Langenstein
(Student)

Renato Stähli
(Student)

4 Vereinbarung zur Verwendung und Weiterentwicklung der Arbeit

*D*iese Vereinbarung basiert auf den Muster-Vereinbarungen in den Richtlinien der HSR zur Durchführung von Projekt-, Studien-, Diplom- oder Bachelorarbeiten vom 16. Februar 2009.

1. **Gegenstand der Vereinbarung**
   Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit „Fluglärmmessung mit Raspberry Pi“ von Pascal Langenstein und Renato Stähli unter der Betreuung von Prof. Dr. Peter Heinzmann (für die Arbeit verantwortlicher Professor) geregelt.

2. **Urheberrecht**
   Die Urheberrechte stehen der Studentin/dem Student zu.

3. **Verwendung**
   Die Ergebnisse der Arbeit dürfen sowohl von allen an der Arbeit beteiligten Parteien, d.h. von den Studenten, welche die Arbeit verfasst haben, vom verantwortlichen Professor sowie vom Industriepartner verwendet und weiter entwickelt werden. Die Namensnennung der beteiligten Parteien ist bei der Weiterverwendung erwünscht, aber nicht Pflicht.

Rapperswil, den ....
Pascal Langenstein (Student)

Rapperswil, den ....
Renato Stähli (Student)

Rapperswil, den ....
Prof. Dr. Peter Heinzmann
(Verantwortlicher Professor)

5 Management Summary

5.1 Ausgangslage

Aufgabenstellung

Die “Air Traffic Monitoring“ Anwendung atm.cnlab.ch erfasst Lärm-, Bild- und Bewegungsdaten von Flugzeugen im Raum Zürich. Nun soll das Lärmerfassungssystem und die Webanwendung erneuert werden. Insbesondere sollen die Modulbus-Ethernet-Lärmmessstationen durch eine Lösung mit Raspberry Pi ersetzt werden.

Messsysteme

Momentan existieren für Amateur-Fluglärmmessungen zwei Systeme.
Zum einen die von ”modul bus” entwickelten Messsysteme, welche einen laufenden PC oder einen “Serial-to-Ethernet“ Konverter benötigen. An diesem System können bis zu zwei Lärmmesssonden angeschlossen werden. Die Zielgruppe sind Personen, welche eine Station betreiben wollen, keine exakten Messungen benötigen und bereit sind diesen ”Bausatz” selber zusammen zu bauen. Der Kostenpunkt mit Konverter liegt bei ca. 500 CHF.
Die andere Lösung von GfS mit Namen “Schallpegel-Monitor SPM 483“ ist ein vorkonfiguriertes und geeichtes Gerät mit Mikrofon. Dieses Komplettsystem hat ein wetterfestes Mikrofon, lässt sich einfach installieren und benötigt keinen PC. Der Kostenpunkt liegt hier bei ca. 3000 CHF. [Ges13]

Visualisierungen

Für Schweizer Flughäfen bestehen bisher zwei Internetauftritte welche Fluglärmmessungen anzeigen und diese Messsysteme unterstützen. In der Tabelle 5.1 ist eine Aufstellung aller aktiven Stationen vom 26.Mai 2013.

	DFLD[1]	ATM[2]
Flughafen Zürich	11 Stationen	8 Stationen
Flughafen Basel	12 Stationen	-
Flughafen Genf	-	2 Stationen

Tabelle 5.1: Aktive Stationen

Problematik

Mit dieser Ausgangslage ergeben sich folgende Hauptpunkte, welche verbessert werden können:

- Die Kosten für die komplette Messstation sind zu hoch.
- Die Inbetriebnahme von neuen Stationen ist zu komplex.
- Die Visualisierung ist für Fachpersonen bedienbar, für durchschnittliche Internetbenutzer jedoch zu kompliziert. Zudem ist die Antwortzeit der Seite zu langsam und gewisse Funktionen arbeiten nicht korrekt.

Ziel

Für das neue Lärmerfassungssystem soll das Raspberry Pi eingesetzt werden können. Es soll eine neue Webanwendung zur Anzeige der Lärmmesswerte und Auswahl der Stationen verfügbar sein. Das neue Lärmerfassungssystem soll bei mindestens drei Messstationen im Einsatz sein und Messwerte zu atmng.cnlab.ch liefern.

5.2 Vorgehen

Vorstudie

In der Vorstudie wurden die Möglichkeiten, welche das Raspberry Pi mit diversen Modifikationen bietet, eruiert. In der ersten Phase gingen wir auf die Anforderungen zur Verkehrszählung und -messung ein. Wir erkannten die Möglichkeiten, dass mittels Ultraschall, Ein- und Zweiweglichtschranken sowie Radarsensoren nicht nur die Anzahl, sondern auch die Geschwindigkeit von Fahrzeugen messen zu können. Das Ergebnis war nicht zufriedenstellend. Bei den Ultraschall-Sensoren war die Reichweite zu kurz um eine Geschwindigkeit zu messen. Gleichzeitig vertieften wir uns in die Thematik der Lärmmessung. So eigneten wir uns das benötigte Wissen für die bevorstehende Arbeit, deren Thema die Fluglärmmessung ist, an. In der zweiten Phase war die Herausforderung den bestehenden ”modul bus” Sensor zu reverse-engineeren und zu erkennen, wie wir dieses Signal für die Fluglärmmessungen benutzen können. Dies war nötig, da die Dokumentation der Firma “modul bus“ nicht auf den Lärmmesssensor sondern auf das Umrechnungsmodul bezogen ist. Hier half uns die Diplomarbeit zweier ZHAW-Absolventen, in welcher wir die entsprechende Umrechnungsformel [Sta04, s 36] fanden.
Nachdem diese Vorstudie abgeschlossen war, entschieden wir uns für einen Analog-Digitalwandler, welcher über die GPIO-Pins mit dem Raspberry Pi kommuniziert. Dieser ermöglichte uns das Signal des Messmodul abzugreifen und auszulesen. Entsprechende Beispiele fanden wir im Internet[htt13].
Jegliche Schritte, Bedenken und Probleme konnten wir bei den wöchentlichen Sitzungen diskutieren. Am Ende der ersten Hälfte erstellten wir eine Raspberry Pi Messtation und konnten unser Konzept somit beweisen.

Client und Server

In der zweiten Hälfte des Projektes ging es darum eine Client- und Serverapplikation zu erstellen. Die Technologievorgaben von der cnlab ag waren klar und wir sollten auf Produkte setzen, für die es bereits Know-How gibt. Dies ist in den Bereichen Java, Mysql und Highcharts vorhanden. Somit kann Wartung und Weiterentwicklung mit dem bestehenden Wissen durchgeführt werden und benötigt keine zusätzliche Einarbeitungszeit.

5.3 Ergebnisse

Es gelang uns die Probleme und Schwierigkeiten der Verkehrsmessung aufzuzeigen. Diese sind Anforderungen an die Sensoren, Montage, aber auch die Schwierigkeit einer Geschwindigkeitsmessung mit Doppler-Effekt. Wir konnten aufzeigen, dass es mit dem Raspberry Pi möglich ist den Fluglärm zu messen und sind in der Lage das bestehende ”modul bus” Messmodul Lärm zu verwenden.

In der zweiten Hälfte unserer Studienarbeit entwickelten wir die Software für Raspberry Pi und den Datenserver.
Die Visualisierung (Abbildung 5.3) zeigt die Lärmkurven der Messstationen sowie deren Standorte an. Es ist möglich die Lärmkurven der einzelnen Stationen mühelos zu vergleichen.

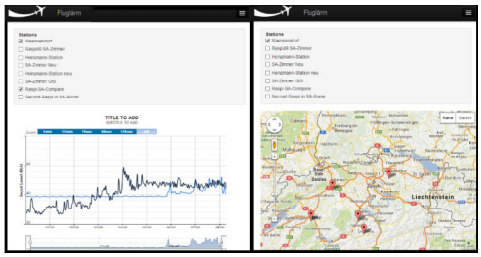
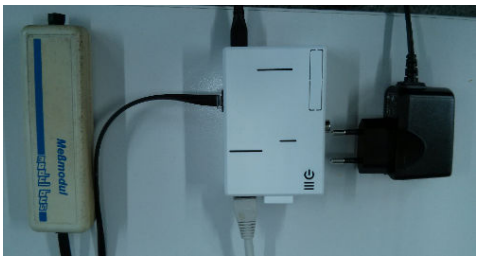


Abbildung 5.1: Webvisualisierung

Abbildung 5.2: Messstation komplett



Die Raspberry Pi Lösung spart 60% der Kosten. Die Kostenaufstellung in Tabelle B.1 führt nur benötigte Hardware für die Raspberry Pi-Messstationen ohne Lärmsensor auf. Die gesamten Hardwarekosten belaufen sich auf rund 210 CHF. Für das Löten und die Modifikationen am Gehäuse sollten ungeübte Personen zwei Stunden einrechnen. Abbildung 5.2 zeigt eine einsatzbereite Messstation.

Lernpunkte

Es zeigt sich, dass die Entwicklung des Raspberry Pi noch weitere Verbesserungen im USB-Bereich benötigt. So stellten wir fest, dass beim Ein- oder Ausstecken von USB-Geräten das Raspberry Pi abstürzte und nicht mehr hochfahren konnte. Im Gegensatz zu früheren Betriebssystemversionen hatten wir keine Probleme mit USB-Abstürzen ohne User Aktionen während dem laufenden Betrieb.

5.4 Ausblick

Zu beachten ist, dass die Applikation nicht von sich aus ältere Datenbankeinträge löscht. Pro Messung wird 27 Byte Speicherplatz benötigt, eine Messstation belegt am Tag somit ~2.2 Megabyte und monatlich ~66 Megabyte. Darum wurde überlegt einen Task zu erstellen der alte Einträge im JSON Format archiviert. Dieser Task ist in der Software auch schon vorbereitet.

Die Applikation wird als Open-Source Projekt auf GitHub gestellt. Somit kann jede Person diese Applikation weiter entwickeln. Um die Administration zu erleichtern, sollte der Administrationsbereich implementiert werden, sonst müssen alle Konfigurationen und Mutationen in der MySQL-Datenbank vorgenommen werden. Die Prototypen für einen Administrationsbereich finden sich im Anhang E. Vorbereitete und angedachte Features sind im Anhang G festgehalten.

Inhaltsverzeichnis

6 Einleitung

Im Rahmen der Studienarbeit wurde ein Fluglärm Messsystem entwickelt welches es ermöglicht in Regionen mit täglichem Fluglärm kontinuierliche Messungen durchzuführen. Dabei wurde eine Messstation entwickelt, welche es ermöglicht einen Lärmpegel zu erfassen und diesen über längere Zeit auf einen Datenserver zu speichern. Eine weitere Aufgabe bestand darin die erfassten Lärmpegel in einer Webplattform darzustellen und mit anderen Messstationen vergleichen zu können.

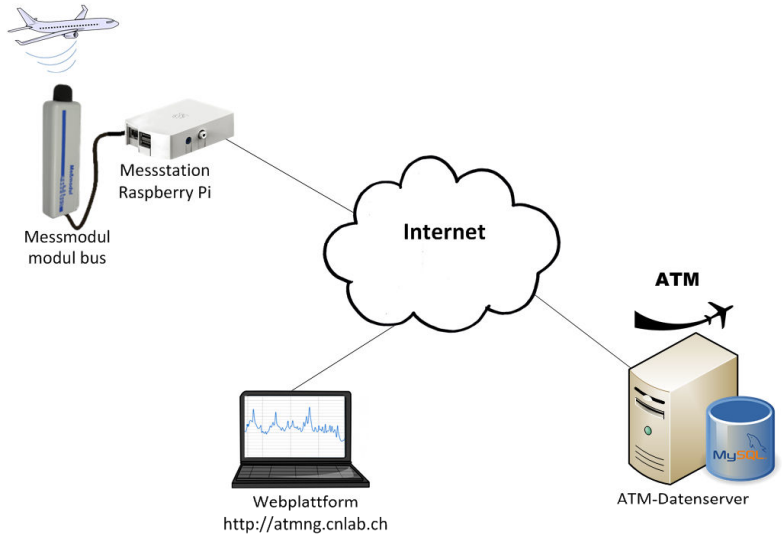


Abbildung 6.1: Big Picture

Die Idee Fluglärm aufzuzeichnen ist nicht neu. Es gibt verschiedene Produkte welche es ermöglichen Lärmpegel aufzuzeichnen. Der Preis der auf dem Markt verfügbaren Messstationen beträgt zwischen 500 - 3000 CHF. Dies ist für Privatpersonen die sich über den Lärm an ihrem Standort interessieren zu viel. Die entwickelte Messstation welche auf einem Raspberry Pi aufbaut, sollte daher deutlich günstiger sein. Mit einem Materialpreis von rund 200 CHF für die gesamte Messstation ist dies gelungen. Nebst dem Preisvorteil bietet die neue Messstation auch Vorteile in der einfacheren Handhabung, sowie in einer verringerten Fehleranfälligkeit.

Kapitel 7 - Systemübersicht: richtet sich an Leser, die eine schnelle und grobe Übersicht erhalten wollen.

Kapitel 8 - Messstation: richtet sich an Leser, die sich für die Hard- und Software der Messstation interessieren.

Kapitel 9 - Datenserver: richtet sich an Leser, welche die Serversoftware warten oder weiterentwickeln wollen.

Kapitel 10 - Datenvisualisierung: richtet sich an Benutzer der Webapplikation.

Kapitel 11 - Testing: richtet sich zum einen an Leser, welche wissen wollen wie die Software und Hardware getestet wurde.

Anhang A - Schnittstellenbeschreibung JSON: ist für Leser, welche selber einen Client implementieren oder eine eigene Lärmkurvenvisualisierung erstellen wollen.

Anhang B - Bausatz und Kosten: richtet sich an Leser, die selbst eine Messstation bauen wollen.

Anhang C - Installationsanleitung: richtet sich an Leser, welche eine Messstation aufsetzen wollen.

7 Systemübersicht

Das gesamte Fluglärmmesssystem besteht insgesamt aus vier Komponenten, welche in Abbildung 7.1 aufgezeigt sind. Dazu wird zuerst ein grober Überblick über die beteiligten Komponenten gegeben. Danach wird auf die zwei Perspektiven ”Messstation zu Datenserver” und ”Datenserver zu Webplattform” genauer eingegangen.

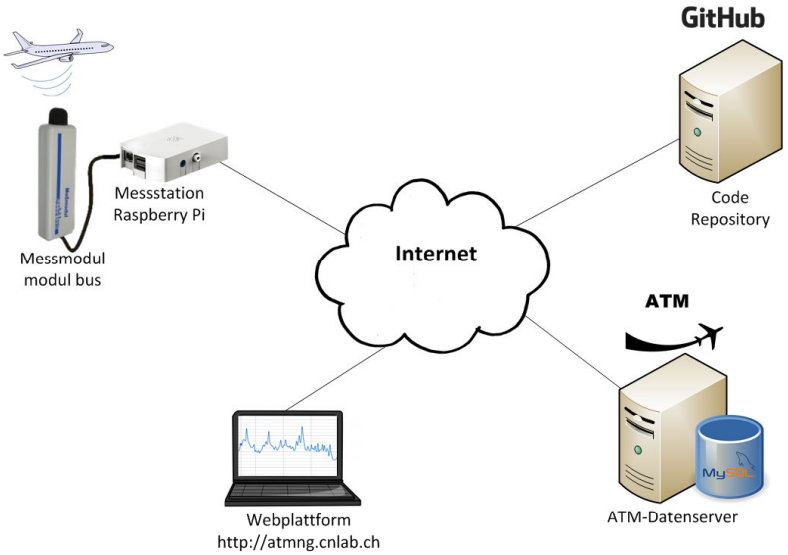


Abbildung 7.1: Systemübersicht, Perspektive Raspberry Pi

Messstation: Die Messstation mit Messmodul ist für das Erfassung des Lärmpegels zuständig und liefert die Daten über das HTTP Protokoll an den Server weiter.

GitHub Repository: Das GitHub Repository verwaltet den aktuellen Quellcode für die Messstationen.

Datenserver: Der Datenserver hat mehrere Aufgaben. Zum Einen empfängt er die Lärmpegel der Messstationen, zum Anderen stellt er über die REST-Schnittstelle die gespeicherten Lärmpegel img JSON-Format zur Verfügung.

Webplattform: Über die Webplattform werden die Daten in einer Lärm-Zeit-Chart dargestellt. Es können dabei verschiedene Lärmkurven miteinander verglichen werden.

7.1 Perspektive Messstation

Die Messstation sendet im Sekundentakt die erfassten Lärmpegel an den Datenserver [blauer Pfeil]. Die Übertragung der Daten erfolgt über das HTTP Protkoll im JSON Format.
Beim Starten sendet die Messstation dem Datenserver ihre MAC-Adresse [roter Pfeil]. Der Datenserver antwortet mit den benötigten Informationen. Der wichtigste Parameter dabei ist die Stations ID, mit welcher ab nun die Lärmpegel an den Server gesendet werden. Dieser Art der Konfiguration ermöglicht es keinerlei Konfigurationseinstellungen auf der Messstation zu speichern. Daher kommt auf allen Messstation der gleiche Code zum Einsatz.
Täglich um 1 Uhr Morgens prüft die Messstation ob neue Softwareupdates auf dem GitHub Repository zur Verfügung stehen [grüner Pfeil]. Ist dies der Fall, lädt es diese herunter und startet anschliessend das Betriebssystem mit der nun aktualisierten Anwendung neu.

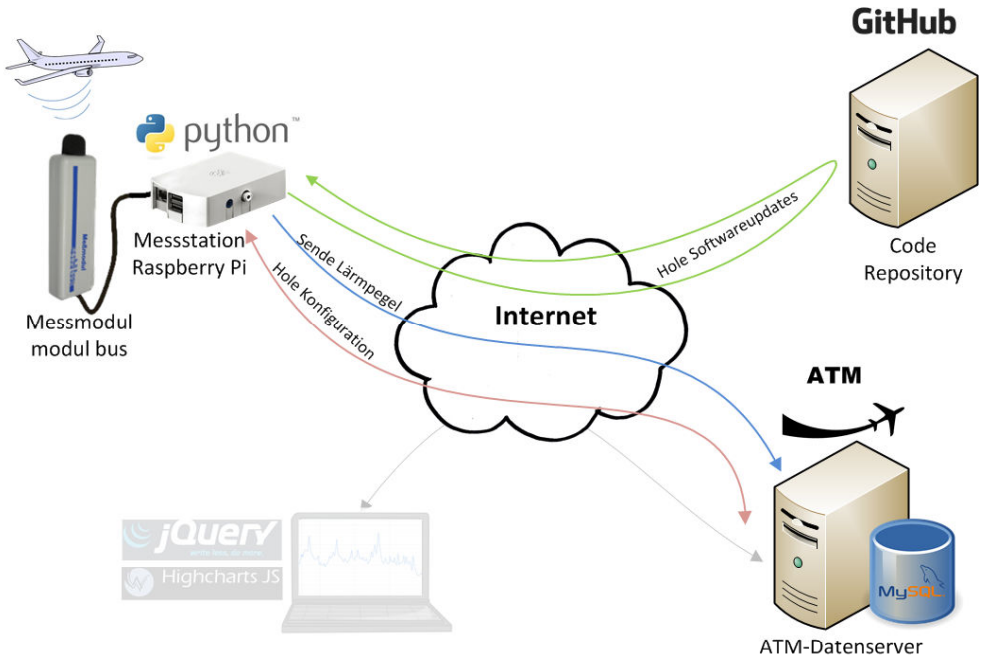


Abbildung 7.2: Systemübersicht, Perspektive Raspberry Pi

7.2 Perspektive Webplattform

Die zweite Perspektive zeigt den Aufruf der Webplattform. Hier wird zuerst die Website über HTTP geladen. Das JavaScript Framework jQuery lädt über HTTP über die REST-Schnittstelle im JSON Format die Stationsliste. Klickt der Benutzer eine Messstation an, lädt jQuery die gewünschten Daten und übermittelt diese dem Highcharts Framework. Dieses bereitet die im JSON-Format erhaltenen Daten auf und zeigt die Lärmkurven an. Für jede Messstation werden die Daten einzeln geladen. Damit ist es möglich neue Stationen in der aktuelle Ansicht hinzuzufügen, ohne dabei alle Stationen nachladen zu müssen.

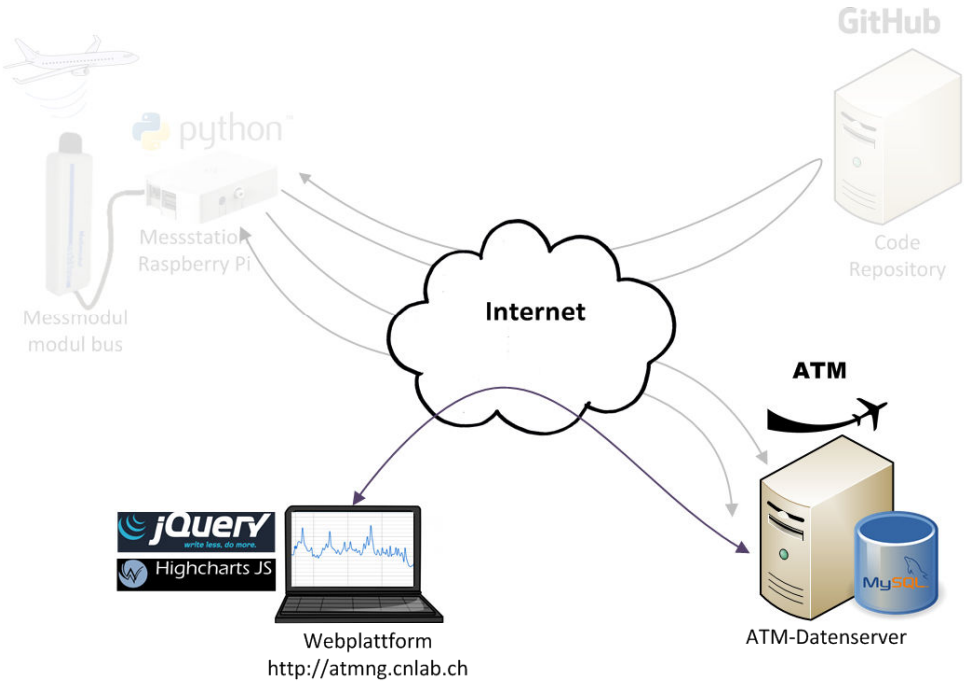


Abbildung 7.3: Systemübersicht, Perspektive Webplattform

8 Messstation

8.1 Übersicht Messstation

Die Messstation ist für das Erfassen des Lärmpegels zuständig. Die Messstation besteht aus insgesamt drei Komponenten, welche in Abbildung 8.1 dargestellt sind. Das Messmodul [1] erfasst den Lärmpegel und stellt den Wert als analoge Spannung zur Verfügung. Dieser Spannungswert wird über ein Analog-Digital-Wandler[2] in ein digitales Signal umgewandelt. Das digitale Signal wird anschliessend über den I2C-Bus des Raspberry Pi[3] verarbeitet. Eine genaue Beschreibung der einzelnen Komponenten folgt im nächsten Abschnitt.

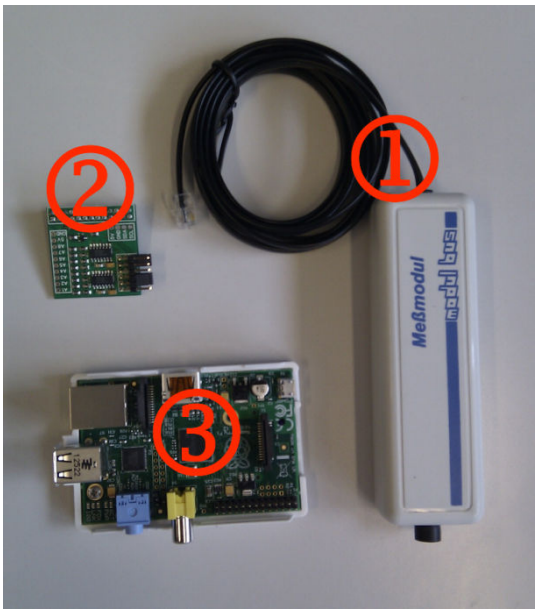


Abbildung 8.1: Messstation - Messmodul[1], Analog-Digital-Wandler[2] Raspberry Pi[3]

8.2 Hardware Komponenten

8.2.1 Messmodul Modulbus

Abbildung 8.2: Messmodul Lärm der Firma ”modul bus”

Das Messmodul der Firma ”modul bus” funktioniert nach folgendem Prinzip: Das Kondensatormikrofon gibt eine zur Schallintensität proportionale Wechselspannung ab, die durch den Vorverstärker verstärkt wird. Es folgt ein Filter zur Bewertung der Intensität nach der genormten dB(A)-Kurve, welche der Empfindlichkeit des menschlichen Gehörs angepasst ist.

Das gefilterte Signal durchläuft einen logarithmischen Gleichrichter, sodass die Ausgangsspannung logarithmisch mit der Schallintensität wächst. Sie ist damit proportional zum bewerteten Schallpegel in dB(A). Die Kalibrierung erfolgt durch ein internes Trimmpotentiometer. [Sta04]

Technische Daten:

- Betriebsspannung: 4V...6V
- Stromaufnahme: ca. 5mA
- Messbereich: 30dB(A) .... 100dB(A)
- Ausgangsspannung $(\frac{U}{V})$: ca. 1,0V ... 2,5V
- Übertragungsfunktion: $L = (100 * (\frac{U}{V}))/2) - 25)dB(A)$

8.2.2 Analog-Digital-Wandler

Der Analog-Digital-Wandler der Firma AB Electronics UK wurde speziell für das Raspberry Pi entwickelt. Es handelt sich dabei bereits um die zweite Version. Der Wandler passt sehr genau in das Gehäuse. Dazu müssen einzig die verlängerten Pins abgetrennt werden. Diese sind für die Verwendung von mehreren aufeinander gesteckten Wandler gedacht. Die Lötstellen für das Anbringen des Kabels sind optimal positioniert. Preislich ist der Analog-Digital-Wandler mit umgerechnet 27 CHF etwas teuer. Insgesamt können bis zu 8 Analogwerte eingelesen werden. Es können verschiedene Datenraten mit entsprechender Auflösung konfiguriert werden. Diese findet man im Datenblatt[1].

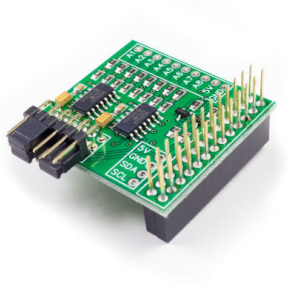


Abbildung 8.3: Analog-Digital-Wandler

1. http://www.abelectronics.co.uk/docs/stock/raspberrypi/deltasigma/mcp3424.pdf

Messmodul Adapter

Damit das Lärmmodul nicht direkt mit dem Analog-Digital-Wandler verlötet werden muss, wurde ein RJ11-Adapter eingebaut. Der Stecker des Lärmmoduls kann so an der Buchse des Raspberry Pi eingesteckt werden und ist mit dem Analog-Digital-Wandler verbunden. Die genaue Anleitung ist im Anhang B angefügt. Damit der Adapter in das Gehäuse passt, sind Anpassungen am Gehäuse notwendig.

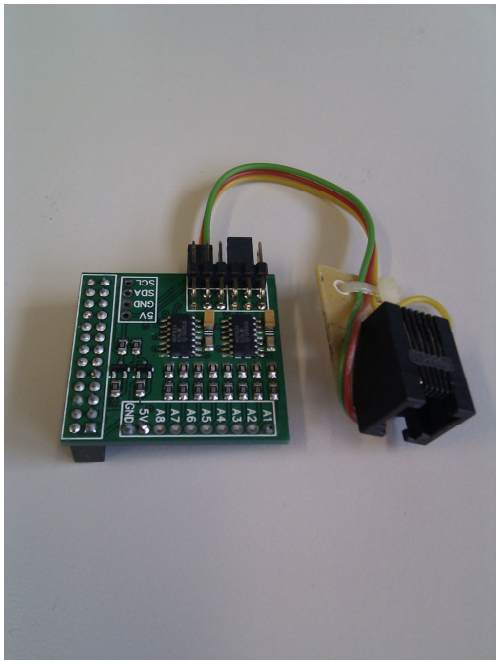


Abbildung 8.4: Messstation Mikrofon Print

8.2.3 Raspberry Pi

Das Raspberry Pi ist ein kreditkartengrosser Computer, welcher in zwei unterschiedlichen Versionen erhältlich ist. [2] Die eingesetzte B-Version (TypB) verfügt über einen 700-MHz-ARM-Prozessor sowie 512MB Arbeitsspeicher. Die B-Version besitzt zusätzlich eine Ethernet-Schnittstelle sowie zwei USB Anschlüsse. Für die persistente Speicherung der Daten steht eine SD-Schnittstelle zur Verfügung. Zudem stehen 16 programmierbare GPIO Pins bereit, welche es ermöglichen speziell dafür entwickelte Hardwarekomponenten zu installieren. Neben der B-Version des Raspberry Pi ist auch eine A-Version erhältlich. Da diese Version keinen Netzwerkanschluss hat sowie die preisliche Differenz von 12 CHF sehr klein ist, wird nur die B-Version eingesetzt.

Da eine normale SD-Speicherkarte aus dem Gerät hervorsteht und zudem sehr locker sitzt, kann es durch äussere Einflüsse zu ungewollten Störungen kommen. Die Lösung des Problems bietet ein Adapter, der die Verwendung einer microSD-Karte ermöglicht. Die microSD-Karte wird seitlich in den Adapter eingeschoben. Anschliessend wird der Adapter in den Raspberry Pi SD-Slot eingeschoben. Die Speicherkarte hat damit einen festen Sitz und kann nicht unabsichtlich entfernt werden. Der Adapter ist

2. http://de.wikipedia.org/wiki/Raspberry_Pi#Spezifikationen

etwas dicker als eine normale Speicherkarte, daher ist das vom Hersteller des Adapters empfohlene Gehäuse zu verwenden. Als Betriebssystem kommt das auf Debian basierende Raspbian zum Einsatz. Die genaue Beschreibung dazu finden Sie im nächsten Abschnitt.

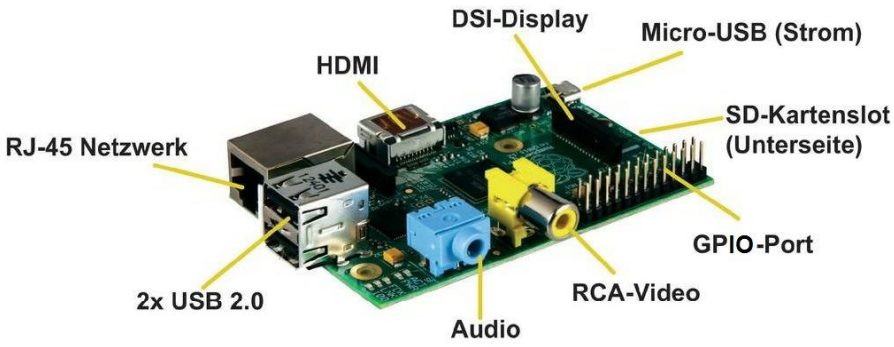


Abbildung 8.5: Raspberry Pi

Technische Daten:

- CPU: ARM1176JZF-S (700 MHz)
- Arbeitsspeicher: 512 MB
- USB 2.0 Anschlüsse: 2
- Schnittstellen: Bis zu 16 GPIO-Pins, SPI, I²C, UART

8.3 Software

8.3.1 Betriebssystem

Raspbian
Für das Raspberry Pi sind verschiedene Open-Source-Betriebssysteme verfügbar. Das bekannteste Betriebssystem heisst Raspbian, welches auf dem Betriebssystem Debian basiert. Die verwendete Version heisst Raspbian “wheezy“. Für die Installation des Raspbian Images sowie der Aktivierung der benötigten Treiber für den Analog-Digital-Wandler befindet sich die Installationsanleitung im Anhang C.

8.3.2 SoundLevelMeter

Programmiersprache
Die Software, welche den Lärmpegel auf dem Raspberry Pi erfasst und verarbeitet, ist in Python geschrieben. Die Wahl fiel auf die Sprache Python, weil die Library, welche auf den Analog-Digital-Wandler zugreift, in Python geschrieben ist. Der Name dieser Library lautet ”quick2wire”. Es hätte ebenfalls die Möglichkeit gegeben mit der Java Library ”Pi4J” direkt auf den SPI[3]-Bus zuzugreifen. Der Nutzen sowie der Aufwand hätten sich jedoch nicht gelohnt.

3. Serial Peripheral Interface

Ablauf der Datenerfassung
Das Python Script fragt alle 0.33 Sekunden den Digital-Analog-Wandler nach einem Wert. Nachdem 3 Werte gesammelt sind, wird aus diesen der Mittelwert berechnet und mit dem aktuellen Zeitstempel an den Server gesendet. Es werden daher im Sekundentakt Messwerte an der Server gesendet.

Abbildung 8.6: Messung im Sekundentakt

Um einen konstanten Takt von einer Sekunde einzuhalten, wartet das Raspberry Pi bis die Sekunde vorüber ist, um mit der neuen Messung zu beginnen. Die Übertragung der Daten erfolgt über das HTTP Protokoll im JSON-Format. Für das Senden des Wertes wird die Python Library ”requests” verwendet. Es handelt sich dabei um ein POST Request. Die genaue Beschreibung, an welche URL die Daten gesendet werden, ist im Anhang A zu finden.

8.4 Installation

Das Einrichten des Betriebssystems, die anschliessende Installation der benötigten Treiber sowie der Lärmpegel-Software ist im Anhang C beschrieben.

8.5 Betrieb

Für den Betrieb der Station ist ein Internetanschluss über ein Kabel sowie ein Stromanschluss erforderlich.

Die Lärmpegel-Software startet automatisch bei Systemstart und läuft danach selbstständig. Das Script wurde so entwickelt, dass Fehler im laufenden Betrieb nicht zum Absturz führen, beispielsweise bei Unterbruch der Internetverbindung. Der Fehler wird in einem Logfile gespeichert. Mit dem Logfile lässt sich mitverfolgen welche Daten an den Server gesendet werden. Die Anleitung ist im Anhang C.3.

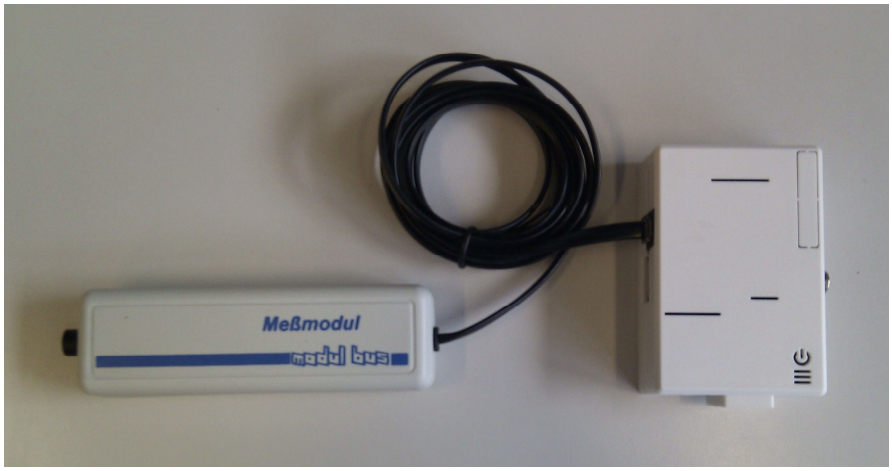


Abbildung 8.7: Messstation komplett

9 Datenserver

Dieses Kapitel ist an Entwickler gerichtet, es beinhaltet den Aufbau und die Struktur der Applikation.

9.1 Struktur der Applikation

Die Applikation läuft mit Java 1.6 auf einem Tomcat 7 Webserver und -container. Sie ist in drei verschiedene Schichten aufgebaut. Auf der Abbildung 9.1 sind diese ersichtlich. Als wichtigstes Framework wurde Spring eingesetzt, die vollständige Liste ist im Anhang F

9.2 Schichtenmodell

Der Persistence-Layer regelt den Datenbankzugriff. Auf diesem setzt der Service-Layer auf, der die gesamte Business-Logik abarbeitet. Im UI-Layer befinden die Controller, welche von aussen ansprechbar sind. Dies geschieht über eine REST Schnittstelle und dem JSON-Format. Es wird mit Dependency Injection gearbeitet, die durch das Spring Framework gemanagt wird. Wie in der Grafik 9.1 ersichtlich, ist die Applikation in drei

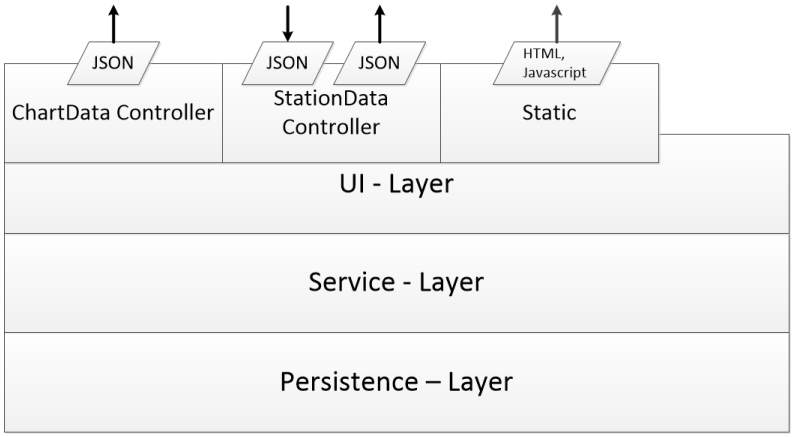


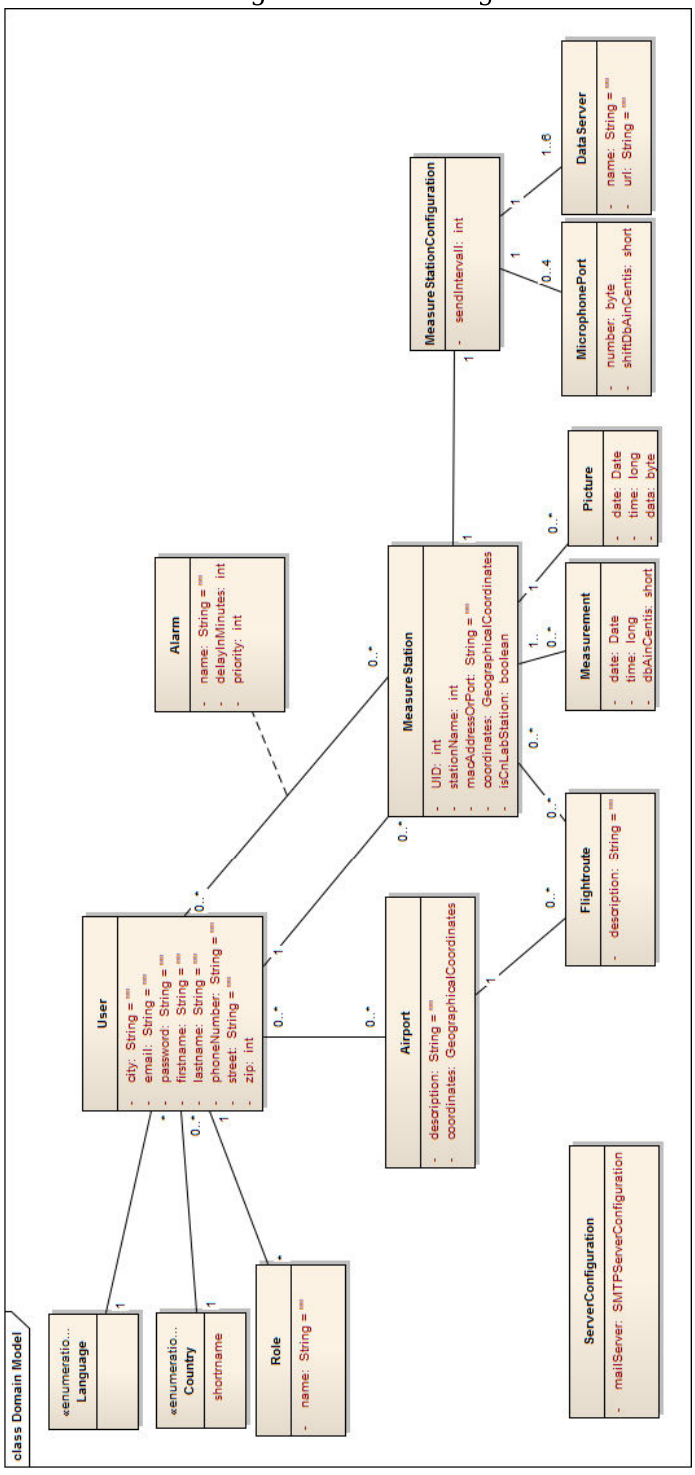
Abbildung 9.1: Struktur der Applikation

Layers geteilt. Der Persistence-Layer speichert und liest von der Datenbank. Der Service-Layer bereitet die Daten entsprechend auf, hier wird auch die Businesslogik abgewickelt. Im Web-Layer (Facade) werden die aufbereiteten Daten in JSON-Modelle geschrieben oder Anfragen für den Service Layer vorbereitet.

9.3 Klassendiagramm

Aus diesem Klassendiagramm (Abbildung 9.2) wird das Datenbankschema generiert. Zu beachten ist das die Enums ”Language” und ”Country” nicht in der Datenbank abgelegt werden, dies würde zu einem grossen Overhead führen.

Abbildung 9.2: Klassendiagramm



9.4 Datenbankschema

Das Datenbankschema wird von Hibernate aufgrund der Annotationen in den Domainobjekten gebildet. Daher entfällt ein eigentliches Schema.

9.5 Datenverarbeitung

Die Daten werden zwischen den Clients und dem Datenserver immer im JSON-Format über HTTP und Port 80 ausgetauscht. Dies verhindert Firewall Probleme und ermöglicht ein einfacheres Debuggen mit dem Google Chrome Dev Tool.

9.5.1 Datenschnittstelle

Als einheitliche Datenschnittstelle wird JSON benutzt. Auf Serverseite existieren dafür JSON-Models, welche die Inhalte klar spezifizieren. Die Dokumentation zur Schnittstelle findet sich unter Appendix A.

9.5.2 Scheduler

Auf dem Server ist ein Scheduler eingerichtet. Dieser wird stündlich gestartet und erstellt eine Mittelung der Stundendurchschnitte für jede Station. Um den Service zu konfigurieren, muss die XML-Datei “scheduling-context.xml“ angepasst werden.

```
<task:scheduled-tasks scheduler="taskScheduler">
        <task:scheduled ref="cronService" method="performService" cron="0␣15␣*␣*␣*␣?"
             />
</task:scheduled-tasks>
```

Hier wird ein Cronjob definiert, wobei zu beachten ist, dass bei dieser Art Cronjob die Sekunde das erste Feld ist. Dieser wird jede Stunde einmal um .15 gestartet.

```
<bean id="cronService" class="ch.cnlab.atm.scheduling.CronServiceImpl" />
```

Das Mapping zur konkreten Implementation wird mittels Bean-Tag realisiert.

```
<task:scheduler id="taskScheduler" pool-size="4" />
```

Zu beachten ist, dass der Thread-Pool noch Worker zugewiesen bekommt. Dies passiert ebenfalls in dieser Datei.

Alternativ stehen noch ”fixed-delay” und ”fixed-rate” zur Auswahl. Der Service “fixed-delay“ wartet nach der Ausführung die entsprechende Zeit ab, bevor er den selben Task wieder startet. Der “fixed-rate“ Service hingegen startet den Task immer nach Ablauf des eingestellten Intervalls.

Im Package ch.cnlab.atm.scheduling ist ein Interface abgelegt.

```
public abstract interface Service {

    public abstract void performService() throws InterruptedException;
}
```

Um einen eigenen zeitgesteuerten Task zu erstellen, kann man dieses Interface implementieren. Zur definierten Zeit wird performService() aufgerufen.

10 Datenvisualisierung

Die Präsentation der aufgezeichneten Lärmdaten erfolgt über die Webseite ”http://atmng.cnlab.ch”. Die Webseite besteht aus insgesamt zwei Ansichten. Beim Aufruf der URL landet man direkt in der Zeitverlaufs-Ansicht, welche den Lärmpegel über einen Lärm-Zeit-Graphen anzeigt. Über den Navigationspunkt ”Karte” wird auf die zweite Seite gewechselt. Diese bietet mittels Google Maps einen Überblick über alle registrierten Messstationen. Auf beiden Ansichten wird das gleiche Stationsmenu verwendet, um dem Benutzer ein übersichtliches und einheitliches Auswählen der Stationen zu erlauben. Das An-, sowie Abwählen von Stationen wird zwischen den zwei Ansichten synchronisiert. Beim Wechsel bleiben somit die ausgewählten Stationen erhalten. Die nachfolgende Abbildung zeigt die zwei Hauptansichten.

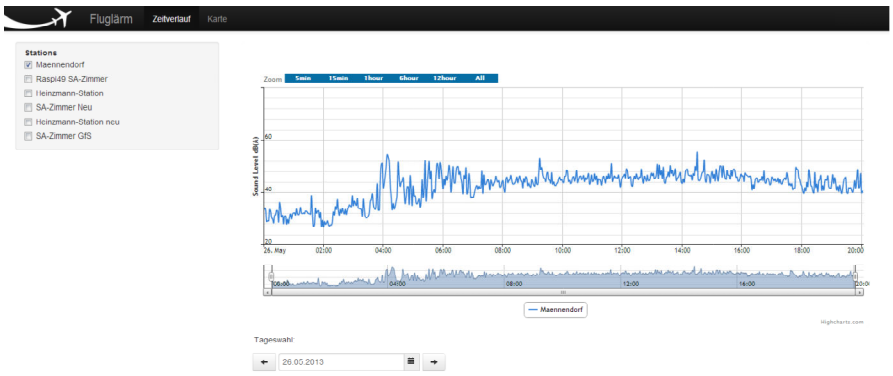


Abbildung 10.1: Webplattform Zeitverlauf

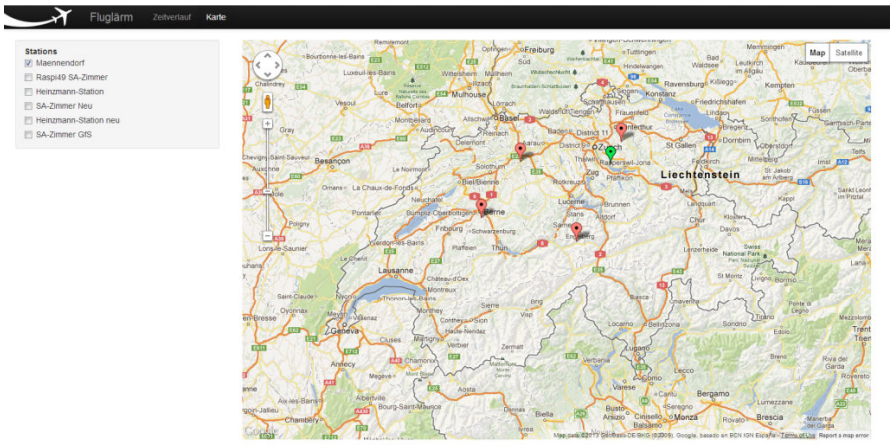


Abbildung 10.2: Webplattform Kartenansicht

10.1 Zeitverlauf

Es war wichtig den Zeitverlauf des Lärmpegels so übersichtlich wie möglich darzustellen, um auch neuen Besuchern einen leichten Einstieg zu gewährleisten. Die Wahl des Zeitbereichs war dabei von zentraler Wichtigkeit. Uns hat das Prinzip des Deutschen Fluglärmdienstes DFLD sehr gut gefallen. DFLD zeigen den Lärmpegel des ganzen Tages an. Dies bietet einerseits eine gute Übersicht und ist nach unseren Recherchen diejenige, welche dem Verwendungszweck am Besten entspricht. Eine andere Möglichkeit wäre gewesen nur die letzte Stunde anzuzeigen und über eine vordefinierte Auswahl weitere Zeitbereiche anzuzeigen.
Mit dem Entscheid den ganzen Tag anzuzeigen, ist es sehr einfach verschiedene Bereiche des Tages genauer anzuschauen. Mit dem Markieren eines Zeitbereichs kann in die einzelnen Stunden hineingezoomt werden. Mit der Navigator-Scrollbar[1] kann im Tagesverlauf vorwärts und rückwärts navigiert werden. Über die Zeitauswahl [2] können vordefinierte Bereiche ausgewählt werden. Ebenfalls ist es möglich mit der Maus einen Bereich auf der Hauptkurve[3] zu wählen, um damit diesen genauer zu untersuchen.

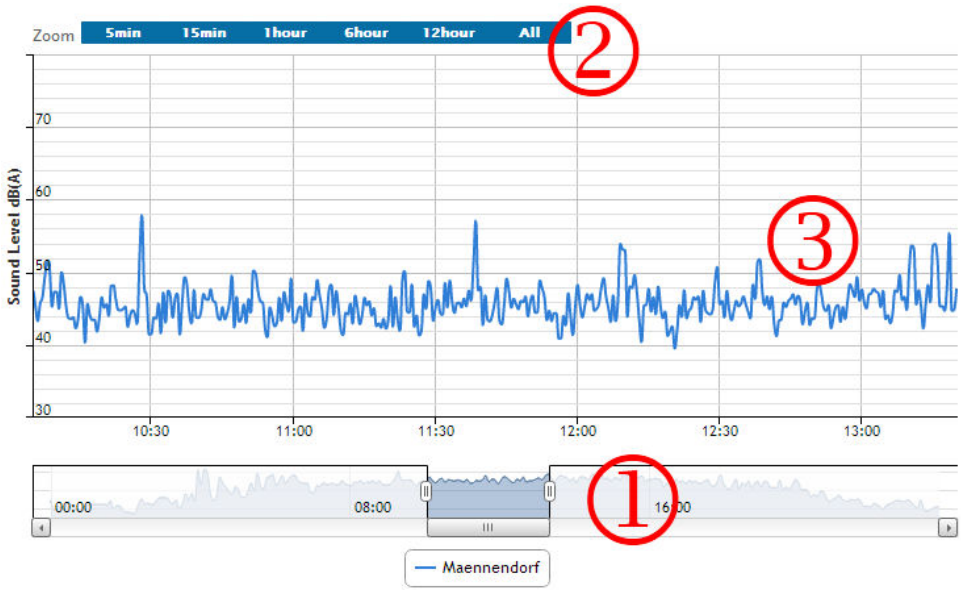


Abbildung 10.3: Zeitverlauf

10.2 Kartenansicht

Bei der Kartenansicht werden alle vorhandenen Stationen auf der Karte mit einem Marker angezeigt. Dabei sind ausgewählte Stationen grün und nicht markierte Stationen rot auf der Karte dargestellt. Es ist ebenfalls möglich die Stationen mit einem Klick auf die Marker zu selektieren.

10.3 Technologien

Highcharts

Für die Darstellung haben wir uns in Absprache mit Herr Prof. Heinzmann für das Framework Highcharts entschieden. Dieses darf für nicht kommerzielle Projekte ohne Lizenzkosten verwendet werden. Das Framework ist sehr genau dokumentiert und bietet zu fast jeder Option ein selbsterklärendes Beispiel. Wir verwenden dabei das Produkt Highstock[1], da dieses die automatische Gruppierung mehrerer Messpunkte ermöglicht. Das Framework benötigt die JavaScript Library jQuery, welches im HTML-Header eingebunden werden muss.

GoogleMaps

Als Kartendienst wird Google Maps verwendet. Es wird die API Version 3[2] eingesetzt.

Bootstrap

Abbildung 10.4: Responsive Design

1. http://api.highcharts.com/highstock
2. https://developers.google.com/maps/documentation/javascript/reference

Als Grundlayout der Webplattform wird das Open-Source Framework Bootstrap von Twitter eingesetzt. Dieses passt das Layout der Website an die Bildschirmgrösse an (”Responsive Design”). In der Abbildung 10.4 sind verschiedene Auflösungen dargestellt. Beim kleinsten Bildschirm sieht man den Umbruch. Abbildung 10.5 zeigt einen Screenshot. Die Navigationsleiste ist jetzt über einen Menü-Button zugänglich. Die Lärmkurve ist nun unterhalb der Stationsliste.

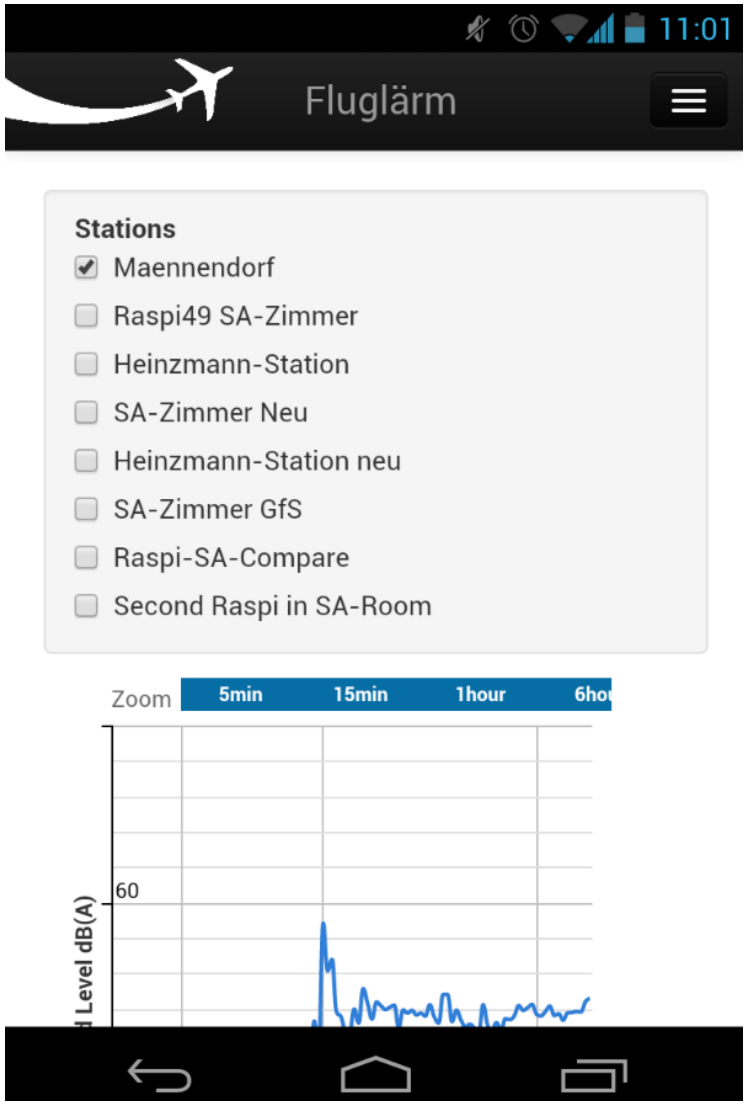


Abbildung 10.5: Responsive Design Mobile

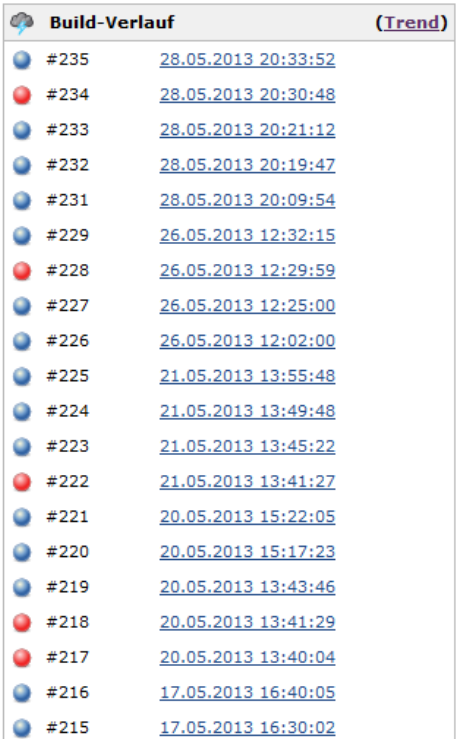
11 Testing

Dieses Kapitel richtet sich an Personen, welche die Applikation warten und noch wenig oder keine Erfahrungen mit dem Testen haben. Um wart- und erweiterbare Software zu schreiben, ist Testen unbedingt notwendig. Ebenso verlässt sich ein Benutzer der REST-Schnittstelle auf valide Daten.

11.1 Datenserver

Um die Funktionen manuell zu Testen, wurde die RestConsole verwendet, welche eine Erweiterung für den Google Chrome Browser ist. Ein Jenkins Server sorgt für die kontinuierliche Ausführung von Tests. Auf Abbildung 11.1 zeigt den Build-Verlauf. Ist ein Fehler aufgetreten wird die Ampel rot, ist sie blau bedeutet, dies, dass der Build fehlerfrei durchgelaufen und kein Test fehlgeschlagen ist. Wenn nun fehlerhafter Code in die Versionsverwaltung eingecheckt wird, kann genau eruiert werden, wer, wann, was geändert hat.

Abbildung 11.1: Jenkins-Buildhistory



11.1.1 Restassured

Während der Entwicklung verwendeten wir das Framework REST-assured[Joh13]. Es ermöglicht eine REST Schnittstelle mit JSON zu Testen. Folgendes Beispiel prüft ob die Station “Männedorf“ mit der ID “1“ vorhanden ist.

```
@Before
public void setUp() {
        RestAssured.registerParser("text/json", Parser.JSON);
}
@Test
public void testStationsListMaennedorf() {
        expect().statusCode(200)
        .body("stationList.id", hasItems(1), "stationList.name", hasItem("
            Maennedorf"))
        .when().get("/public/0/getStations");
}
```

11.1.2 JUnit und Mockito

JUnit

Um die Funktionalitäten zu Testen, verwendeten wir das Java Framework JUnit. Folgendes Beispiel zeigt einen Test für das Domain-Objekt “Data-Server“.

```
package ch.cnlab.atm.domain;

import junit.framework.Assert;

import org.junit.Before;
import org.junit.Test;

public class DataServerTest {
        DataServer ds;

        @Before
        public void setUp() throws Exception {
                ds = new DataServer();
                ds.setName("foo");
                ds.setUrl("http://foo.bar");
        }

        @Test
        public void testGetName() {
                Assert.assertEquals("foo", ds.getName());
        }

        @Test
        public void testSetName() {
                Assert.assertEquals("foo", ds.getName());
                ds.setName("bar");
                Assert.assertEquals("bar", ds.getName());
        }

        @Test(expected = IllegalArgumentException.class)
        public void testSetEmptyname() {
                ds.setName("");
                Assert.fail("should␣throw␣exceptoin");
        }

        @Test(expected = IllegalArgumentException.class)
        public void testSetNullname() {
                ds.setName(null);
                Assert.fail("should␣throw␣exceptoin");
        }

        @Test
        public void testGetUrl() {
                Assert.assertEquals("http://foo.bar", ds.getUrl());
        }

        @Test
        public void testSetUrl() {
                Assert.assertEquals("http://foo.bar", ds.getUrl());
                ds.setUrl("atmnb.cnlab.ch/atm");
                Assert.assertEquals("atmnb.cnlab.ch/atm", ds.getUrl());
        }

        @Test(expected = IllegalArgumentException.class)
        public void testSetNullUrl() {
                ds.setUrl(null);
                Assert.fail("should␣throw␣exception");
        }
}
```

Mockito

Mit dem Mockito Framework lassen sich die einzelnen Objekte und ihr Verhalten mocken. Wichtig ist, dass der zu prüfende Layer, Dependencies mittels Setter-Methode oder Konstruktor injiziert bekommt. Damit die Mockito Funktionalitäten verwendet werden können, muss ein statischer Import von ”org.mockito.Mockito.*” vorgenommen werden.

```
import static org.mockito.Mockito.*;
```

Nachfolgend ein Test, welcher überprüft, ob der Controller die Daten richtig ins JSON-Modell abfüllt.

```
public class MeasureStationControllerV0Test {
        MeasureStationControllerV0 controller;
        MeasureStation ms = new MeasureStation();
        MeasureStationConfiguration config = new MeasureStationConfiguration();
        DataServer dataserver = new DataServer();

        final static String DS_NAME = "test";
        final static String DS_URL = "http://www.test.ch";

        @Before
        public void setUp() throws Exception {
                ms.setId(1l);
                ms.setStationName("muster");

                List<DataServer> list = new LinkedList<DataServer>();
                dataserver.setId(1l);
                dataserver.setName(DS_NAME);
                dataserver.setUrl(DS_URL);
                list.add(dataserver);
                config.setDataServers(list);
                ms.setConfiguration(config);
                MeasureStationService measureStationService = mock(
                    MeasureStationService.class);
                when(measureStationService.loadMeasureStaionById(1l)).thenReturn(ms);
                MeasurementService measurementService = mock(MeasurementService.class
                    );
                controller = new MeasureStationControllerV0(measureStationService,
                    measurementService);
        }

        @Test
        public void testGetConfigForStation() {
                JSONStationConfig conf = controller.getConfigForStation(1l);
                JSONDataServer ds = conf.getDataServers().get(0);
                Assert.assertEquals(DS_NAME, ds.getName());
                Assert.assertEquals(DS_URL, ds.getUrl());
        }

}
```

11.2 Messstation

Testaufbau

Um zu überprüfen ob die entwickelte Messstation korrekte Werte liefert, wurde ein Testaufbau eingerichtet. Dabei kamen zwei Raspberry Pi Messstationen, sowie eine Gfs Referenzstation zum Einsatz. Die Aufnahmen der Vergleichsmessungen wurden im Studienarbeits-Zimmer der HSR durchgeführt. In dem zwei Raspberry Pi Messstationen betrieben wurden, stellten wir sicher das die Resultate validiert werden können. So konnten beispielsweise Differenzen erkannt werden, welche von einer fehlerhaften Konfiguration stammten. Der Messaufbau ist in Abbildung 11.2 ersichtlich.

Abbildung 11.2: Testmessungen im Studienarbeits-Zimmer

Vergleich Raspberry Pi Messstation mit Gfs Referenzgerät

Der erste Test vergleicht die erfassten Werte der neuen Raspberry Pi Messstationen, mit dem Referenzgerät von Gfs. In Abbildung 11.3 sehen Sie eine Gegenüberstellung der zwei Messkurven über einen Zeitraum von vier Stunden. Es ist zu beachten, dass es sich um gruppierte Werte handelt, da für die Auflösung zuviele Messpunkte vorhanden sind. Verkleinert man den Bereich auf einige Minuten, sieht man die tatsächlichen Werte wie dies in Abbildung 11.4 ersichtlich ist. Der Vergleich der Messkurven zeigt, die erfassten Lärmpegel bei niedrigem sowie gleichbleibendem Lärmpegel gut übereinstimmt. Bei sich schnell änderndem Lärmpegel ist zu erkennen, dass die Kurve des Referenzgerätes (Gfs) weniger steil abfällt als die der Raspberry Pi Messstation.

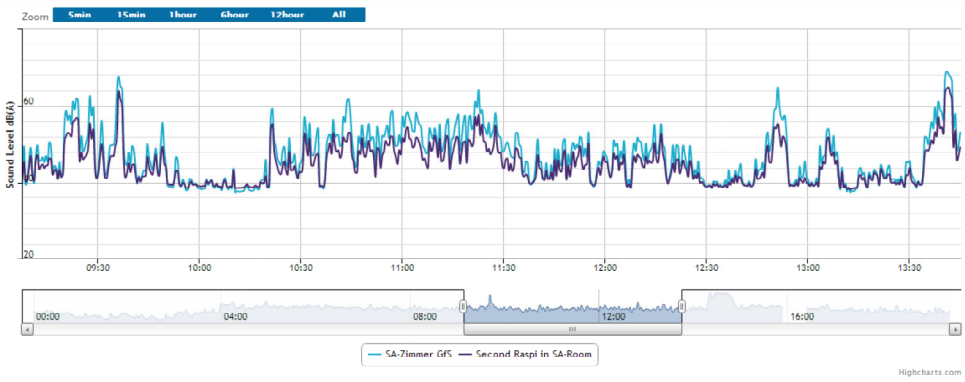


Abbildung 11.3: Testmessungen Gfs und Raspberry Pi - Zeitbereich vier Stunden

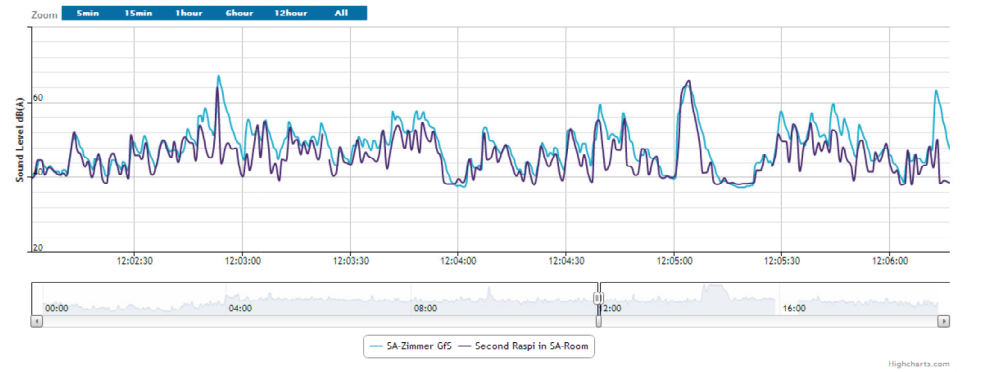


Abbildung 11.4: Testmessungen Gfs und Raspberry Pi Zeitbereich vier Minuten

Vergleich zweier Raspberry Pi Messstation

Als zweiter Test zwei gleiche Raspberry Pi Messstationen miteinander verglichen, um festzustellen ob diese dieselben Werte liefern. Die Kurven verlaufen sehr ähnlich. Es ist eine konstante Verschiebung der beiden Messkurven feststellbar. Diese Verschiebung kann durch eine Kalibrierung des Messmodul korrigiert werden.

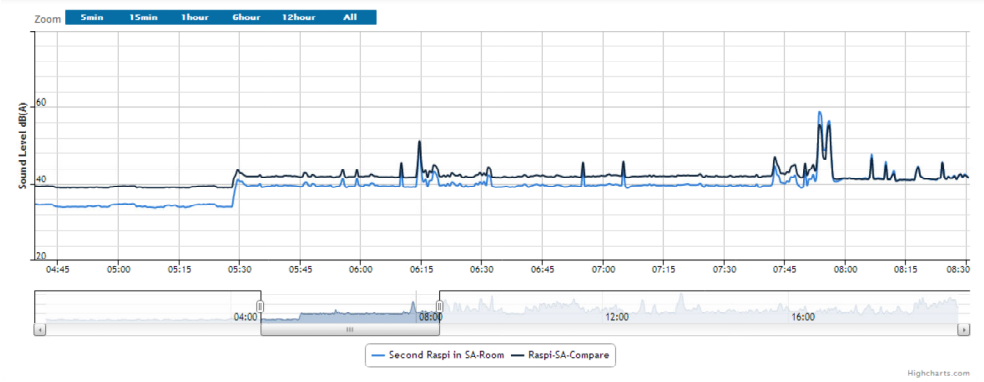


Abbildung 11.5: Testmessung über 4 Stunden

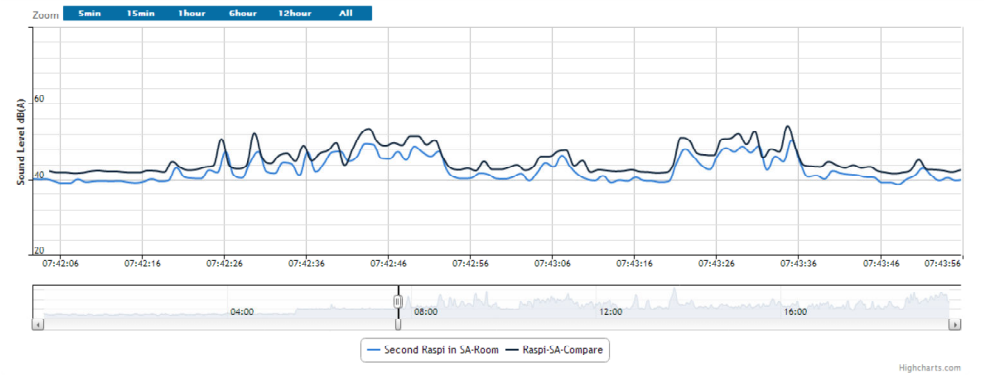


Abbildung 11.6: Testmessung über 2 Minuten

12 Schlussfolgerung

12.1 Zusammenfassung

Vorstudie

In den ersten Wochen der Studienarbeit, recherchierten wir welche Möglichkeiten es gibt, Lärmmessungen mit dem Raspberry Pi durchzuführen.

Dabei legten wir den Fokus zuerst auf die Auswertung über den Mikrofoneingang. Beim Raspberry Pi existiert kein solcher, weshalb wir die Lösung mit einer USB-Soundkarte prüften. Als Problematisch stellte sich dabei heraus, dass jedes Mikrofon andere Eigenschaften bei der Tonaufnahme besitzt und die USB-Schnittstelle auch mit Raspian nicht problemlos funktioniert.

Prototyp

Der zweite Lösungsansatz war, mithilfe eines Analog-Digital-Wandlers die bestehenden ”modul bus” Lärmmesssensoren einzubinden.
Wir konnten mit diesen Komponenten eine preiswerte, stabile und robuste Lösung erarbeiten, welche den Anforderung entspricht. Dies wurde geprüft in dem ein Prototyp gebaut wurde.

Datenserver

Um die erfassten Lärmpegel persistent zu speichern wurde ein Datenserver entwickelt. Dafür haben wir bei der Software-Architektur an zukünftige Weiterentwicklungen gedacht.

Visualisierung

Beim Entwurf des User-Interfaces wurde der Adminbereich als Mockup entworfen. Dieser enthält die Verwaltung von Messtationen, Alarmen und Benutzern, zudem ermöglicht es die Konfiguration der Messtationen. Diese Entwürfe finden sich unter Anhang E.
Ein weiterer Teil der Arbeit war die Visualisierung der erfassten Lärmpegel. Da die Technologie vorgegeben war, verloren wir keine Zeit mit dem Vergleich verschiedener Frameworks. Die Einarbeitung in das Framework ”Highcharts” und dessen Dokumentation stellte sich als zeitaufwändig heraus.

Fazit

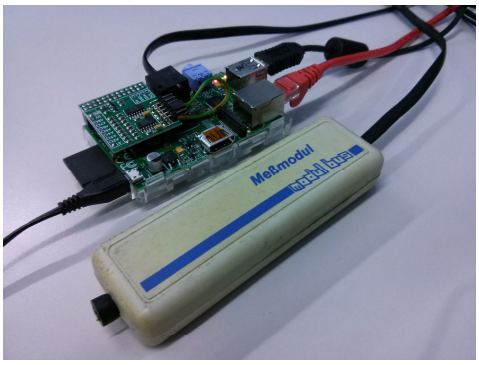


Abbildung 12.1: erste Raspberry Pi Messstation

Trotz der kurzen Implementierungsphase ist es uns gelungen ein Messsystem basierend auf dem Raspberry Pi zu entwickeln. Welches kostengünstiger in der Anschaffung und einfach in der Bedienung ist. Die Raspberry Pi-Messstation konnte so aufgebaut werden, dass mit einem serienproduzierten Gehäuse gearbeitet werden kann. Es muss minimal modifiziert werden. Das erste Raspberry Pi welches zum Einsatz kam, ist auf Abbildung 12.1 ersichtlich. Bei dieser Version musste die RCA-Buchse entfernt werden. Ebenfalls gelang es uns eine schlanke Visualisierung zu erstellen, die schneller Daten liefert als die bestehende[1]. Im Gegensatz zu DFLD zeigt die Einstiegsseite alle Stationen an. Der Benutzer kann sofort Lärmmessungen ansehen und braucht nicht durch ein Menü zu navigieren. Durch Verwendung von Responsive Design passt sich das Layout allen Bildschirmgrössen an.

12.2 Ausblick

Wird an der Applikation weiter entwickelt, so soll der Fokus auf dem Administrations– und Verwaltungsbereich gelegt werden. Die entsprechenden Mockups im Anhang E wurden bei den Besprechungen ausgearbeitet und können so implementiert werden. Die Serverarchitektur wurde so entworfen das dieser Schritt ohne Probleme möglich ist.

Da die Software Open-Source und auf GitHub abgelegt ist, kann jeder diese Weiterentwickeln. Die Adressen sind folgende:

- Server: `https://github.com/ITA-HSR-hei/atm_server`
- Client: `https://github.com/ITA-HSR-hei/atm_client`

Die Verwaltung erfolgt durch die HSR und wird eventuell weiterentwickelt.

1. http://atm.cnlab.ch

Glossar

db(A)

Steht für einen mit A-Filter bewerteten Schalldruck. 3

Dependency Injection

Als Dependency Injection bezeichnet man das Einfügen von Abhängigkeiten in Objekte. 25

DFLD

Deutscher Fluglärmdienst. 2, 29, 39

ECTS

Das European Credit Transfer and Accumulation System (ECTS) soll sicherstellen, dass die Leistungen von Studenten an Hochschulen des Europäischen Hochschulraumes vergleichbar und bei einem Wechsel von einer Hochschule zur anderen, auch grenzüberschreitend, anrechenbar sind. Dies ist möglich durch den Erwerb von Leistungspunkten (englisch credits), das sind Anrechnungseinheiten, die in der Hochschulausbildung durch Leistungsnachweise erworben werden. Diese Art der Bewertung von Leistungen an Hochschulen findet sich vorwiegend in den Bachelor- und Master-Studiengängen an Hochschulen. (1 Credit = 30 Arbeitsstunden). 58

Framework

Ein Framework ist ein Programmiergerüst, das in der Softwaretechnik, insbesondere im Rahmen der objektorientierten Softwareentwicklung sowie bei komponentenbasierten Entwicklungsansätzen, verwendet wird. Im allgemeineren Sinne und nicht als dezidiertes Softwareprodukt verstanden, bezeichnet man mit Framework auch einen Ordnungsrahmen. 25, 30, 32–34, 38, 58

Gantt-Diagramm

Ein Gantt-Diagramm oder Balkenplan ist ein nach dem Unternehmensberater Henry L. Gantt (1861–1919) benanntes Instrument des Projektmanagements, das die zeitliche Abfolge von Aktivitäten grafisch in Form von Balken auf einer Zeitachse darstellt. 58

GitHub

GitHub ist ein webbasierter Hosting-Dienst für Software-Entwicklungsprojekte. Er verwendet namensgebenderweise das Versionsverwaltungs-System Git. 12, 16, 17

Google Chrome

Google Chrome ist ein Webbrowser vom Unternehmen Google Inc. 27, 32

GPIO

Allzweckeingabe/-ausgabe (engl. GPIO - General Purpose Input/Output) ist ein allgemeiner Kontaktstift an einem integrierten Schaltkreis, dessen Verhalten, unabhängig, ob als Eingabe- oder Ausgabekontakt, durch logische Programmierung frei bestimmbar ist. 10, 21, 22

HTTP

Das Hypertext Transfer Protocol (HTTP, deutsch Hypertext-Übertragungsprotokoll) ist ein Protokoll zur Übertragung von Daten über ein Netzwerk. Es wird hauptsächlich eingesetzt, um Webseiten aus dem World Wide Web (WWW) in einen Webbrowser zu laden. 1, 17, 18, 23, 27

I2C

I2C, für englisch Inter-Integrated Circuit, ist ein von Philips Semiconductors (heute NXP Semiconductors) entwickelter serieller Datenbus. 19, 55

Jenkins

Jenkins ist ein erweiterbares, webbasiertes System zur kontinuierlichen Integration. 32

JSON

steht für JavaScript Object Notation. Ist human-readable und eine alternative zu XML, JSON überzeugt durch wenig Overhead, benötigt weniger CPU-Leistung und Arbeitsspeicher. 1, 12, 16–18, 23, 25, 27, 32, 34

microSD

microSD ist ein sehr kompaktes Flash-Speicherkartenformat, das elektrisch mit SD Memory Card identisch ist. 21

MySQL

MySQL ist eine relationale Open-Source Datenbank. 1, 12

Port

Ein Port ist der Teil einer Netzwerk-Adresse, der die Zuordnung von TCP- und UDP-Verbindungen und -Datenpaketen zu Server- und Client-Programmen durch Betriebssysteme bewirkt. Zu jeder Verbindung dieser beiden Protokolle gehören zwei Ports, je einer auf Seiten des Clients und des Servers. 1, 27

Raspberry Pi

Der Raspberry Pi ist ein kreditkartengroßer Einplatinen-Computer, der von der Raspberry Pi Foundation entwickelt wurde. 1, 10, 11, 16, 21, 23, 35, 36, 38, 39, 54, 71

Redmine

Redmine ist ein webbasiertes Projektmanagement-Tool. Es kann für Benutzer- und Projektverwaltung, Diskussionsforen, Wikis, zur Ticketverwaltung oder Dokumentenablage genutzt werden. 58

Responsive Design

Beim Responsive Webdesign (im Deutschen auch responsives Webdesign) handelt es sich um einen gestalterischen und technischen Ansatz zur Erstellung anpassungsfähiger Websites. Daher wird anstelle von ”responsiv” häufig auch von einem ”adaptiven Layout” gesprochen. Der grafische Aufbau einer ”responsiven” Webseite erfolgt anhand der Anforderungen des jeweiligen Gerätes, mit dem die Seite betrachtet wird. Dies betrifft insbesondere die Anordnung und Darstellung einzelner Elemente, wie beispielsweise Navigationen, Seitenspalten und Texte. Technische Basis hierfür sind neuere Webstandards wie HTML5, CSS3 und JavaScript. 39

REST

steht für Representational State Transfer. 1, 16, 18, 25, 32

reverse-engineeren

Reverse Engineering (engl., bedeutet: umgekehrt entwickeln, rekonstruieren, Kürzel: RE), auch Nachkonstruktion, bezeichnet den Vorgang, aus einem bestehenden, fertigen System oder einem meistens industriell gefertigten Produkt durch Untersuchung der Strukturen, Zustände und Verhaltensweisen, die Konstruktionselemente zu extrahieren. Aus dem fertigen Objekt wird somit wieder ein Plan erstellt. Im Gegensatz zu einer funktionellen Nachempfindung, die ebenso auf Analysen nach dem Black-Box-Prinzip aufbauen kann, wird durch Reverse Engineering angestrebt, das vorliegende Objekt weitgehend exakt abzubilden. 10

Spring

Das Spring Framework (kurz Spring) ist ein quelloffenes Framework für die Java-Plattform. Ziel des Spring Frameworks ist es, die Entwicklung mit Java/Java EE zu vereinfachen und gute Programmierpraktiken zu fördern. Spring bietet mit einem breiten Spektrum an Funktionalität eine ganzheitliche Lösung zur Entwicklung von Anwendungen und deren Geschäftslogiken; dabei steht die Entkopplung der Applikationskomponenten im Vordergrund. Spring bietet das Zusammenspiel unterschiedlichster Plattformen und Tools, von Java EE-Servern über Persistenztools bis hin zur Web-Integration. 25, 58

URL

Ein Uniform Resource Locator (Abk. URL; englisch für einheitlicher Quellenanzeiger) identifiziert und lokalisiert eine Ressource wie z. B. eine Website über die zu verwendende Zugriffsmethode (z. B. das verwendete Netzwerkprotokoll wie HTTP oder FTP) und den Ort (engl. location) der Ressource in Computernetzwerken. 28

USB

Der Universal Serial Bus (USB) ist ein serielles Bussystem zur Verbindung eines Computers mit externen Geräten. Mit USB ausgestattete Geräte oder Speichermedien (USB-Speichersticks) können im laufenden Betrieb miteinander verbunden (Hot-Plugging) und angeschlossene Geräte sowie deren Eigenschaften automatisch erkannt werden. 11, 38

XML

Die Extensible Markup Language (engl. für „erweiterbare Auszeichnungssprache“), abgekürzt XML, ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien. XML wird u. a. für den plattform- und implementationsunabhängigen Austausch von Daten zwischen Computersystemen eingesetzt, insbesondere über das Internet. 27

Literaturverzeichnis

[Ges13] Gesellschaft für Sonder-EDV-Anlagen mbH. Schallpegel Monitor. http://www.gfs-hofheim.de/pdf/Schallpegel-Monitor_SPM483_de.pdf, 2013. [Online; accessed 28.05.2013.

[htt13] http://quick2wire.com/. quick2wire/quick2wire-python-api. https://github.com/quick2wire/quick2wire-python-api, 2013. [Online; accessed 28.05.2013.

[Jü09] Jürgen H. Maue. *0 Dezibel + 0 Dezibel = 3 Dezibel*. Erich Schmidt Verlag, 9 edition, 2009. Einführung in die Grundbegriffe und die quantitative Erfassung des Lärms.

[Joh13] Johan Haleby. rest-assured. http://code.google.com/p/rest-assured/, 2013. [Online; accessed 28.05.2013.

[Mic07] Michael Möser. *Technische Akkustik*. Springer, 7 edition, 2007.

[Sta04] Stahel, Tobias and Studer, Pascal. Lärmerfassung mit Pc-Soundkarte und Visualisierung im Internet. diploma thesis, Zürcher Hochschule Winterthur - ZHAW, Oktober 2004.

[UdK04] UdK Berlin Sengpiel. Formelblatt 'Zusammenhang der akustischen Grössen'. http://www.sengpielaudio.com/ZusammenhangDerAkustischenGroessen.pdf, 2004. [Online; accessed 05.03.2013.

[Wik13a] Wikipedia. Bel (Einheit). http://de.wikipedia.org/wiki/Dezibel, 2013. [Online; accessed 05.03.2013.

[Wik13b] Wikipedia. Frequenzbewertung. http://de.wikipedia.org/wiki/Frequenzbewertung, 2013. [Online; accessed 05.03.2013.

[Wik13c] Wikipedia. Glossareinträge. http://de.wikipedia.de, 2013. Glossareinträge stammen von Wikipedia.de [Online; accessed 05.03.2013.

[Wil10] Willi Nickel Brüel and Kjaer GmbH - Düsseldorf. Was ist ein Schallpegelmesser? http://www.bruelkjaer.de/Products/~/media/Germany/Produkte/SPM/Was%20ist%20ein%20Schallpegelmesser_Juni2010.ashx, 2010. [Online; accessed 05.03.2013.

Abbildungsverzeichnis

Tabellenverzeichnis

A Schnittstellenbeschreibung JSON

Die Schnittstelle ist unter der Service-URL zu erreichen. Die Anfragen gestalten sich wie folgt: **Service-Url/{domain}/{version}/{methodenname}**.

Keyword	Beschreibung
{Service-Url}	steht für die URL, in unserem Fall http://atmng.cnlab.ch/atm
{version}	entspricht der Schnittstellenversion und wird von 0 an inkrementiert.
{domain}	**public,user,admin** steht für die Rolle, welche nötig ist um auf die Inhalte zuzugreifen[1]
{methodenname}	entspricht dem Namen der jeweiligen Methode

A.0.1 ChartData

Hier finden sich die Schnittstellen-Beschriebe für den GUI-Teil.

/public/0/getStations

Request: GET http://atmng.cnlab.ch/public/0/getStations
Payload: -
Response: Liste aller Stationen
Payload:
Ein Array mit allen Stationen und deren ID's.

```
{
    "stationList": [
        {
            "id": 1,
            "name": "Maennendorf"
        },
        {
            "id": 2,
            "name": "Raspi49 SA-Zimmer"
        }
    ]
}
```

1. Implementiert ist nur public

/public/0/getChartData

Request: GET http://atmng.cnlab.ch/atm/public/0/getChartData
Payload: ?stationId=1&timestampFrom=1369526400000&timestampTo=1369612799999
Response: Messwerte im geforderten Bereich
Payload:
Anmerkung: Hier wurde das “data“ Array aus Darstellungsgründen verkürzt.


```
{
    "id": 1,
    "name": "Maennendorf",
    "pointInterval": 1000,
    "dateFrom": 1369526400000,
    "dateTo": 1369612799999,
    "data": [
        33,
        33.3,
        32.3,
        31.8,
        32.5,
        32.3,
        33.1,
        47.7,
        44.3,
        46.3,
        42.2,
        40.7,
        39.4,
        39.9,
        43.1
    ]
}
```


A.0.2 MeasureStation

Hier findet sich der Beschrieb für alle Kommunikationen welche für den Client-Teil implementiert wurden.

/public/0/receiveData

Request: POST http://atmng.cnlab.ch/atm/public/0/receiveData
Payload:


```
{
    "stationId": 1,
    "soundlevel": "523",
    "timestamp": 1369422420300
}
```


Response: 200 OK oder einen 404/500-Fehler mit der entsprechenden Begründung.

Payload: -

/public/0/getIdFromMac

Request: GET http://atmng.cnlab.ch/atm/public/0/getIdFromMac
Payload: ?macAddress=B8:27:EB:FF:AA:E3
Response: Den Stationsname und die entsprechende Id oder nur eine Id, welche durch diesen Request angelegt wurde. Hier wird das REST-Prinzip trotzdem nicht Verletzt weil der Request keinen Schaden anrichten kann.
Payload:

```
{
    "id": 1,
    "name": "Maennendorf"
}
```

/public/0/getConfigForId

Request: GET http://atmng.cnlab.ch/atm/public/0/getConfigForId
Payload: ?stationId=1
Response: Die Konfiguration[2] für entsprechende Station.
Payload:

```
{
    "dataServers": [
        {
            "id": 1,
            "name": "cnlabng",
            "url": "http://atmng.cnlab.ch/atm/receiveData"
        },
        {
            "id": 2,
            "name": "dfld",
            "url": "http://raspb.dfld.de/receiveData"
        }
    ]
}
```

2. nicht implementiert - nur Demowerte

B Bausatz und Kosten

B.1 Materialliste

In Abbildung B.1 sowie in der Tabelle B.1 sind alle für den Bau der Messstation benötigten Teile aufgeführt.

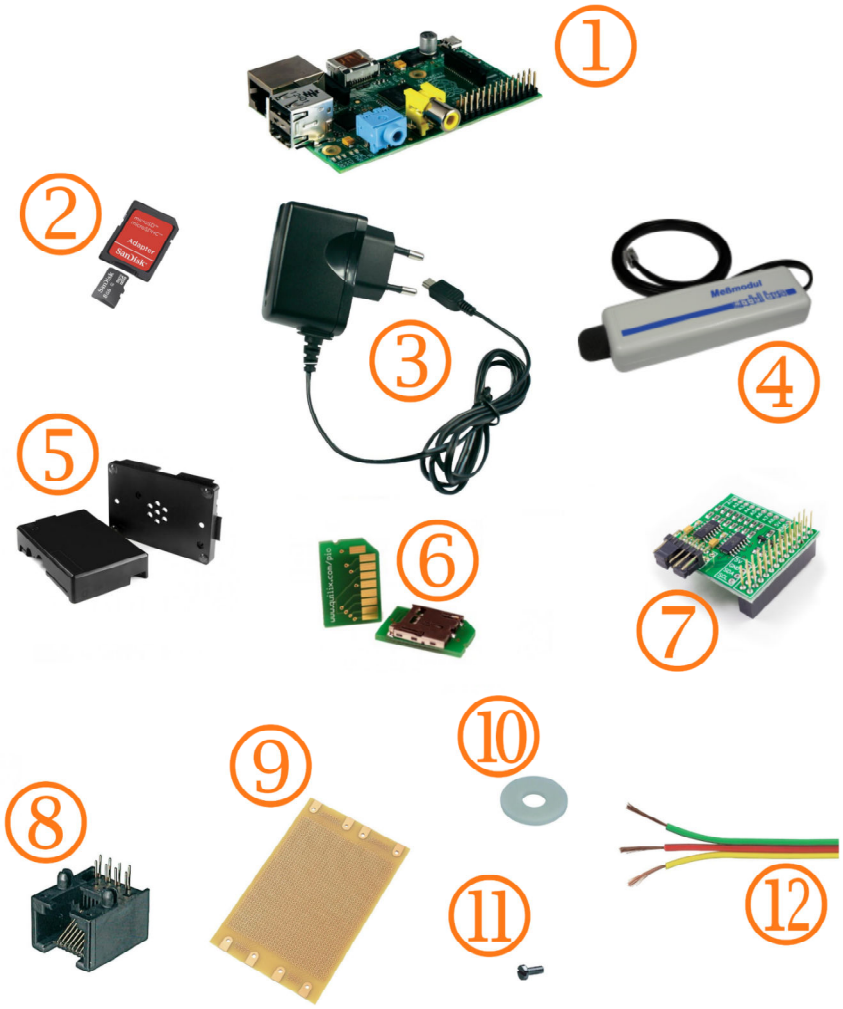


Abbildung B.1: Material

Position	Beschreibung	Preis	Weblink
1	Raspberry Pi Typ B	CHF 48.50	https://shop.klingler.net/product-details.php?productid=10001
2	Speicherkarte microSD card 8GB, Class 10	CHF 14.90	https://shop.klingler.net/product-details.php?productid=10045
3	Steckernetzteil 5V 1.2A	CHF 14.90	http://www.pi-shop.ch/steckernetzteil-5v-1-2a
4	Messmodul Lärm	€ 59.00	http://www.ak-modul-bus.de/cgi-bin/iboshop.cgi?showd200!0,625396181324355,MML
5	Raspberry Pi Case Stealth Black	£ 5.99	https://www.modmypi.com/raspberry-pi-cases/single-colour/raspberry-pi-case-black
6	Raspberry Pi microSD Card Adaptor	£ 7.49	https://www.modmypi.com/raspberry-pi-accessories/raspberry-pi-micro-sd-card-adaptor
7	ADC Pi V2 - Raspberry Pi Analogue to Digital converter	£ 17.99	http://www.abelectronics.co.uk/products/3/Raspberry%20Pi/17/ADC-Pi-V2—Raspberry-Pi-Analogue-to-Digital-converter
8	Einbaubuchse Buchse RJ11	CHF 1.05	http://www.conrad.ch/ce/de/product/716172/Modulare-Einbaubuchse-Buchse-Einbau-Pole-6P6C-A-20041LP-Schwarz-Assmann-WSW-Inhalt-1-St
9	Europlatine-Punktraster SU529010 160 mm x 100 mm	CHF 5.25	http://www.conrad.ch/ce/de/product/530632/Europlatine-Punktraster-SU529010-L-x-B-160-mm-x-100-mm-Rastermass-254-mm-Hartpapier-mit-Cu-Auflage
10	TOOLCRAFT Kunststoff-Unterlegscheiben M3	CHF 2.65	http://www.conrad.ch/ce/de/product/800281/TOOLCRAFT-Kunststoff-Unterlegscheiben-DIN-125-aus-PA-66-natur-125-Polyamid-PA-66-natur-M3-10-St
11	Zylinderkopf- schrauben M2.5 16 mm	CHF 0.10	http://www.conrad.ch/ce/de/product/888681/Zylinderkopfschrauben-DIN-84-16-mm-Stahl-verzinkt-Guete-48-M25-1-St
12	Flachbandkabel 3 x 0.14 mm² Gelb, Rot, Grün 5 m	CHF 8.95	http://www.conrad.ch/ce/de/product/605805/Flachbandkabel-3-x-014-mm-Gelb-Rot-Gruen-5-m

Tabelle B.1: Materialliste

B.2 Adapter Print

Für den kompletten Bau der Messstationen sind ca. 2 Stunden einzuplanen. Die Arbeiten setzen handwerkliches können und eine Werkstatt mit Werkzeugen voraus.

Die in nachfolgendem Beschreibung erwähnten Positionen beziehen sich auf die Tabelle B.1 sowie auf die Abbildung B.1.

Zu Beginn muss die Printplatine (Position 9) auf das entsprechende Format zugeschnitten werden, sowie die benötigten Löcher gebohrt werden, wie in Abbildung B.2 ersichtlich. Anschliessend kann die RJ11 Buchse (Position 8) auf den Print gelötet werden. Auf der anderen Seite des Prints wird die das Kabel (Position 12) angelötet. Dabei ist auf die gekennzeichnete Aderfarben zu achten. Im Anschluss ist die andere Seite des Kabels mit dem Analog-Digital-Wandler (Position 7) zu verlöten. Die Aderfarben des Kabels ist in Abbildung B.2 zu sehen.

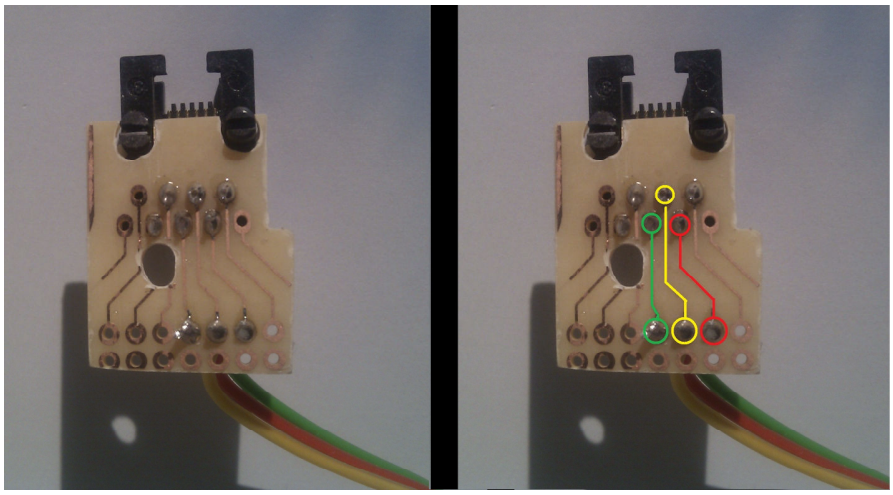


Abbildung B.2: Adapter Print - Ansicht Leiterbahnen

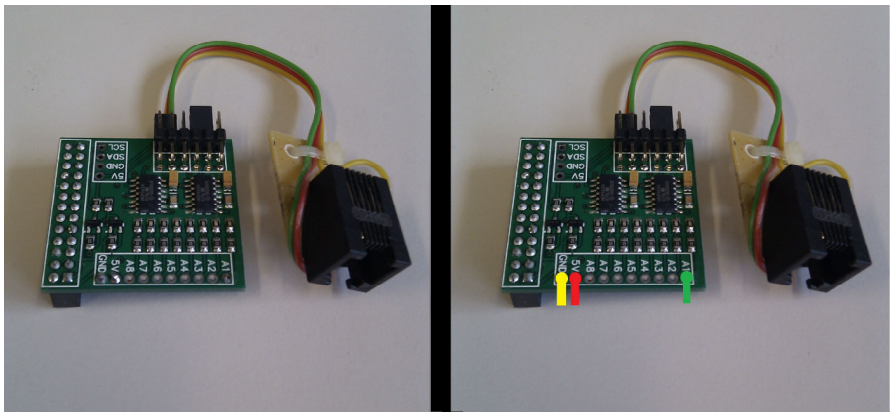


Abbildung B.3:

B.3 Montage

Der Print wird mit der M2.5 Zylinderkopfschraube (Position 11) festgeschraubt. Damit der Print genügend Abstand hat sind die Plastik-Unterlagsscheiben (Position 10) zu verwenden. Der montierten Print ist in Abbildung B.3 zu sehen. Damit das Gehäuse (Position 5) geschlossen werden kann, muss eine entsprechenden Öffnung ausgefeilt werden.

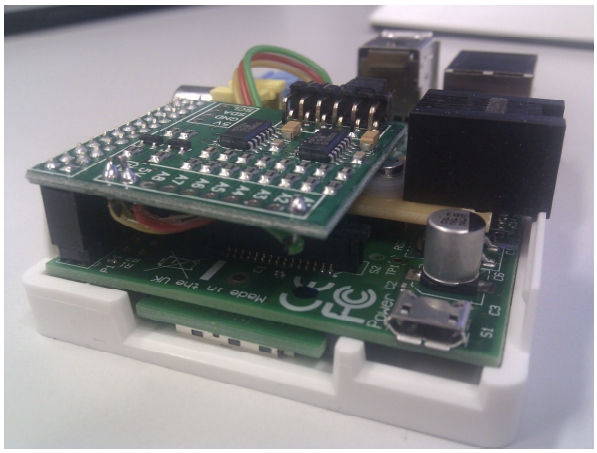


Abbildung B.4: Messstation nach Montage des Print sowie des Analog Digitalwandler

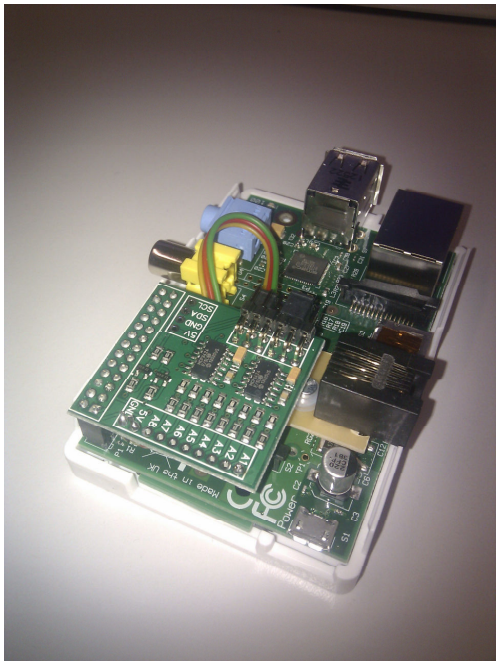


Abbildung B.5: Messstation - Ansicht von oben

C Installationsanleitung

Diese Installationsanleitung beschreibt alle Schritte, die notwendig sind, um die Lärmpegel-Software zu verwenden. Es sind grundlegende Kenntnisse im Umgang mit Linux Betriebssystemen notwendig.

C.1 Grundinstallation Raspberry Pi

Installation Image

Zur Installation des Betriebssystems Raspbian sind folgende Schritte durchzuführen.

1. Herunterladen des aktuellen Raspbian “wheezy” Image von der offiziellen Webseite. http://www.raspberrypi.org/downloads
2. Installation des Image auf der SD Karte mit der Software Win32DiskImager. Wählen Sie dazu das Image sowie das Laufwerk der Speicherkarte aus und klicken auf die Schaltfläche Write.

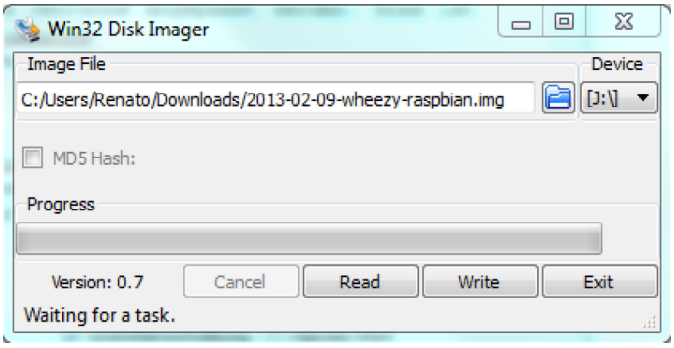


Abbildung C.1: Win32DiskImager

Raspi-Config

1. Nach dem ersten Start des Raspberry Pi wird die Raspi-Config Seite angezeigt. Dort muss die Funktion ”expand_rootfs” ausgeführt werden. Somit wird die ganze SD-Karte als Speicher verwendet. Die Änderungen werden beim nächsten Neustart ausgeführt.

2. Anschliessend muss das Keyboard Layout auf die Schweizer Tastatur umgestellt werden. Dies wird unter dem Punkt ”configure_keyboard” gemacht.

3. Der nächster Punkt ist die Anpassung der Zeitzone. Dies ist unter dem Eintrag ”change_timezone” durchzuführen.

4. Ebenfalls muss das Passwort angepasst werden, dies ist unter dem Punkt ”change_pass”.

Hinweis: Den Raspi-Config Dialog kann man zu jedem Zeitpunkt wieder mit dem Befehl ”raspi-config” anzeigen.

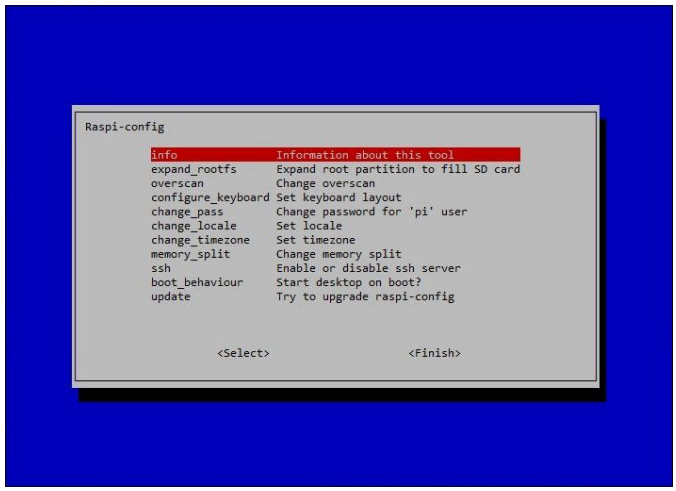


Abbildung C.2: raspi-config

Login auf dem Raspberry

Falls das Passwort nicht geändert wurde kann man sich mit dem Benutzernamen „pi“ und dem Passwort „raspberry“ über die Konsole einloggen. Damit Änderungen am System vorgenommen werden können, muss dies mit root rechten geschehen. Um sich als ”root” anzumelden, wird folgender Befehl eingegeben:

```
sudo -s
```

C.2 Software Installation

Update des Image

Für diesen Schritt wird eine aktive Internetverbindung benötigt. Zum Aktualisieren geben werden folgende Befehle auf der Konsole eingegeben.

```
sudo apt-get install update
sudo apt-get install upgrade
```

I2C Treiber aktivieren

Eine detaillierte Anleitung um den I2C Treiber zu aktivieren ist unter `http://quick2wire.com/articles/physical-python-part-1/` verfügbar

Die Treiber für den I2C Bus sind standardmässig deaktiviert. Um diese zu aktivieren muss die Konfigurationsdatei “raspi-blacklist.conf“ geändert werden. Dazu wird der Editor “nano“ verwendet:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

In der Datei “raspi-blacklist.conf“ müssen die zwei Zeilen auskommentieren wie dies in Abbildung C.3 ersichtlich ist. Zum Auskommentieren der Zeilen wird das # Zeichen verwenden.

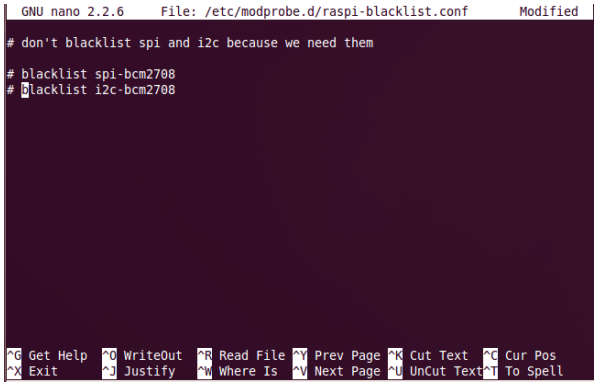


Abbildung C.3: raspi-blacklist.conf Konfiguration

Anschliessend wird in der Datei /etc/modules eine Zeile mit dem Text I2C-dev eingefügt. Dies ist in Abbildung C.4 ersichtlich.

```
sudo nano /etc/modules
```

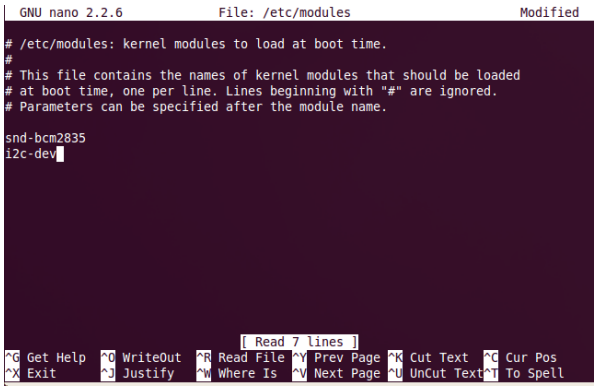


Abbildung C.4: I2C - aktivieren

Damit die Änderungen übernommen werden muss das System neu startet. Hierzu dient der Befehl:

```
sudo reboot
```

Um zu überprüfen ob der Treiber korrekt aktiviert wurde, gibt es ein Tool welches mit folgendem Befehl installieren kann:

```
sudo apt-get install i2c-tools
```

Um das Tool zu starten und zu überprüfen ob der Analog-Digital-Wandler erkannt wurde wird folgender Befehl ausgeführt:

```
sudo i2cdetect -y 1
```

Pip installieren

Pip ist ein Package-Manager welcher es ermöglicht Software Module die in Python geschrieben sind zu installieren.

```
sudo apt-get install python3-pip
```

Requests installieren

Für das versenden der Lärmpegel an den Datenserver verwenden wir die Python Library ”requests”. Die Library ermöglicht es einfach einen GET oder POST Request abzusetzen. Folgender Befehl führt die installation aus:

```
pip-3.2 install requests
```

Installation Git

```
sudo apt-get install git
```

C.3 SoundLevelMeter Software

Klonen des Github Repository

Um die SoundLevelMeter Software aus dem GitHub repository zu klonen muss folgender Befehl ausgeführt werden. Achten Sie dabei dass Sie sich im Ordner /home/pi befinden.

```
sudo git clone https://github.com/ITA-HSR-hei/atm_client.git
```

Ausführen des Installationsscript

Damit die Lärmpegelsoftware automatisch beim ”start” des Betriebssystems gestartet wird und die automatischen Updates täglich durchgeführt werden, muss eine einmalige Installation dieser Komponenten erfolgen. Dafür wurde ein Installationsscript vorhanden welches mit nachfolgendem Befehl ausgeführt wird:

```
sudo /home/pi/soundLevelMeter/administration/installSoundLevelMeter.sh
```

Monitoring

Das Logfile kann mit folgendem Befehl angeschaut werden. Die Option ”-f” bedeutet das es fortlaufend aktualisiert wird, so sind immer die neuesten Einträge ersichtlich.

```
tail -f /home/pi/soundLevelMeter/logs/soundLevelMeter.log
```

D Zeitplanung

Um das Projekt zu planen und die Stunden zu verwalten, wurde Redmine verwendet. Mit dem Gantt-Diagramm (Abbildung D.1) ist immer ersichtlich wie der Projektstand ist und wie weiter geplant werden kann. Dabei schätzen wir für jeden Task die Zeit, so können wir nun ein Fazit ziehen.

Abbildung D.1: Gantt Diagramm

Die Stundenauswertung ist in Abbildung D.2 ersichtlich. Der Balken ”soll” steht dabei für die Arbeitszeit, welche gemäss ECTS-Punkten pro Woche aufgewendet werden sollte: 17.3 h[1]. Dieser Wert ist im Diagramm als Referenz eingezeichnet, weil unsere Arbeitsabschnitte nicht immer gleich lang dauerten.
Interne Besprechungen, Sitzungen mit dem Betreuer und Zeitaufwand für Administratives werden nicht im Diagramm angezeigt. Das Total der visualisierten Stunden beträgt 477.5. Gut erkennbar ist, dass wir bei den Implementierungsphasen die Zeit falsch geschätzt haben. Dies, weil wir das Framework Spring nicht gut kannten und den Umgang nicht so schnell erlernten wie es geplant war.

Problematisch war, dass wir nicht den selben Stundenplan hatten und an verschiedenen Tagen arbeiteten, so blieb nur ein halber Tag pro Woche an dem wir zusammen arbeiten konnten. Nicht in die Planung einbezogen wurde der Zeitaufwand um sich mit den neuen Technologien[2] vertraut zu machen und diese produktiv anzuwenden.
Bei den Meetings ging die Vor- und Nachtbearbeitungszeit vergessen und sie dauerten meistens länger als geplant.

1. 8 ECTS * 30 Stunden / 14 Semesterwochen = 17.1 h
2. Spring, Python, Maven

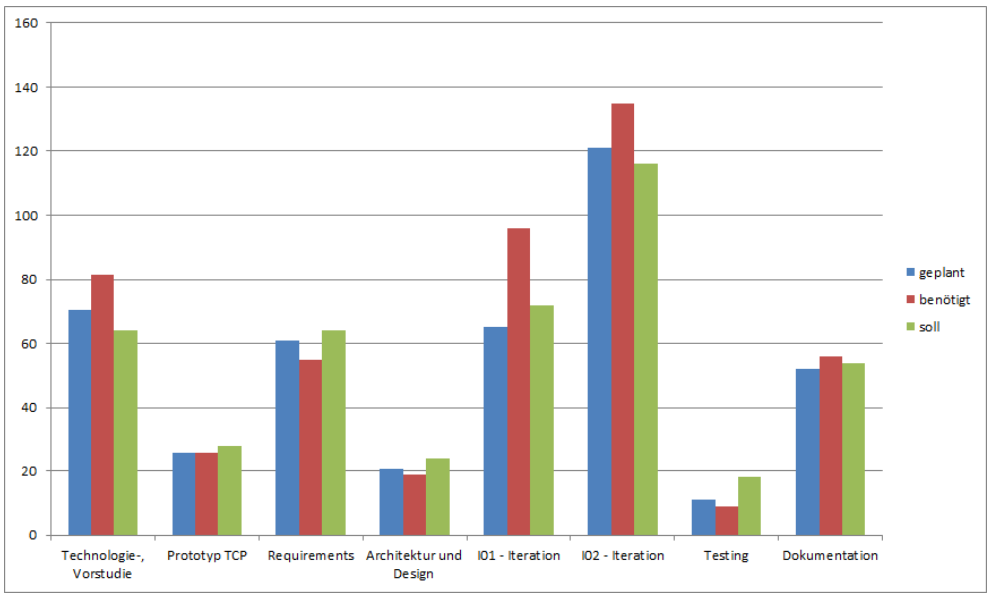


Abbildung D.2: Auswertung der Zeiterfassung

GUI - Administrationsbereich

In diesem Kapitel sind die während Besprechungen entstandenen Ansichten festgehalten.

Abbildung E.1: Administrationsbereich - Dashboard

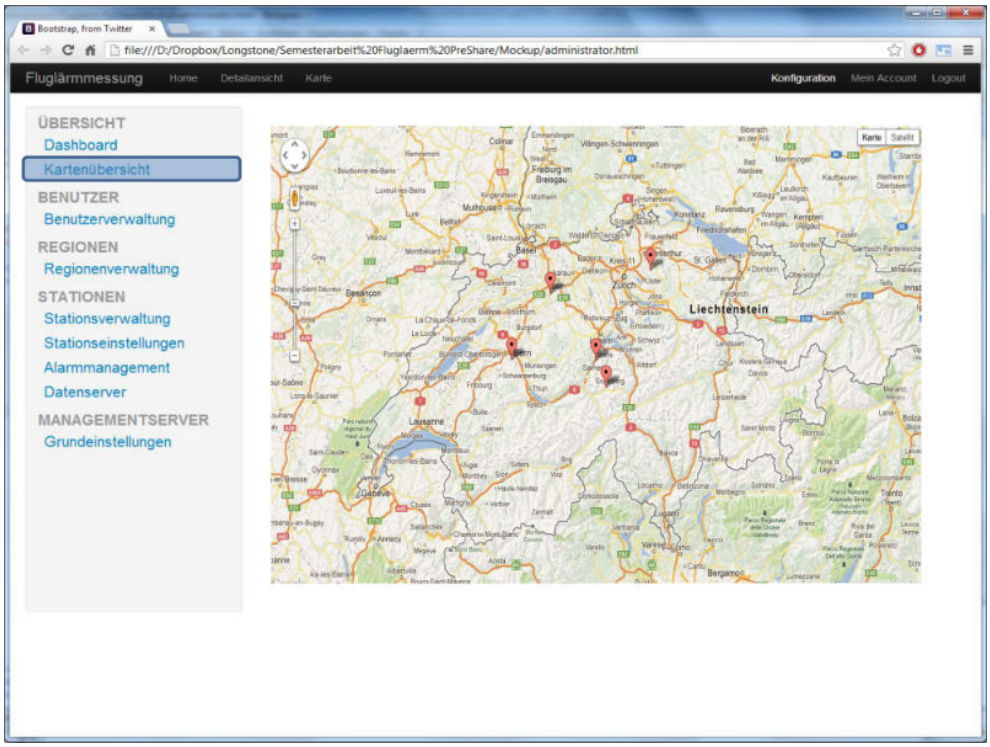


Abbildung E.2: Administrationsbereich - Kartenübersicht

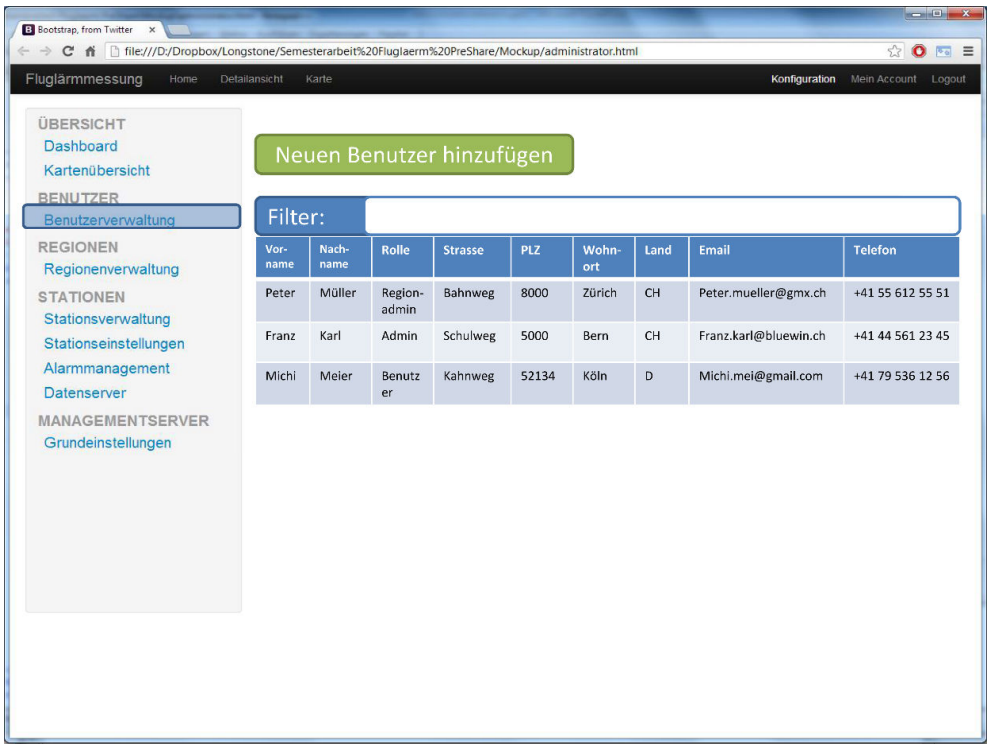


Abbildung E.3: Administrationsbereich - Benutzerverwaltung

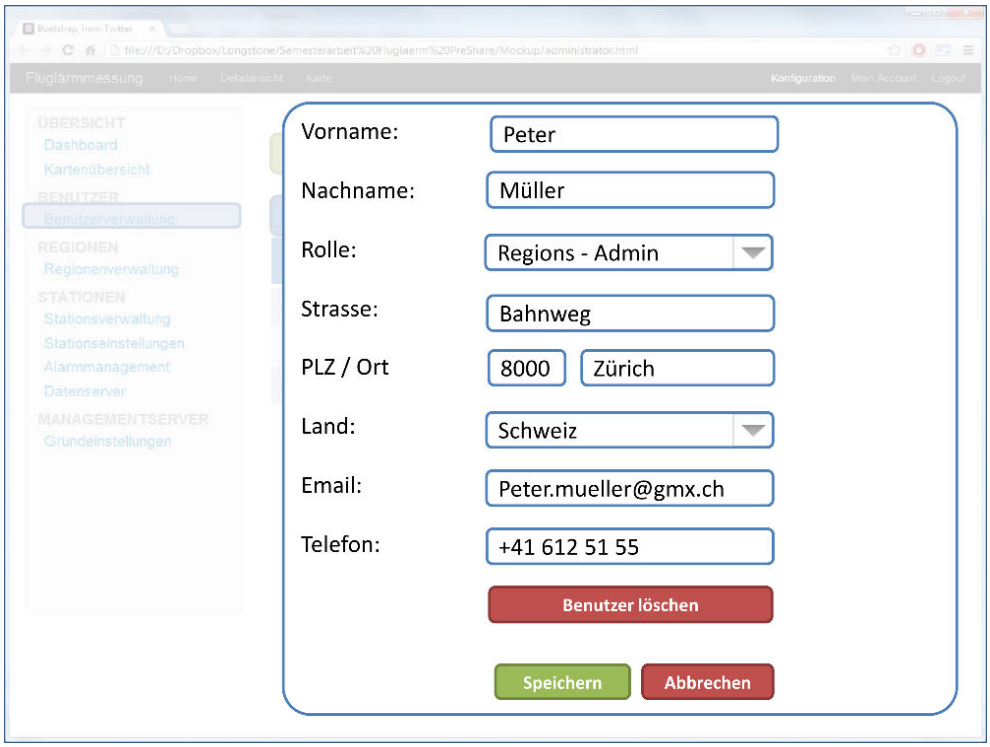


Abbildung E.4: Administrationsbereich - Benutzereditor

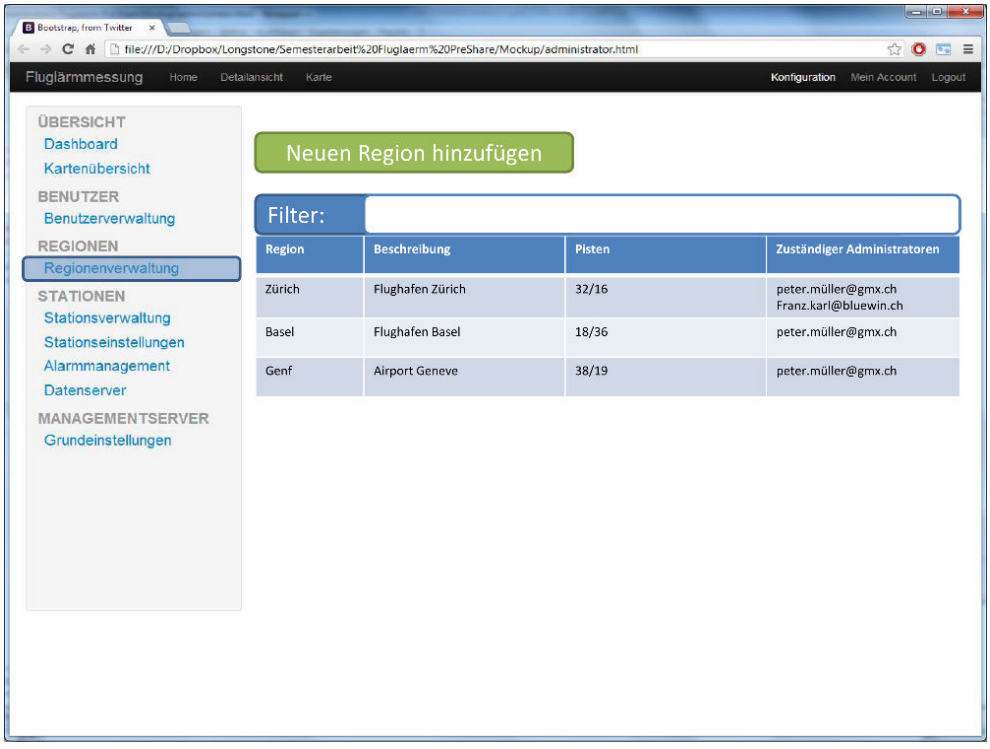


Abbildung E.5: Administrationsbereich - Regionen verwalten

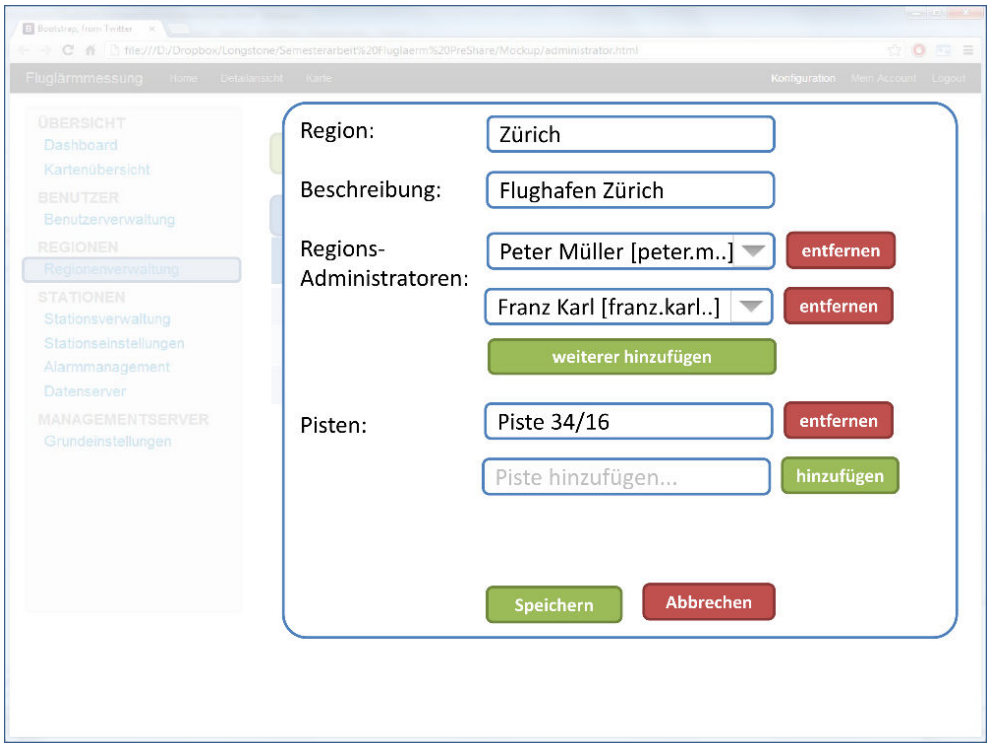


Abbildung E.6: Administrationsbereich - Region bearbeiten

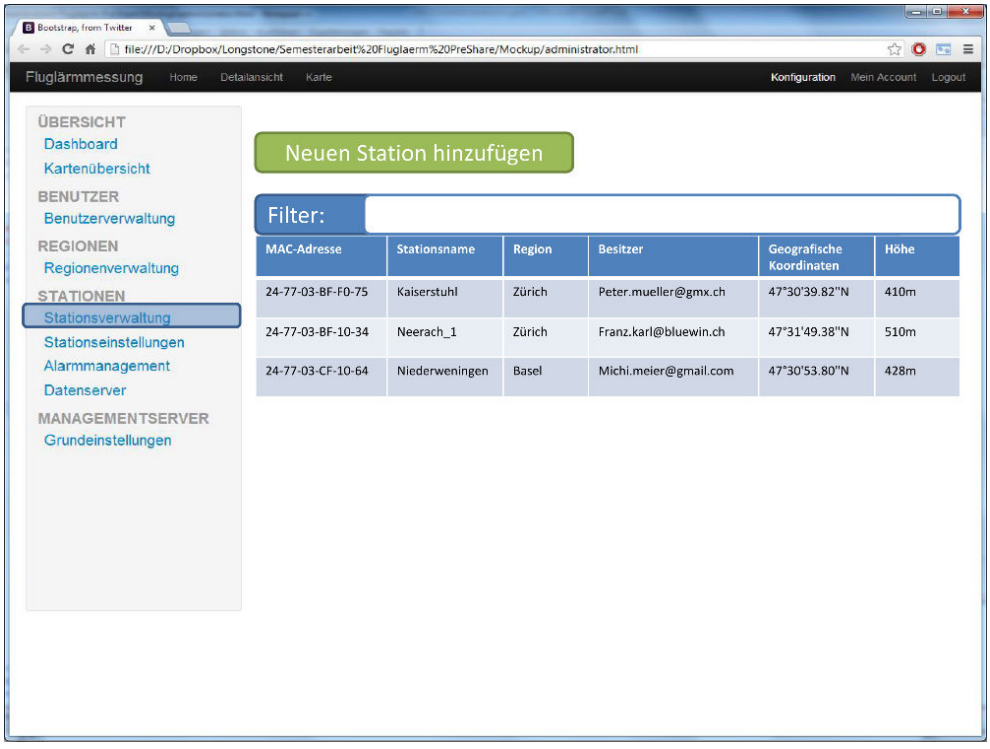


Abbildung E.7: Administrationsbereich - Stationen verwalten

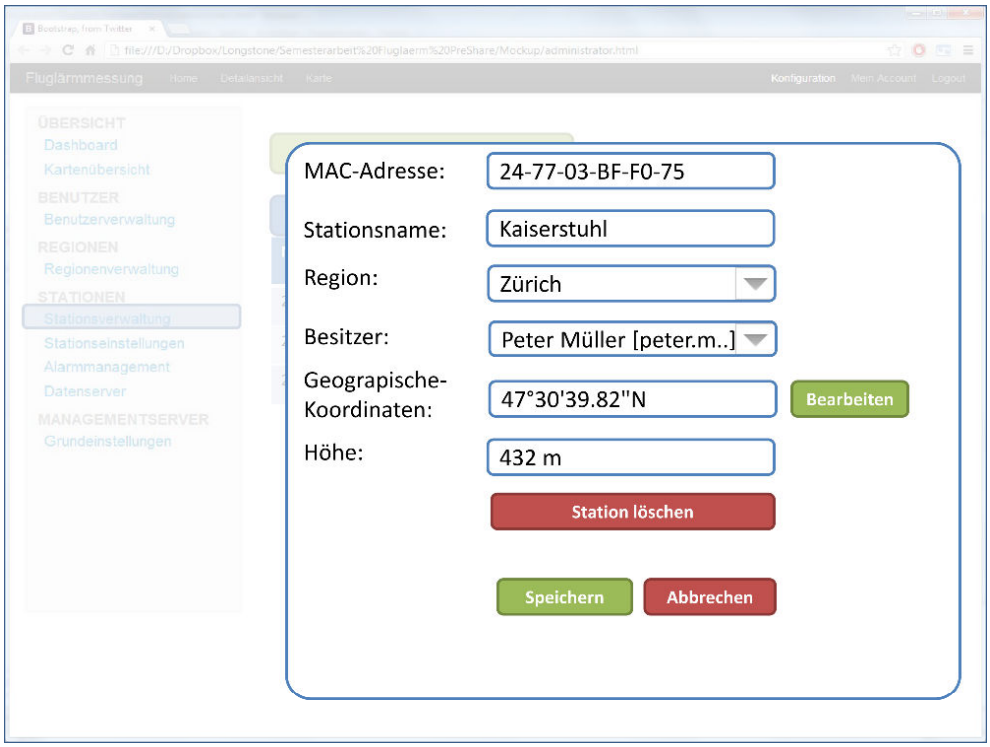


Abbildung E.8: Administrationsbereich - Station bearbeiten

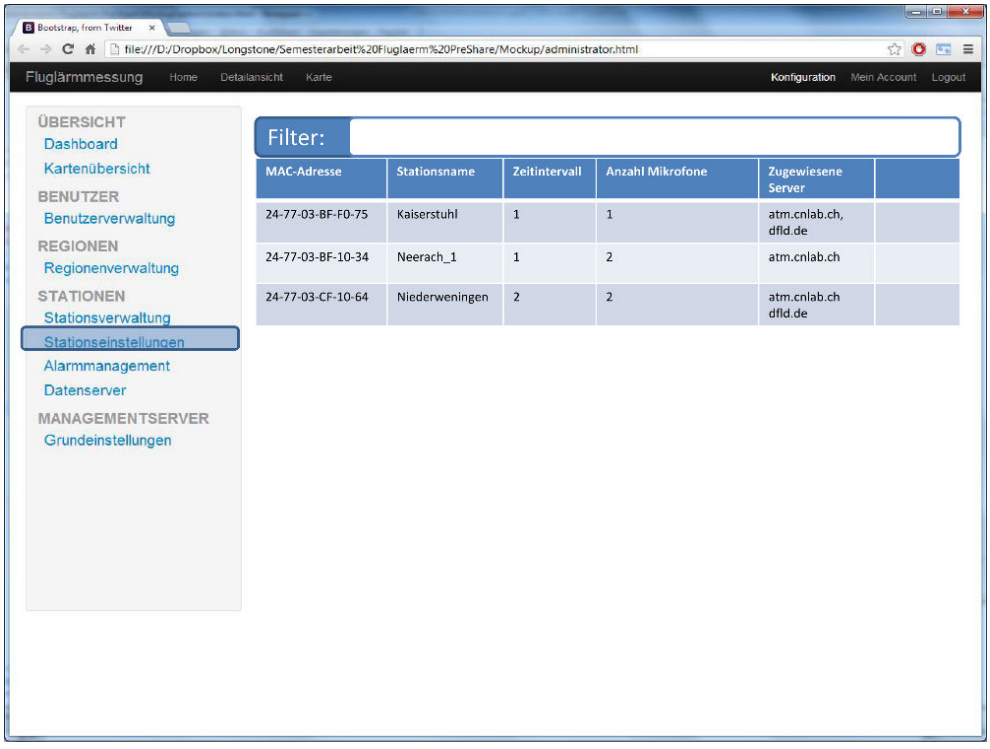


Abbildung E.9: Administrationsbereich - Stationseinstellungen verwalten

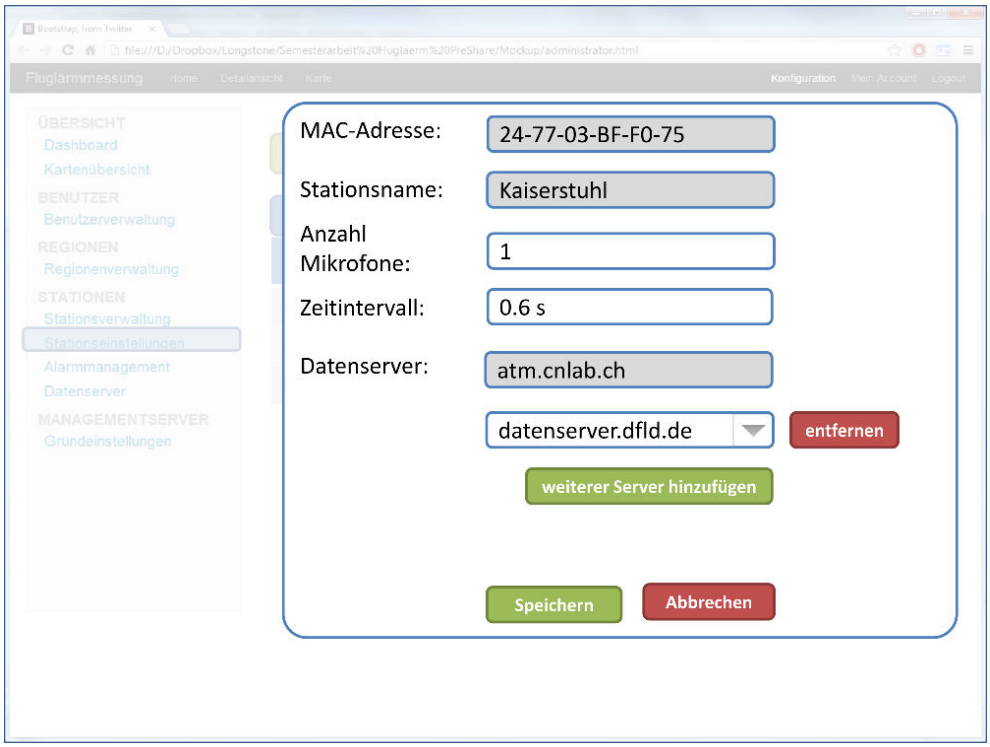


Abbildung E.10: Administrationsbereich - Stationseinstellung bearbeiten

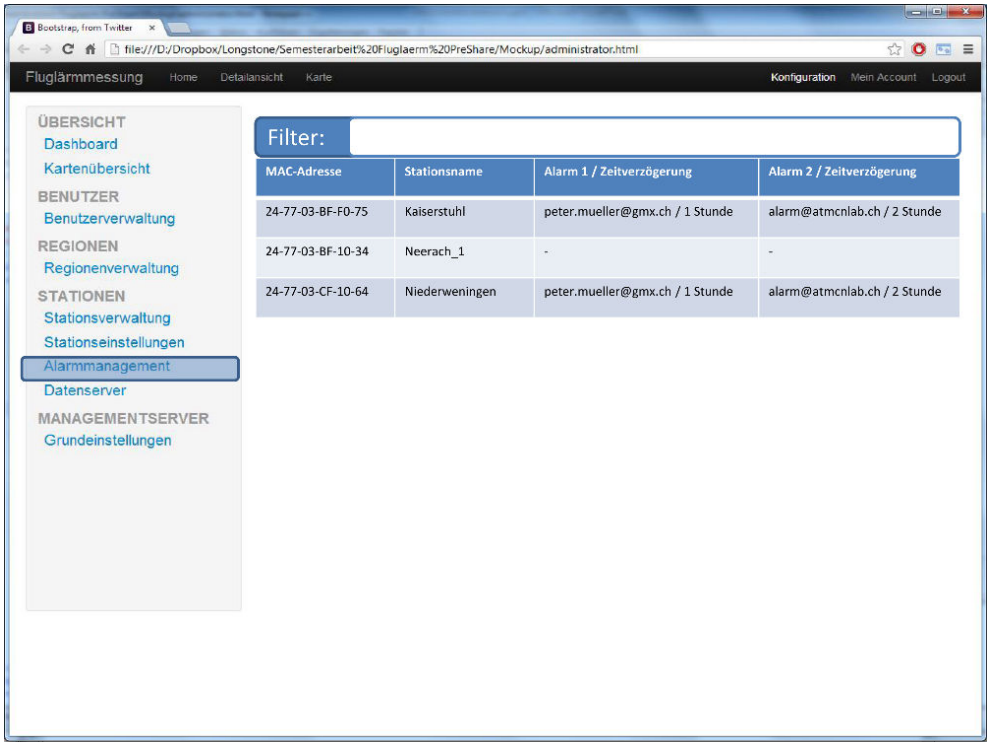


Abbildung E.11: Administrationsbereich - Alarme verwalten

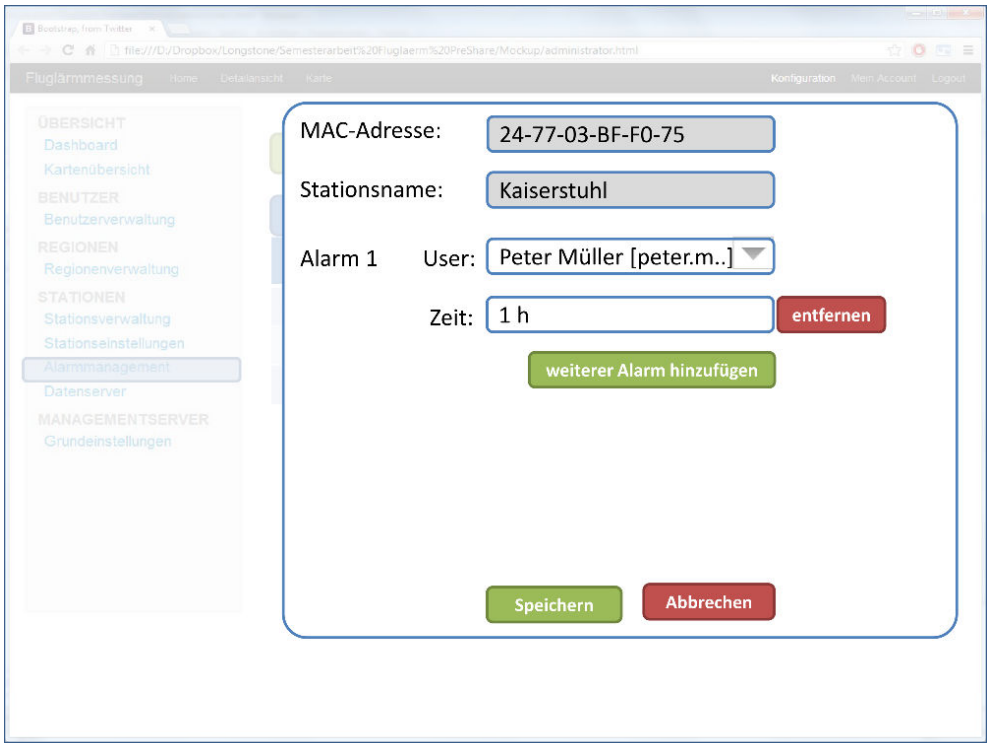


Abbildung E.12: Administrationsbereich - Alarm bearbeiten

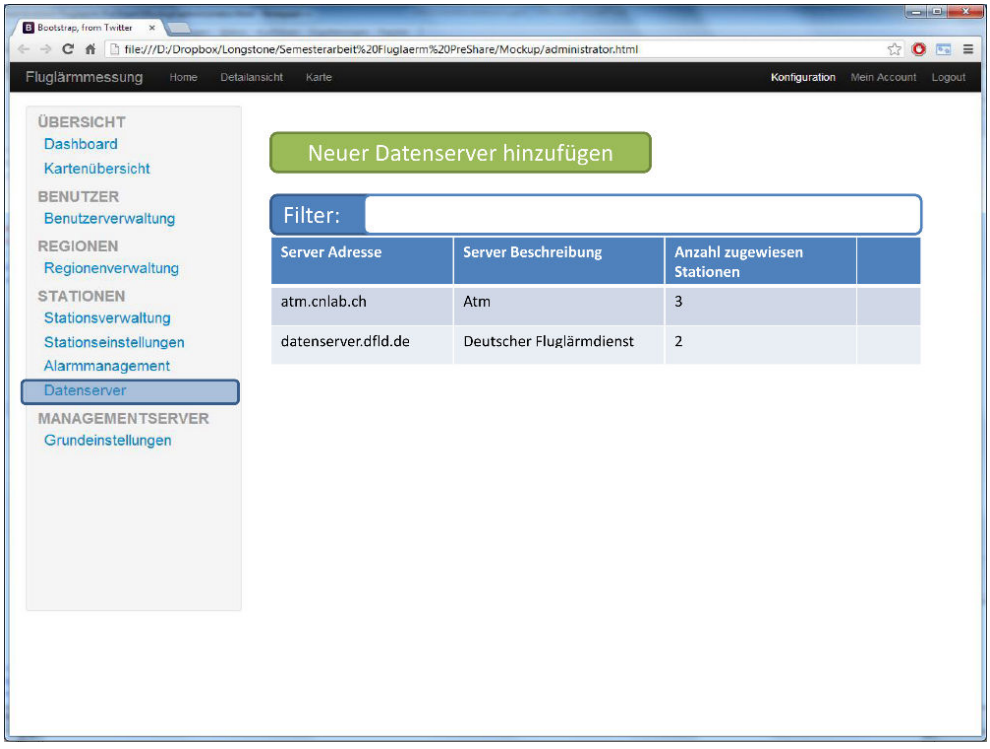


Abbildung E.13: Administrationsbereich - Datenserververwaltung

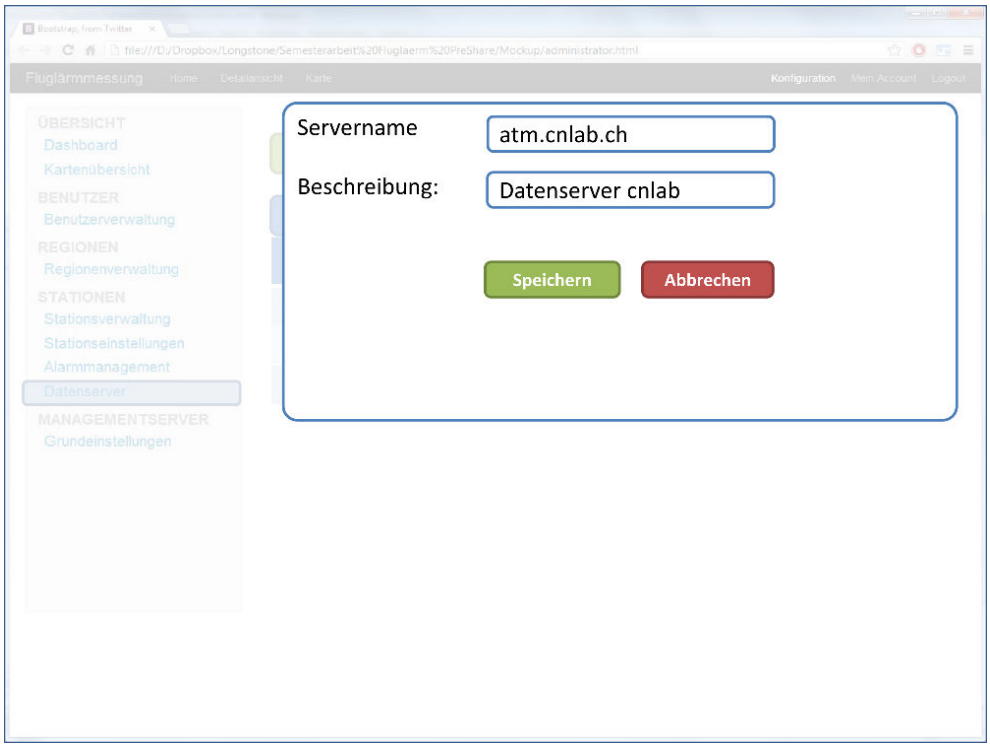


Abbildung E.14: Administrationsbereich - Datenserver bearbeiten

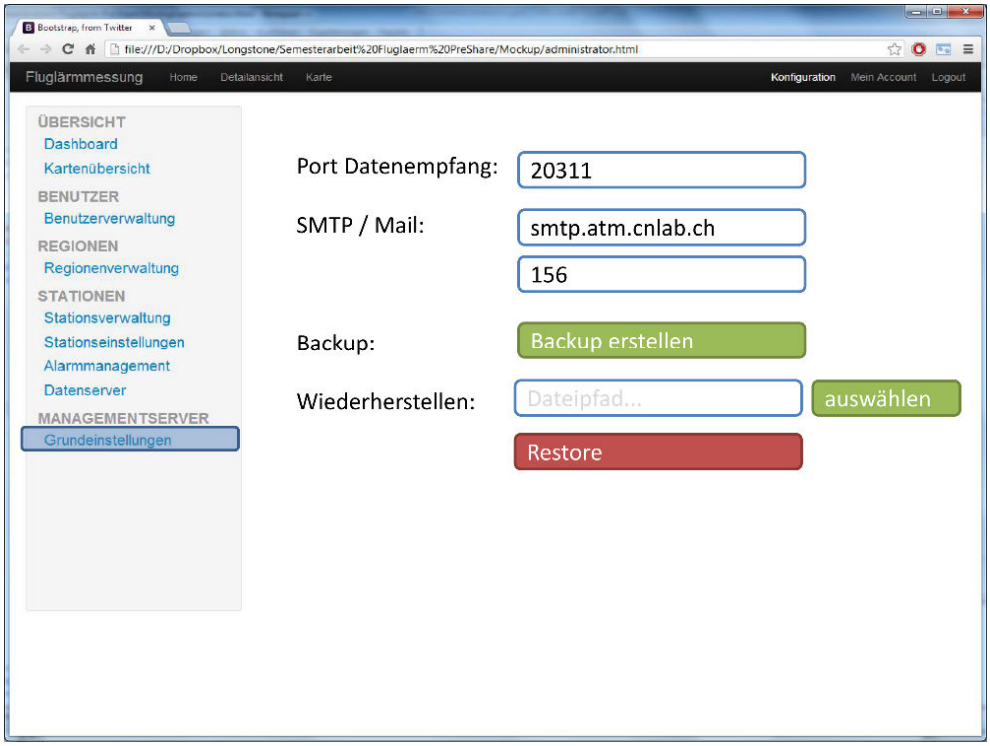
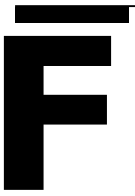


Abbildung E.15: Administrationsbereich - Email, Backup und Restore



Libraries und Frameworks

Datenserver

Im pom.xml sind alle referenzierten Libraries und Frameworks vermerkt. Zu beachten: zum Teil werden weitere Libraries referenziert.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance"
       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0_http://maven.apache.org
           /xsd/maven-4.0.0.xsd">
       <modelVersion>4.0.0</modelVersion>
       <groupId>ch.cnlab.atm</groupId>
       <artifactId>atm-webapp</artifactId>
       <version>0.0.1-SNAPSHOT</version>
       <packaging>war</packaging>


       <properties>
               <java.version>1.6</java.version>
               <spring.version>3.2.2.RELEASE</spring.version>
               <springsecurity.version>3.1.3.RELEASE</springsecurity.version>
               <hibernate.version>4.2.0.Final</hibernate.version>
               <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
       </properties>
       <build>
               <directory>target</directory>
               <finalName>${project.artifactId}-${project.version}</finalName>
               <testOutputDirectory>target/test-classes</testOutputDirectory>
               <sourceDirectory>src/main/java</sourceDirectory>
               <testSourceDirectory>src/test/java</testSourceDirectory>
               <resources>
                       <resource>
                               <directory>src/main/resources</directory>
                       </resource>
               </resources>
               <testResources>
                       <testResource>
                               <directory>src/test/resources</directory>
                       </testResource>
               </testResources>
               <plugins>
                       <plugin>
                               <groupId>org.apache.maven.plugins</groupId>
                               <artifactId>maven-compiler-plugin</artifactId>
                               <version>3.1</version>
                               <configuration>
                                       <source>${java.version}</source>
                                       <target>${java.version}</target>
                               </configuration>
                       </plugin>

                       <plugin>
                               <groupId>org.jacoco</groupId>
                               <artifactId>jacoco-maven-plugin</artifactId>
                               <version>0.6.2.201302030002</version>
                               <executions>
```

```
                                        <execution>
                                                <id>jacoco-initialize</id>
                                                <goals>
                                                        <goal>prepare-agent</goal>
                                                </goals>
                                        </execution>
                                        <execution>
                                                <id>jacoco-site</id>
                                                <phase>package</phase>
                                                <goals>
                                                        <goal>report</goal>
                                                </goals>
                                        </execution>
                                </executions>


                        </plugin>


                </plugins>
        </build>

        <dependencies>

                <!-- Spring Framework -->
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-context</artifactId>
                        <version>${spring.version}</version>
                </dependency>

                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-webmvc</artifactId>
                        <version>${spring.version}</version>
                </dependency>

                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-tx</artifactId>
                        <version>${spring.version}</version>
                </dependency>

                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-orm</artifactId>
                        <version>${spring.version}</version>
                </dependency>

                <dependency>
                        <groupId>org.springframework.security</groupId>
                        <artifactId>spring-security-core</artifactId>
                        <version>${springsecurity.version}</version>
                </dependency>
                <dependency>
                        <groupId>org.springframework.security</groupId>
                        <artifactId>spring-security-web</artifactId>
                        <version>${springsecurity.version}</version>
                </dependency>
                <dependency>
                        <groupId>org.springframework.security</groupId>
                        <artifactId>spring-security-config</artifactId>
                        <version>${springsecurity.version}</version>
                </dependency>

                <!-- For cronjob -->
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-core</artifactId>
                        <version>${spring.version}</version>
                </dependency>
```

```
		<!-- QuartzJobBean in spring-context-support.jar -->
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-context-support</artifactId>
			<version>${spring.version}</version>
		</dependency>


		<!-- Hibernate -->
		<dependency>
			<groupId>org.hibernate</groupId>
			<artifactId>hibernate-core</artifactId>
			<version>${hibernate.version}</version>
		</dependency>

		<dependency>
			<groupId>org.hibernate</groupId>
			<artifactId>hibernate-entitymanager</artifactId>
			<version>${hibernate.version}</version>
		</dependency>

		<dependency>
			<groupId>org.hibernate</groupId>
			<artifactId>hibernate-validator</artifactId>
			<version>${hibernate.version}</version>
		</dependency>


		<!-- MqSQL Connector -->
		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
			<version>5.1.10</version>
		</dependency>

		<!-- Junit -->
		<dependency>
			<groupId>junit</groupId>
			<artifactId>junit</artifactId>
			<version>4.8.2</version>
			<type>jar</type>
			<scope>test</scope>
		</dependency>

		<!-- Guava -->
		<dependency>
			<groupId>com.google.guava</groupId>
			<artifactId>guava</artifactId>
			<version>14.0.1</version>
		</dependency>

		<!-- Jackson JSON Mapper -->
		<dependency>
			<groupId>org.codehaus.jackson</groupId>
			<artifactId>jackson-mapper-asl</artifactId>
			<version>1.7.1</version>
		</dependency>

		<dependency>
			<groupId>commons-dbcp</groupId>
			<artifactId>commons-dbcp</artifactId>
			<version>1.2.2</version>
			<type>jar</type>
		</dependency>

		<!-- JSTL for JSP -->
		<dependency>
			<groupId>javax.servlet</groupId>
			<artifactId>jstl</artifactId>
			<version>1.2</version>
		</dependency>
```

```
                <!-- Rest testing -->
                <dependency>
                        <groupId>com.jayway.restassured</groupId>
                        <artifactId>rest-assured</artifactId>
                        <version>1.8.0</version>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>com.jayway.restassured</groupId>
                        <artifactId>json-path</artifactId>
                        <version>1.8.0</version>
                </dependency>
                <!-- mockito, test mocking -->
                <dependency>
                        <groupId>org.mockito</groupId>
                        <artifactId>mockito-all</artifactId>
                        <version>1.9.5</version>
                </dependency>

        </dependencies>
        <url>http://atmng.cnlab.ch:8080/atm-webapp</url>
        <name>ATMNG</name>
        <scm>
                <url>https://github.com/ITA-HSR-hei/atm_server</url>
                <connection>git</connection>
        </scm>
        <ciManagement>
                <system>jenkins</system>
                <url>http://atmng.cnlab.ch/jenkins</url>
        </ciManagement>
</project>
```

Raspberry Pi

Name	Version	Webseite
Python	3.2	http://www.python.org/download/releases/3.2/
quick2wire	-	https://github.com/quick2wire/quick2wire-python-api
requests	1.2.3	http://docs.python-requests.org/en/latest/

Tabelle F.1: Libraries - Raspberry Pi

Web-Visualisierung

Highstocks ist ein Framework für die Visualisierung von Werten in Diagrammen. Bootstrap ermöglicht das Responsive Design und bei jQuery, DatePicker und Cookie-Management handelt es sich um JavaScript Libraries.

Name	Version	Webseite
Highstocks	1.3.1	`http://www.highcharts.com/`
Bootstrap	2.3.1	`http://twitter.github.io/bootstrap/`
jQuery	1.9.0	`http://jquery.com/`
Google Maps	3.12	`https://developers.google.com/maps/`
DatePicker	Version 2012	`http://tarruda.github.io/bootstrap-datetimepicker/`
Cookie-Management	1.3.1	`https://github.com/carhartl/jquery-cookie`

Tabelle F.2: Frameworks - Visualisierung

G Features

Hier sind Features beschrieben, welche in der Software-Architektur Phase besprochen und vorbereitete wurden.

G.1 Messstation

Ziel Server

Die Clientsoftware soll vom Datenserver eine Liste mit Zielservern erhalten. Diesen Servern Sendet der Client die Messdaten im gewünschten Format.

Administration

Administration des Raspberry Pi muss über die Software “Groundskeeper Willie“[1] möglich sein

G.2 Webplattform

Darstellung von Mittelwert über Zeitperiode

Die Mittelwertfunktionen sollen weiter ausgebaut werden. Zusätzlich zum 1-Stunden-Mittelwert sollen auch andere Mittelwertspannen folgen.

Integration der Bilder

Die an den Datenserver übermittelten Bilder werden angezeigt.

Datenexport

Die Fluglärmdaten eines Charts können als .csv exportiert werden.

Login und Registrierung

Stationsbesitzer können sich auf der Website registrieren und einloggen.

G.3 Stationsmanagement

Neue Messstation hinzufügen / entfernen

Neue Messstationen hinzufügen oder entfernen.

Verwalten Benutzer

Benutzerdaten wie Name, Email, Telefonnummer und Adresse.

1. ist eine Applikation der cnlab ag

Verwalten Messstation

Erfassen und Bearbeiten der Daten, welche für die Verwaltung der Messstation notwendig sind. Stationsname, Besitzer, Geographische Koordinaten, Höhe, Alarmadresse

Kalibrierung

Das Datum der letzten Kalibrierung kann eingetragen werden. Nach bestimmten Zeitabständen wird per Email eine Neukalibrierung vorgeschlagen.

G.4 Datenserver

Alarmmanagement

Konfiguration ob Alarm gemeldet wird und an wen der Alarm gemeldet wird. Alarmeskalation: wird der Alarm nicht behoben, wird auf der nächst höheren Stufe alarmiert.

Backup und Restore

Backup und Restore sind ohne grossen Zeitaufwand aus dem Stationsmanagement heraus möglich.