Audit 3

Entwicklung eines Systems zur Überwachung und Automatisierung von Aquaponikanlagen

Anne Germund und Verena Heissbach

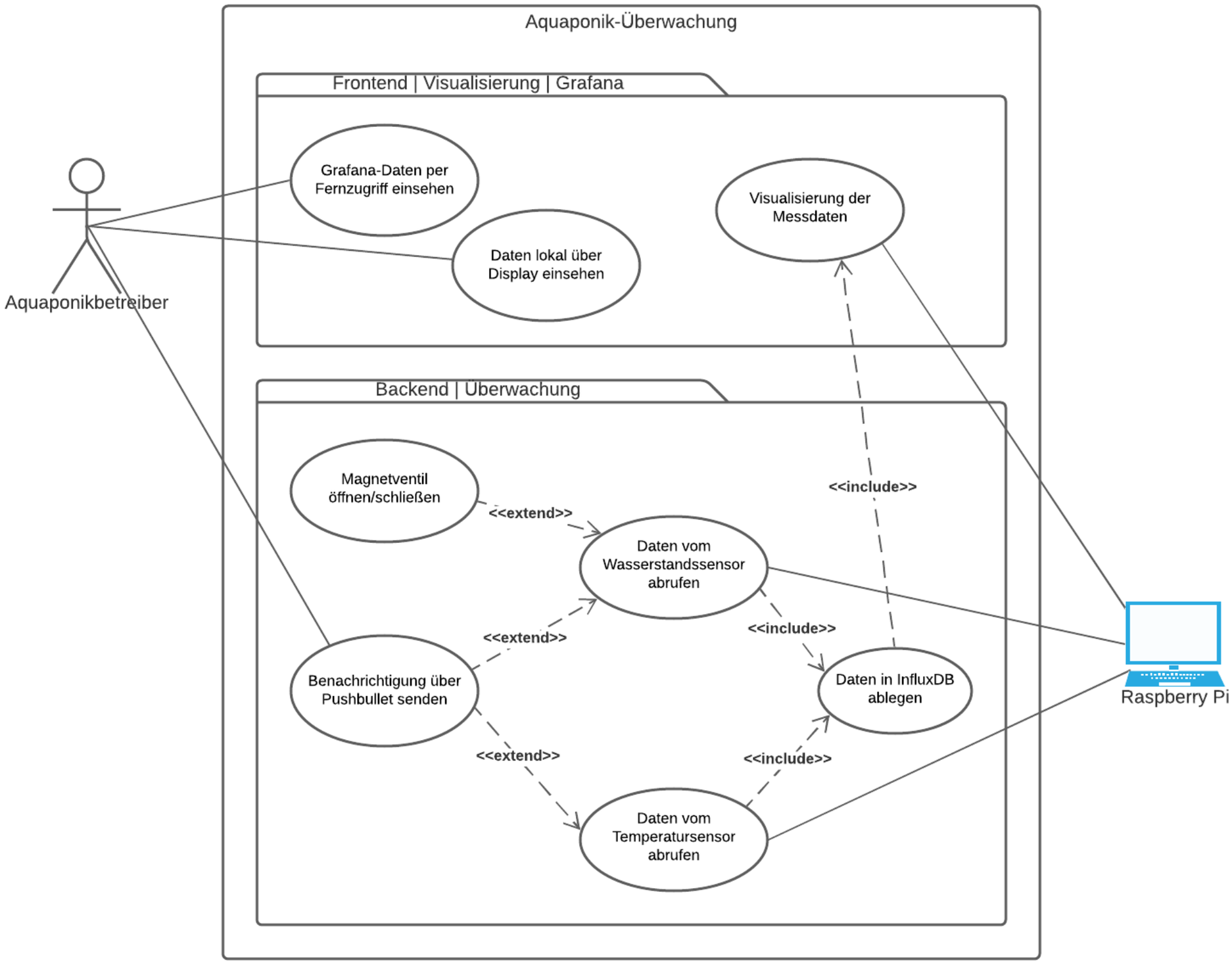
Feedback aus Audit 2 und Open Spaces

1. Architekturdiagramm für Audit 4
2. Weiterer Use Case: Visualisierung über ein am Raspberry Pi angeschlossenes Display
3. Überlegungen zum Online Datenzugriff (SSH, eigener Webserver, DNS)
4. Welches Datenbanksystem wird verwendet und warum?
5. Welche Visualisierungstools werden verwendet und warum?
6. Warum wird in Python programmiert?

Aktivitätsdiagramme und Use Cases

Beschreibung der Anwendungslogik und Funktionsmodellierung

Anwendungsfalldiagramm



Use Case #1

use case Grafana-Daten per Fernzugriff einsehen

actors Aquaponikbetreiber

trigger Der Aquaponikbetreiber möchte die Daten auf dem Grafana-Dashboard per Fernzugriff einsehen.

precondition Verbindung zu lokalem Netz besteht UND Überwachung ist aktiv

main flow

1. Der Betreiber gibt in der Browsersuchleiste die IP-Adresse des Raspberry Pi sowie den zu Grafana gehörigen Port ein.
2. Der Betreiber loggt sich mit seinen Zugangsdaten bei Grafana ein.
3. Der Betreiber wählt das anzuzeigende Dashboard aus.

postcondition Der Betreiber kann die aufgezeichneten Sensordaten einsehen.

exceptional flow

2.a Der Betreiber kennt seine Login-Daten nicht.

3.a Es werden keine aktuellen Daten angezeigt.

postcondition Kein Einsehen der Daten möglich

end Grafana-Daten per Fernzugriff einsehen

Use Case #2

use Case Daten lokal über Display einsehen

actors Aquaponikbetreiber

trigger Der Aquaponkibetreiber möchte die vom System gemessenen Daten, lokal über ein Display einsehen.

precondition Es sind bereits Messungen vorhanden (Überwachung ist aktiv), das Display ist am Raspberry Pi angeschlossen und mit Grafana verknüpft. Zusätzlicher Grafana User ist vorhanden und eingeloggt, Dashboards wurden erstellt.

main flow

1. Der Aquaponikbetreiber geht zum Display
2. Er kann nun die gemessenen Daten über das Display sehen

alternative flow

2.a Der Betreiber sieht die Daten Remote ein (siehe Use Case Grafana-Daten per Fernzugriff einsehen).

2.b Messdaten über die Kommandozeile des Raspberry Pi anzeigen lassen.

postcondition Der Aquaponikbetreiber konnte die gewünschten Daten einsehen.

exceptional flow

2.b Das Display zeigt keine Daten bzw. die falschen Daten an.

postcondition Es können keine Daten eingesehen werden

end Daten lokal über Display einsehen

Use Case #3

use case Daten vom Wasserstandssensor abrufen

actors Raspberry Pi

trigger Das Skript zur Wasserstandskontrolle wird ausgeführt.

precondition Stromversorgung besteht UND Wasserstandssensor ist angeschlossen

main flow

1. Der Raspberry Pi liest über den Kanal des AD-Wandlers, an welchen der Sensor angeschlossen ist, zehnmal die digitalen Wasserstandsdaten aus und speichert sie in einer Variable.
2. Der Raspberry Pi berechnet den durchschnittlichen Wasserstandswert, indem der Variablenwert durch die Anzahl der Messungen geteilt wird.
3. Der Raspberry Pi legt den ermittelten Wert in der Influx-Datenbank ab (include „Daten in InfluxDB ablegen“).
4. Der Raspberry Pi prüft den ermittelten Wert auf die festgelegten Wasserstandsgrenzen.

postcondition Der aktuelle Wasserstand wurde in der Datenbank abgelegt.

exceptional flow Kein Datenabruf

1.a Der Raspberry Pi kann keine Daten auslesen.

exceptional flow Wasserstand zu niedrig

4.a Sollte der Wasserstand zu niedrig sein, startet der Raspberry Pi die Befüllung (extension point: Wasserstand zu niedrig).

4.b Der Raspberry Pi sendet eine Benachrichtigung über den zu niedrigen Wasserstand sowie die Befüllung an den Betreiber (extension point: Benachrichtigung).

postcondition Der Wasserstand wurde korrigiert und eine Benachrichtigung an den Betreiber der Anlage gesendet.

end Daten vom Wasserstandssensor abrufen

Use Case #4

use case Daten vom Temperatursensor abrufen

actors Raspberry Pi

trigger Das Skript zur Temeperaturkontrolle wird ausgeführt.

precondition Stromversorgung besteht UND Temperatursensor ist angeschlossen

main flow

1. Der Temperatursensor führt je 10 Messungen aus (diese 10 Messungen ergeben dann einen Temperaturwert). Diese werden an den Raspberry Pi gesendet.
2. Der Raspberry Pi berechnet die durchschnittliche Temperatur, indem der Variablenwert durch die Anzahl der Messungen geteilt wird.
3. Der Raspberry Pi legt den ermittelten Wert in der Influx-Datenbank ab (include „Daten in InfluxDB ablegen“).
4. Der Raspberry prüft den Messwert auf die definierten Grenzen.

postcondition Die aktuelle Temperatur wurde in der Datenbank abgelegt.

exceptional flow

1.a Der Temperatursensor konnte keine Messungen durchführen.

1.b Es konnten keine Messwerte an den Raspberry Pi nicht gesendet werden.

3.a Die Daten konnten nicht in der Datenbank gespeichert werden.

4.a Sollte der Messwert außerhalb der definierten Grenzen liegen wird der Nutzer über Pushbullet darüber benachrichtigt (extension point: Benachrichtigung)

postcondition Es können keine Messungen abgerufen werden

end Daten vom Temperatursensor abrufen

Use Case #5

use case Daten in InfluxDB ablegen

actors Raspberry Pi

trigger Es wurden aktuelle Sensordaten abgerufen.

precondition InfluxDB ist installiert UND es existiert eine Datenbank für die Aquaponik-Sensordaten

main flow

1. Der Raspberry Pi erstellt den Datenpunkt mit den zugehörigen Messwerten und Sensortyp im JSON-Format.
2. Der Raspberry Pi übergibt die Sensordaten im JSON-Format an den InfluxDB-Client.

postcondition Der Datenpunkt konnte erfolgreich in die Influx-Datenbank abgelegt werden.

exceptional flow Kein Zugang zur Datenbank möglich

2.a Im Skript sind die falschen Zugangsdaten zur Datenbank hinterlegt.

postcondition Der Datenpunkt konnte nicht in der Influx-Datenbank abgelegt werden.

end Daten in influxDB ablegen

Use Case #6

use case Visualisierung der Messdaten mittels Grafana

actors Raspberry Pi

trigger Grafana kann auf die gespeicherten Messdaten in der Datenbank zugreifen.

precondition Grafana ist installiert und mit der Datenbank verknüpft. Es befinden sich Messdaten in der Datenbank. Der Betreiber ist als User bei Grafana angmeldet und eingeloggt.

main flow

1. Grafana greift auf die Datenbank der Aquaponiknalage zu.
2. Der Betreiber wählt das anzuzeigende Dashboard aus.
3. Die Messdaten werden im gewählten Dashboard dargestellt.

postcondition Der Betreiber kann die aufgezeichneten Sensordaten einsehen.

alternative flow

3.a Messdaten über die Kommandozeile des Raspberry Pi anzeigen lassen.

postcondition Die Daten konnten dargestellt werden.

exceptional flow

1.a Grafana kann nicht auf die Messwerte der Datenbank zugreifen.

2.a Es wurde kein Dashboard zur Visualisierung ausgewählt.

3.b Es werden keine aktuellen Daten angezeigt.

postcondition Keine Darstellung der Daten mittels Grafana möglich.

end Visualisierung der Messdaten mittels Grafana

Use Case #7

use case Benachrichtigung über Pushbullet senden

actors Raspberry Pi

trigger Der Raspberry Pi hat einen kritischen Sensorwert festgestellt und hat ggf. das Magnetventil geöffnet.

precondition Pushbullet ist installiert UND der API-Key ist hinterlegt UND das Gerät des Betreibers ist Pushbullet registriert

main flow

1. Der Raspberry Pi sendet über die Pushbullet-API eine Benachrichtigung über den kritischen Temperaturwert an das Gerät des Betreibers.

alternative flow

1.a Der Raspberry Pi sendet über die Pushbullet-API eine Benachrichtigung über den kritischen Wasserstand und das Öffnen des Magnetventils.

1.b Sobald der Wasserstand wieder im Normalbereich ist, sendet der Raspberry Pi eine Entwarnung an das Gerät des Betreibers.

postcondition Der Betreiber erhält eine Benachrichtigung über die kritischen Messwerte sowie mögliche Aktivitäten des Systems.

exceptional flow

1.a Das Gerät kann nicht gefunden werden.

postcondition Es konnte keine Benachrichtigung gesendet werden.

end Benachrichtigung über Pushbullet senden

Use Case #8

use case Magnetventil öffnen/schließen

actors Raspberry Pi

trigger Es wurde ein zu niedriger Wasserstand festgestellt und der Raspberry Pi leitet die Befüllung ein.

precondition Relais sind angeschlossen

main flow

1. Der Raspberry Pi versorgt den GPIO-Pin, über welchen das Relais-Modul angeschlossen ist, mit Spannung, um das Ventil zu öffnen.

alternative flow

1a. Der Raspberry Pi entfernt die Spannung vom GPIO-Pin, über welchen das Relais-Modul angeschlossen ist, um das Ventil zu schließen.

postcondition Das Magnetventil wurde erfolgreich geöffnet bzw. geschlossen.

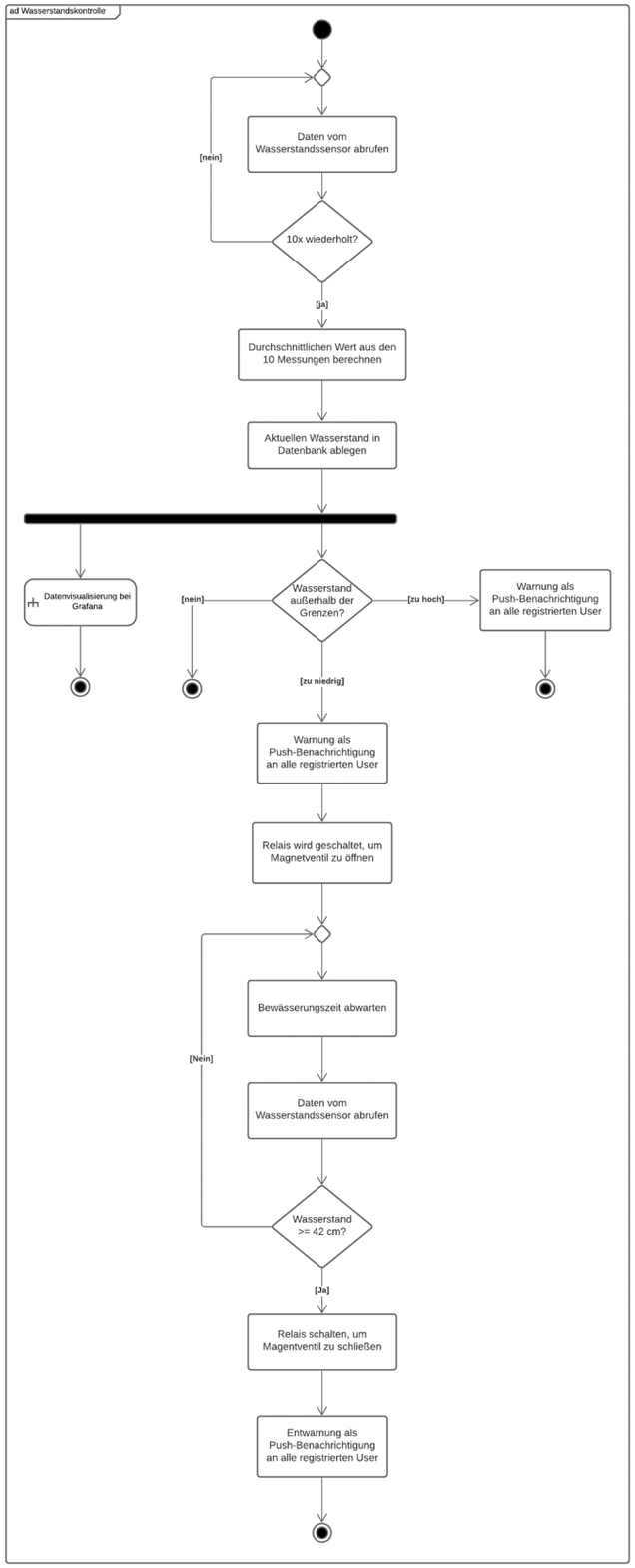
exceptional flow

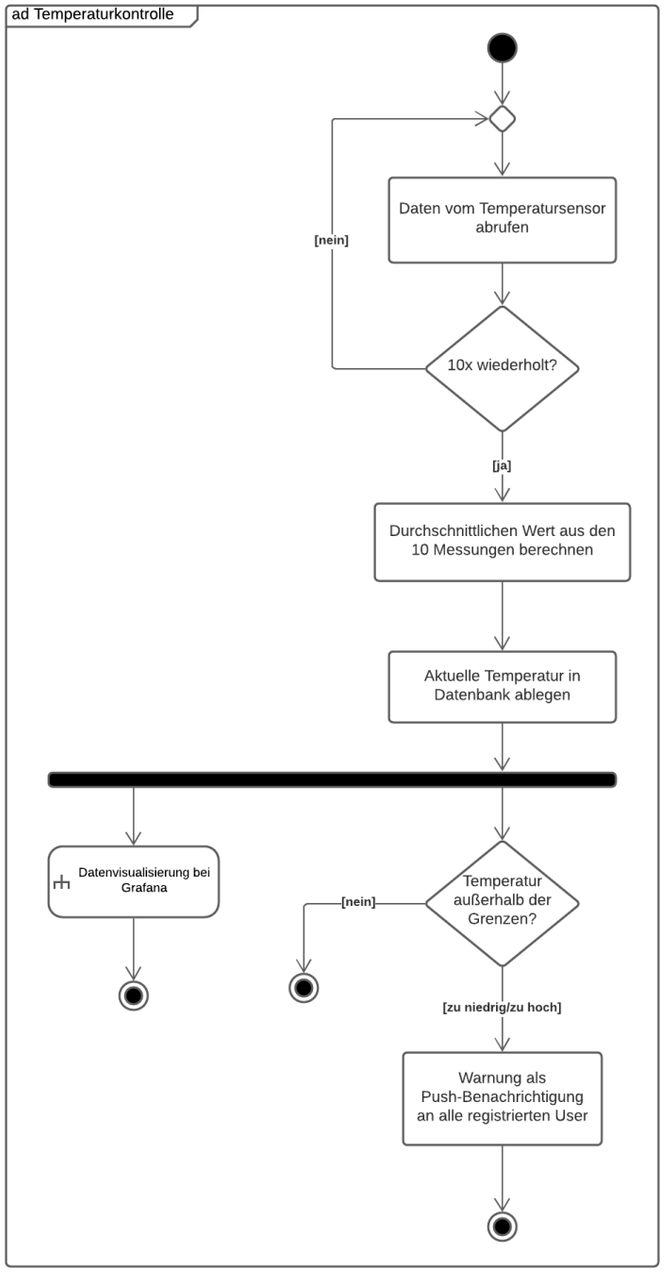
1.a Das Relais schaltet nicht.

postcondition Das Magnetventil kann nicht geöffnet bzw. geschlossen werden.

end Magnetventil öffnen/schließen

Aktivitätsdiagramme | Anwendungslogik





Aktivitätsdiagramme für Wasserstands- und Temperaturkontrolle.
Quelle: Eigene Darstellung

Durchgeführte Proof-of-Concepts

Proof-of-Concept #1 | Durchführung

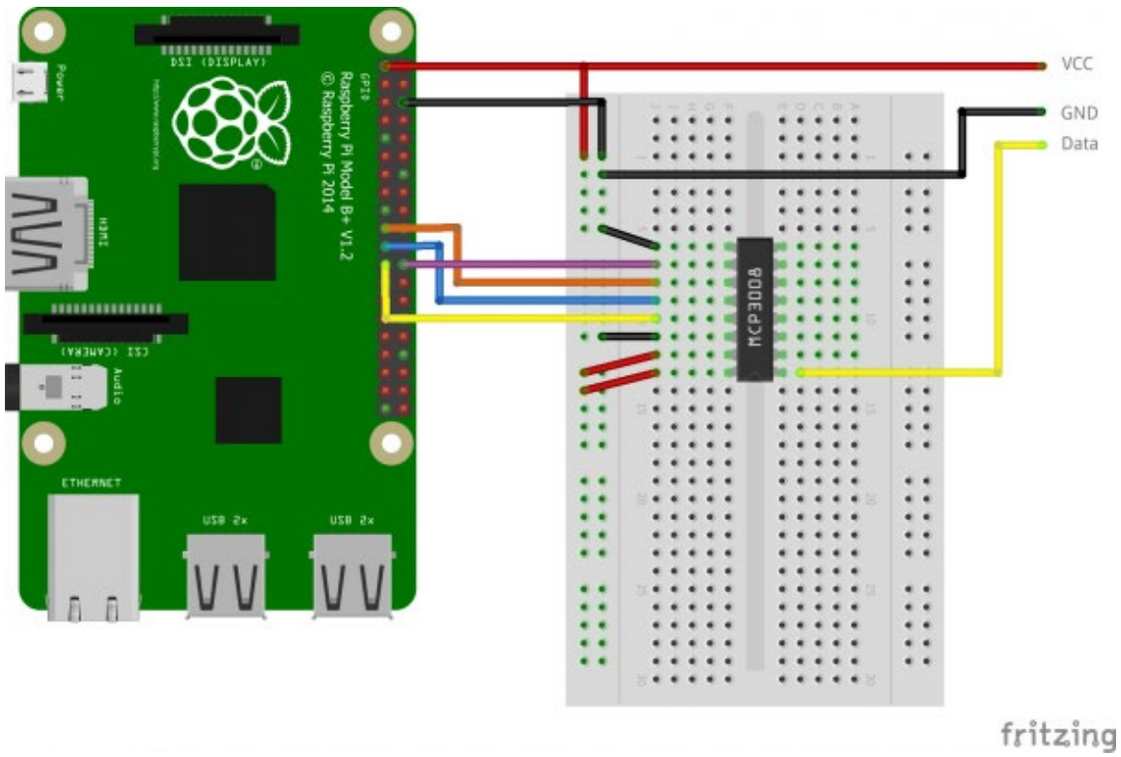
Erste Implementierungen und Installationen am Raspberry Pi

Die Erstinstallation des Raspberry Pi hat zum größten Teil problemlos funktioniert. Es wurde noch keine SSH-Verbindung eingerichtet, da zurzeit an einem lokalen Arbeitsplatz mit Tastatur und Bildschirm gearbeitet werden kann.

Da die Installationsdatei der Bibliothek SpiDev das Paket Setuptools benötigte, musste dieses noch zusätzlich installiert werden.

Das Anschalten des Raspberry Pi gestaltet sich zurzeit noch etwas umständlich, da das Stromkabel hierfür aus- und wieder angesteckt werden muss. Hierfür wäre es sicherlich sinnvoll, einen Taster zu installieren.

Proof-of-Concept #2 | Durchführung



Schematische Darstellung der Verbindung zwischen Raspberry Pi und AD-Wandler MCP3008.
Quelle: https://tutorials-raspberrypi.de/raspberry-pi-mcp3008-analoge-signale-auslesen/

Messdaten des Temperatursensors PT 100 am Raspberry Pi empfangen

Für die Installation des AD-Wandlers wurde dieses Tutorial befolgt: https://tutorials-raspberrypi.de/raspberry-pi-mcp3008-analoge-signale-auslesen/

Die Abbildung zeigt den Aufbau der Verbindung zwischen Raspberry Pi und dem AD-Wandler MCP3008. Nicht dargestellt ist der Temperatursensor, der an das hier gelb gefärbte, rechte Kabel angeschlossen ist.

Zunächst wurde zur Durchführung der SPI-Bus aktiviert und anschließend die SpiDev-Bibliothek installiert. Mithilfe einer eigenen Python-Klasse für den MCP3008 konnten die analogen Kanäle erfolgreich ausgelesen werden. Hierbei konnte festgestellt werden, dass der AD-Wandler digitale Werte von 0 bis 4095 ausgibt, was sich in der Formel zum Auslesen der Werte niederschlägt:

(Digitaler Wert / 4095) * 3,3 V

Iteration Proof-of-Concept #2 | Durchführung

Verwendeter Temperatursensor.
Quelle: https://www.technikhaus.de/product_info.php?products_id=1331

Messdaten des neuen Temperatursensors DS18B20 am Raspberry Pi empfangen

Für die Installation des Temperatursensors wurde dieses Tutorial befolgt: https://pimylifeup.com/raspberry-pi-temperature-sensor/

Da der DS18B20 bereits digitale Daten liefert, musste für das Aufnehmen der Messwerte kein zusätzlicher AD-Wandler verwendet werden. Der Anschluss des Sensors verlief ohne Probleme und auch das Abrufen der Daten war mit Hilfe der oben genannten Anleitung schnell möglich.

Für das Auslesen der Daten vom DS18B20 müssen die Module W1-gpio und W1-therm in den Kernel geladen werden, wodurch ein eigenes Verzeichnis für den Sensor angelegt wird. Die Daten können anschließend aus der hier liegenden w1_slave-Datei gelesen werden. Wenn die erste Zeile der Datei aus YES endet, enthält die Datei eine zweite Zeile mit der Temperatur. Die ausgelesene Temperatur muss anschließend nur noch in Celsius umgerechnet werden.

Proof-of-Concept #3 | Durchführung

Verwendeter Wasserstandssensor.
Quelle: https://www.wellenshop.de/tankgeber-11233?number=11242

Messdaten des Wasserstandssensors am Raspberry Pi empfangen

Der verwendete Wasserstandssensor ändert seinen elektrischen Widerstand abhängig von der Füllhöhe. Der AD-Wandler wertet diesen veränderlichen Widerstand aus und liefert die digitalen Messwerte an den Raspberry Pi. Mithilfe einer eigenen Python-Klasse für den MCP3008 konnten die analogen Kanäle erfolgreich ausgelesen werden.

Zur Umrechnung der digitalen Daten in die Füllhöhe [cm] wurde eine Kalibrierung durchgeführt, die zur folgenden Geradegleichung geführt hat:

h = (-0.0336 * float(adc.read(channel = 1) + 157.608) [cm]

Proof-of-Concept #4 | Durchführung

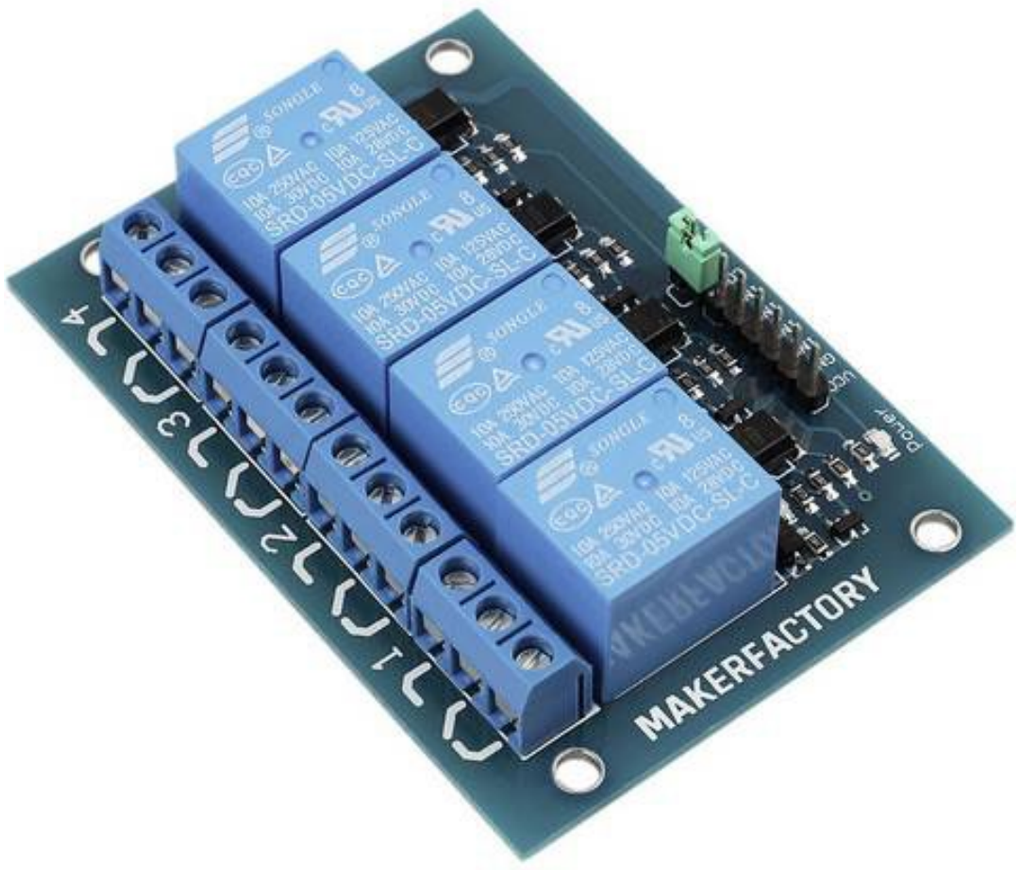
Kompatibilität der Hardwarekomponenten

Während der Durchführung des Proof-of-Concept #2 mussten wir feststellen, dass trotz starker Temperaturänderungen keine passenden Werte vom PT100 ausgelesen werden konnten.

Um die Funktionstüchtigkeit der restlichen Komponenten zu überprüfen und hier eine Fehlfunktion ausschließen zu können, wurde anstelle des PT 100 ein Potentiometer an den AD-Wandler angeschlossen. Da auf dem Raspberry Pi nun realistische Werte festgestellt werden konnten, musste der Rückschluss gezogen werden, dass die sehr feine Messauflösung des Temperatursensors nicht zu unserer restlichen Hardware passt.

Als Konsequenz kommt nun der Temperatursensor DS18B20 zum Einsatz (siehe Iteration PoC #2, Folie 17).

Proof-of-Concept #5 | Durchführung



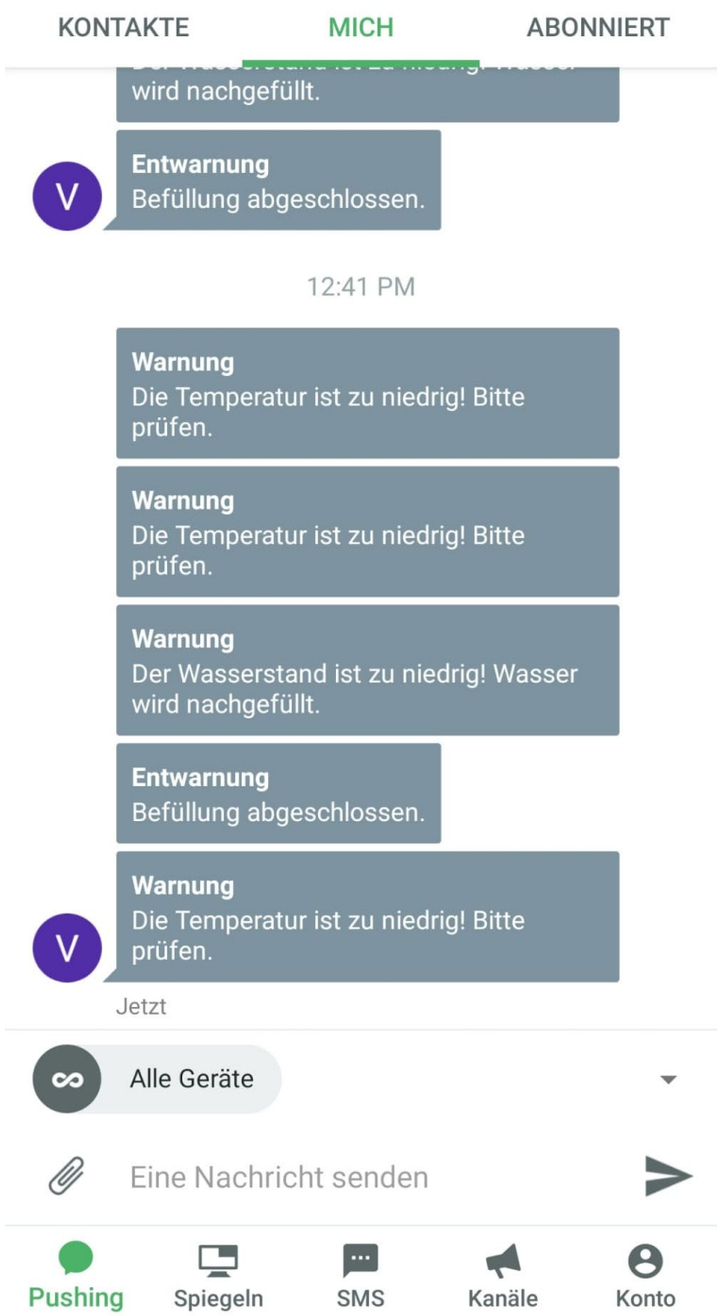
Verwendetes 4-Kanal Relais-Modul.
Quelle: https://www.conrad.de/de/p/makerfactory-mf-6402393-relais-modul-1-st-2134131.html

Ansprechen der Relais (Aktor)

Um die Verbindung zwischen Raspberry Pi und Relais zu testen, wurde dieses Tutorial befolgt: https://tutorials-raspberrypi.de/raspberry-pi-relais-schalter-steuern/

Mithilfe von zwei Jumper Kabeln wurde die Relais-Platine zunächst mit einer Spannung von 3,3 Volt versorgt. Anschließend wurde das Relais 1 mit einem GPIO-Pin des Raspberry Pi verbunden und über ein Python-Skript aktiviert und deaktiviert.

Proof-of-Concept #6 | Durchführung



Screenshot aus der Pushbullet-App.
Quelle: Eigene Darstellung

Senden von Push-Benachrichtigungen über Pushbullet

Für die Einrichtung der Push-Benachrichtigungen via Pushbullet wurde dieses Tutorial befolgt: https://iotdesignpro.com/projects/home-security-system-using-raspberry-pi-and-pir-sensor-with-push-notification-alert

1. Pushbullet Account einrichten, um einen Access Token zu erhalten. Anschließend wurde die Pushbullet App heruntergeladen (gleiches Konto).
2. Pushbullet Bibliothek herunterladen und auf dem Raspberry Pi installieren.
3. Code mit dem aktuellen Access Token und dem Gerätenamen einfügen und jeweilige Benachrichtigungen formulieren (Wasserstand, Temperatur ist zu hoch bzw. zu niedrig).
   Befinden sich nun Messungen außerhalb des definierten Messwertebereichs, wird eine entsprechende Pushbenachrichtigung an das Endgerät gesendet.

→ Tests mit kritischen Werten wurden erfolgreich abgeschlossen.

Proof-of-Concept #7 | Durchführung

Es kommt eine Influx-Datenbank zum Einsatz.
Quelle: https://www.influxdata.com/

Speicherung der Daten in lokaler Datenbank

Für die Installation der Influx-Datenbank wurde dieses Tutorial befolgt: https://pimylifeup.com/raspberry-pi-influxdb/

InfluxDB ist eine Time Series Database und damit speziell für Zeitreihen geeignet. eine Datenbank, welche speziell für Zeitreihen entwickelt wurde. Nach der erfolgreichen Installation konnte über die Influx-Kommandozeile die Datenbank für die Messdaten angelegt und eine Authentifizierung eingerichtet werden.

Um auf InfluxDB im Python-Skript zugreifen zu können, wurde die InfluxDB Client Bibliothek verwendet. Im Skript für den Rapid Prototype wurden außerdem die Influx-Zugangsdaten sowie der Name der zu verwendenden Datenbank hinterlegt. Um die Messdaten in die Datenbank einzufügen, werden diese zunächst im JSON-Format gespeichert und anschließend dem Client übergeben. InfluxDB versieht die Daten selbständig mit einem Zeitstempel, es kann aber auch eine eigene Zeitangabe eingefügt werden.

Da das Echtzeit-Modul für das 4. Audit noch installiert werden soll, können diese Daten dann anstelle des Influx-Zeitstempels gespeichert werden.

Proof-of-Concept #8 | Durchführung

Screenshot des verknüpften Grafana-Dashboards.
Quelle: Platzhalter (ersetzen)

Visualisierung der Messdaten

Für die Einrichtung von Grafana auf dem Raspberry Pi wurde folgendes Tutorial eingesetzt: https://pimylifeup.com/raspberry-pi-grafana/

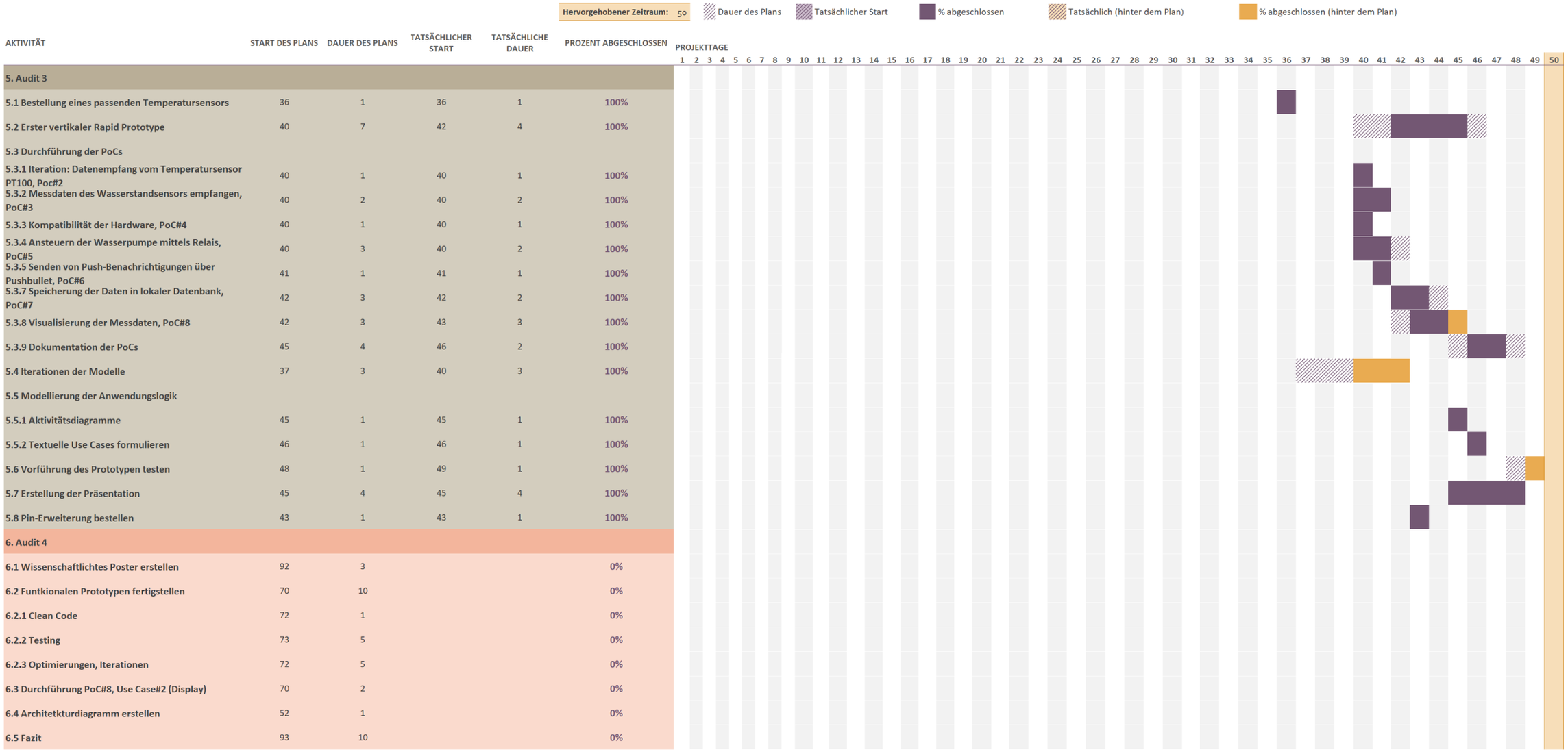
Nach der erfolgreichen Installation von Grafana konnte das zugehörige Web Interface über die Eingabe der IP-Adresse und dem Port 3000 aufgerufen werden. Nach dem Login wurden zunächst neue Zugangsdaten festgelegt, um eine höhere Sicherheit zu gewährleisten – auch in Hinsicht auf den möglichen späteren Online-Zugriff.

Als möglicher Dateninput für Grafana eignen sich Influx-Datenbanken sehr gut. Über das Web Interface war es anschließend möglich, mit den Zugangsdaten der Aquaponik-Datenbank das erste InfluxDB-Dashboard zu erstellen. Hierfür mussten die Zugangsdaten der Aquaponik-Datenbank hinterlegt werden. Nach erfolgreich getesteter Verbindung war es möglich, die ersten Dashboard-Komponenten anzulegen. Für den Rapid Prototype wurden hierfür der zeitliche Verlauf der Temperatur- und Wasserstandsdaten sowie die aktuellen Werte in großer Anzeige ausgewählt.

Projektplan & Deliverables für Audit 3

Gantt-Diagramm zur Projektplanung

Entwicklung eines Systems zur Überwachung von Aquaponikanlagen



AKTIVITÄT	START DES PLANS	DAUER DES PLANS	TATSÄCHLICHER START	TATSÄCHLICHE DAUER	PROZENT ABGESCHLOSSEN
5. Audit 3					
5.1 Bestellung eines passenden Temperatursensors	36	1	36	1	100%
5.2 Erster vertikaler Rapid Prototype	40	7	42	4	100%
5.3 Durchführung der PoCs					
5.3.1 Iteration: Datenempfang vom Temperatursensor PT100, Poc#2	40	1	40	1	100%
5.3.2 Messdaten des Wasserstandsensors empfangen, PoC#3	40	2	40	2	100%
5.3.3 Kompatibilität der Hardware, PoC#4	40	1	40	1	100%
5.3.4 Ansteuern der Wasserpumpe mittels Relais, PoC#5	40	3	40	2	100%
5.3.5 Senden von Push-Benachrichtigungen über Pushbullet, PoC#6	41	1	41	1	100%
5.3.7 Speicherung der Daten in lokaler Datenbank, PoC#7	42	3	42	2	100%
5.3.8 Visualisierung der Messdaten, PoC#8	42	3	43	3	100%
5.3.9 Dokumentation der PoCs	45	4	46	2	100%
5.4 Iterationen der Modelle	37	3	40	3	100%
5.5 Modellierung der Anwendungslogik					
5.5.1 Aktivitätsdiagramme	45	1	45	1	100%
5.5.2 Textuelle Use Cases formulieren	46	1	46	1	100%
5.6 Vorführung des Prototypen testen	48	1	49	1	100%
5.7 Erstellung der Präsentation	45	4	45	4	100%
5.8 Pin-Erweiterung bestellen	43	1	43	1	100%
6. Audit 4					
6.1 Wissenschaftlichtes Poster erstellen	92	3			0%
6.2 Funtkionalen Prototypen fertigstellen	70	10			0%
6.2.1 Clean Code	72	1			0%
6.2.2 Testing	73	5			0%
6.2.3 Optimierungen, Iterationen	72	5			0%
6.3 Durchführung PoC#8, Use Case#2 (Display)	70	2			0%
6.4 Architetkturdiagramm erstellen	52	1			0%
6.5 Fazit	93	10			0%

Projektplan Fokus Audit 3 und 4.
Quelle: Eigene Darstellung