Bildungszentrum Uster
Höhere Berufsbildung Uster

HBU

Pflichtenheft

Autom. Gewächshaus auf Raspberry Pi Basis

Severin Probst, Michael Pfister und Stephan Hauser

1. Allgemeines 3

1.1 Ausgangslage 3
1.2 Sinn und Zweck 3
1.3 Vorgesehene Erweiterungen 3
1.4 Termine / Meilensteine 3

2. Allgemeine Beschreibung 4

2.1 Übersicht 4
2.1.1 Abgrenzung des Umfelds (Systemgrenze) 4
2.1.2 Rahmenbedingungen 4
2.2 Hardware 4
2.3 Software 5
2.3.1 Module (Django-APPs) 5
2.3.2 Working Prototype 6
2.4 Übersicht der Funktionen 6
2.5 Muss Ziele 7
2.6 Kann Ziele 7

3. Funktionsbeschreibung 8

3.1 Detaillierte Funktionsbeschreibungen 8
3.2 Meldungen 9
3.2.1 Systemmeldungen 9
3.2.2 Fehlermeldungen 9
3.2.3 Logdateien 9

4. Datenbasis 9

5. Externe Schnittstellen 10

5.1 Benutzeroberfläche 10
5.1.1 Startseite 10
5.1.2 Eventlogseite 10
5.1.3 Relaissteuerung 11
5.1.4 Sensorseite 11
5.2 Hardwareschnittstellen 11
5.3 Kommunikationsschnittstellen 11
5.4 Interne Schnittstellen 11

6. Leistungsanforderungen 12

7. Kontrollfunktionen 12

7.1 Fehlerdokumentation 12

1. Allgemeines

1.1 Ausgangslage

Es besteht bereits ein funktionierendes Gewächshaus. Unsere Herausforderung besteht darin, das Gewächshaus zu automatisieren. Dafür werden wir einen Raspberry Pi einsetzen. Über diesen werden diverse Sensoren angesteuert. Das Ganze soll über eine Web-Oberfläche steuerbar sein.

1.2 Sinn und Zweck

Automatisierung eines Gewächshauses, welches autonom funktionieren kann, optional auch ferngesteuert

1.3 Vorgesehene Erweiterungen

Kurzfristig:

- E-Mail-Benachrichtigung (z.B. bei tiefem Wasserstand in Wassertank)
- Füllstand im Wassertank überwachen
- Sensoren interagieren untereinander (z.B. Bodenfeuchtigkeitssensor kann nach Bedarf Bewässerungsanlage aktivieren.)
- Schutzsensor gegen Überschwemmung
- Notabschaltung

Langfristig:

Unser Ziel ist es ein **marktreifes Produkt** zu erstellen. Nach dem Abschluss dieses Projektes wollen wir entscheiden, ob wir bei der Diplomarbeit unsere Software weiterentwickeln. Ein wesentlicher Aspekt wird dabei sein, die **Sicherheit** zu optimieren. Unter anderem SQL-Injektionschutz oder das implementieren eines Multiusersystems, welches gegebenenfalls eine Schnittstelle zu einem **Cloud-Service** hat.

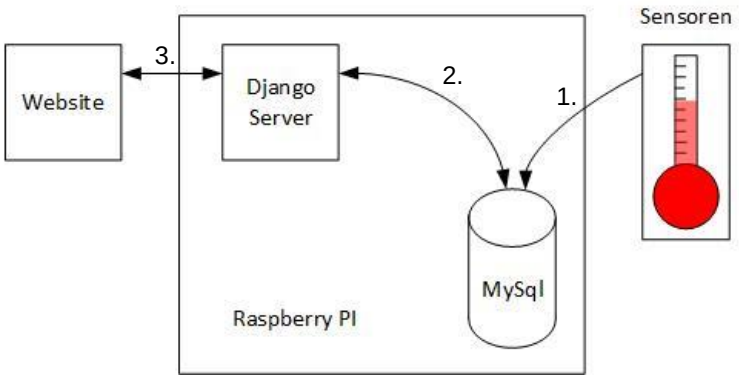
1.4 Termine / Meilensteine

Nach jeder Phase gibt es einen Meilenstein, um diese zu besprechen. Start und Ende sind fix. Dazwischen liegen folgende Termine:

01.10.2018	Start
16.11.2018	Meilenstein Phase Vorstudie
20.11.2018	Meilenstein Phase Hardware
03.12.2018	Meilenstein Phase Layout
21.12.2018	Meilenstein Phase Konzept
11.01.2019	Meilenstein Phase Serverseitige Programmierung
18.01.2019	Meilenstein Phase Clientseitige Programmierung
25.01.2019	Meilenstein Phase Kontrolle
04.02.2019	Abgabe Vordiplomarbeit
09.03.2019	Präsentation Vordiplomarbeit

2. Allgemeine Beschreibung

2.1 Übersicht



1. Die Sensoren liefern Werte, welche in einer MySQL-Datenbank auf dem Raspberry Pi abgespeichert werden.
2. Über den Django-Webserver werden mit Python die Werte aus der Datenbank ausgelesen und auf der Website dargestellt.
3. Die Website wird auf dem Django-Webserver betrieben. Sie basiert auf HTML5, CSS3 und JavaScript.

2.1.1 Abgrenzung des Umfelds (Systemgrenze)

In dieser Arbeit werden wir den Fokus auf die Basisfunktionalität legen. Ein Sicherheitskonzept ist erst in einem zweiten Schritt angedacht.

2.1.2 Rahmenbedingungen

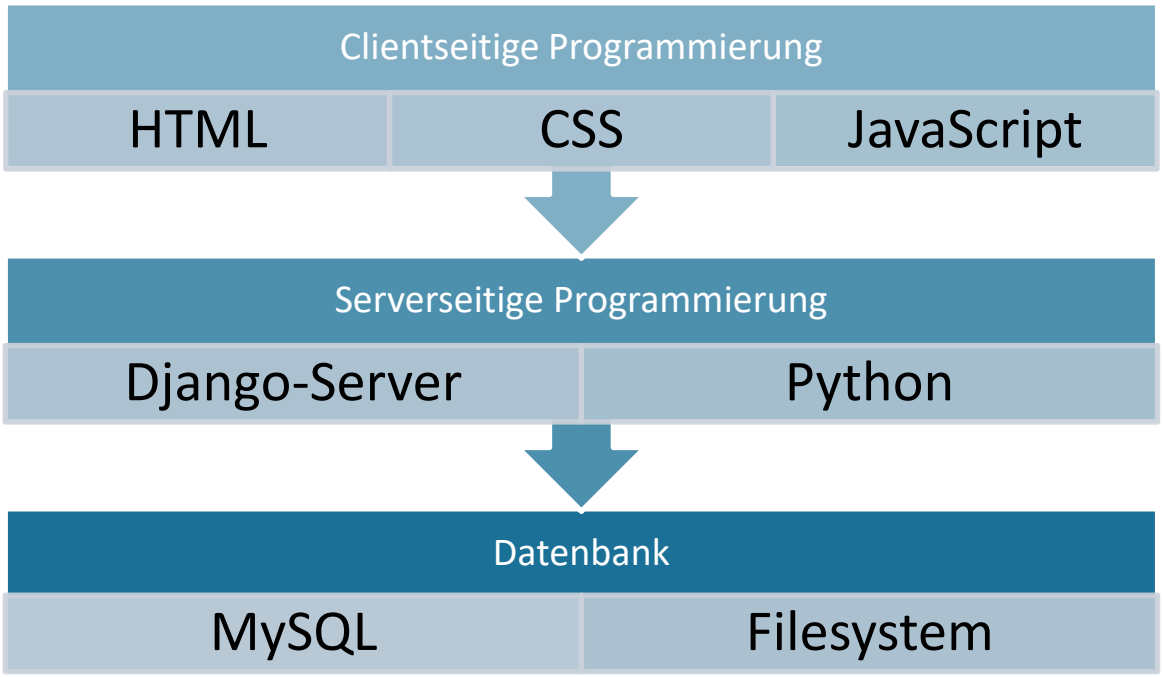
Das Webinterface soll responsive sein und W3C konform.
Animationen werden in JQuery und CSS3 realisiert.
Serverseitig wird Python verwendet.
Python API wird zum Auslesen der Sensoren und Ansteuern der Aktoren verwendet.

2.2 Hardware

- 1 x Raspberry Pi 3
- 3 x DTH22 (Luftfeuchtigkeits- und Temperatursensoren)
- 2 x Bodenfeuchtigkeitssensor
- Diverse Relais zur Steuerung des Belüftungs- und Bewässerungssystems, sowie der Steuerung der Notabschaltung

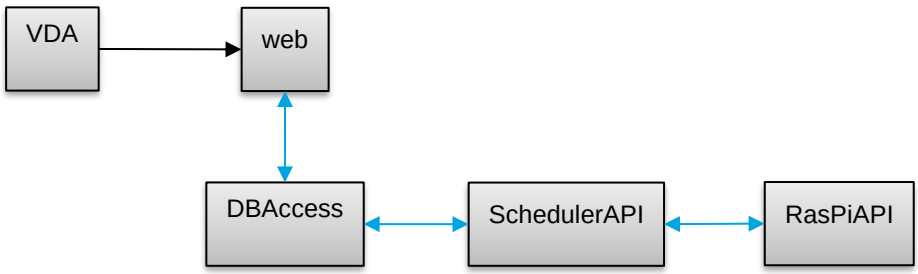
2.3 Software

Die Software haben wir in folgenden drei Schichten aufgeteilt, welche bis zu einem gewissen Grad unabhängig entwickelt werden können. So ist gewährleistet, dass wir zu dritt am selben Projekt programmieren können.



2.3.1 Module (Django-APPs)

Die Django-Installation soll in folgende fünf Apps aufgeteilt werden.



VDA
In diesem Modul befinden sich die Settings des Django-Frameworks. Seine Aufgabe ist es, die Anfragen (Requests) des Benutzers über die Weboberfläche abzufangen und an die entsprechende Django-View (Logic) weiterzuleiten.

web
Das web-Modul umfasst die notwendigen Ressourcen für das Darstellen der Webseite. Der Zugriff auf die DB erfolgt immer über das Modul DBAccess und nie direkt aus diesem Modul.

DBAccess
In diesem Modul befinden sich die Django-Model, welche den Aufbau der Datenbank bestimmen.
Des Weiteren hat es pro Model einen Infoprovider, welcher den direkten Zugriff auf das entsprechende Model kapselt. So ist sichergestellt, dass der Aufruf nach aussen immer gleich ist. Ausnahme sind Delete-, Insert- und Update-Funktionen.

SchedulerAPI
Über dieses Modul wird das Auslesen der Sensoren gesteuert.

RasPiAPI
Dieses Modul darf Code beinhalten, welcher nur auf dem RasPi lauffähig ist. Für die Testumgebung wird ein Dummy-Modul implementiert, welches Random-Werte zurückliefert anstelle der Sensorwerte.

2.3.2 Working Prototype

Uns ist wichtig, dass wir zu jeder Zeit einen funktionierenden Prototype haben, damit wir eine lauffähige Arbeit abgeben können. Sollte die Zeit nicht reichen, werden wir entsprechende Module auf später verschieben.

2.4 Übersicht der Funktionen

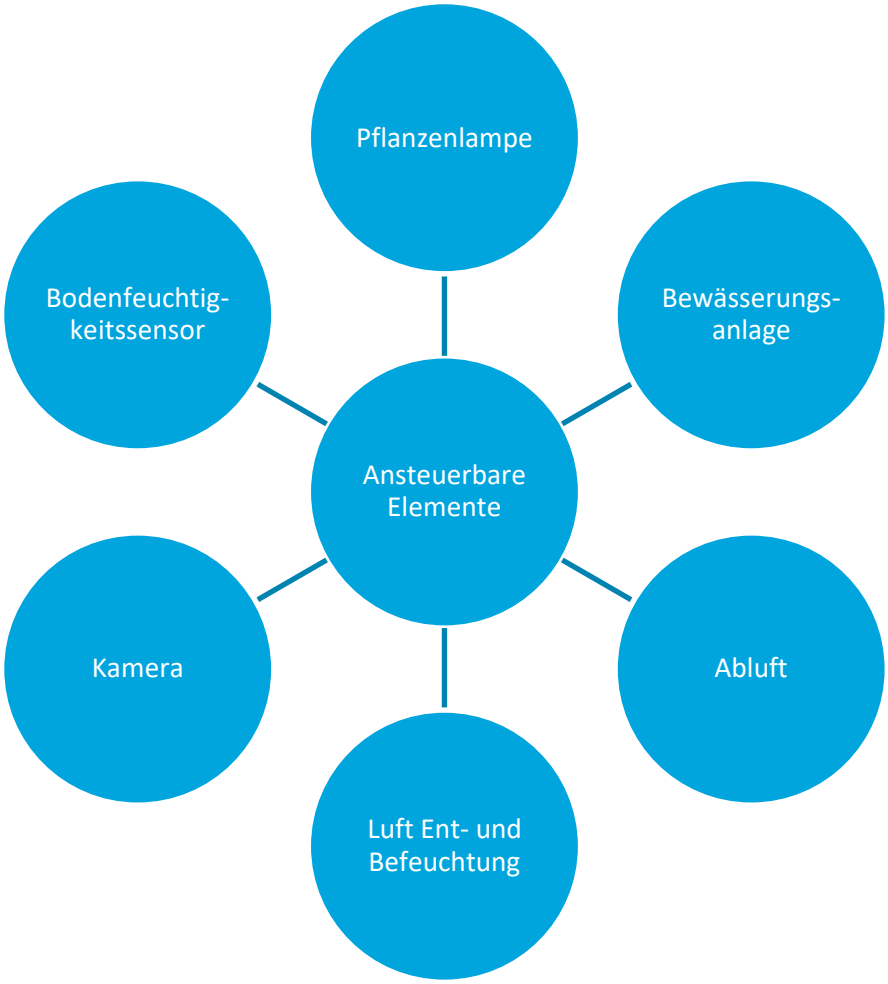
Nicht authentifizierte Benutzer werden automatisch auf eine Login-Seite weitergeleitet, welche als Landingpage gestaltet ist.

Nach dem Login wird der Benutzer auf die Startseite des Tools weitergeleitet.

Die am Raspberry Pi angeschlossenen Sensoren werden über ein Python-Skript in regelmässigen Abständen in einer MySQL Datenbank abgespeichert. Über das Webinterface können die Intervalle pro Sensor definiert werden.

Der Benutzer kann sich auf unserer Webplattform einloggen. Danach hat er die Möglichkeit, über eine Kamera das ganze System optisch zu überwachen. Der Benutzer sieht die Daten der Sensoren, die laufend graphisch aufbereitet werden. Er hat z.B. die Möglichkeit, manuell zu bewässern oder in der Weboberfläche einen Timer zu stellen. Das Ganze basiert auf einem Raspberry Pi und wird mit Python programmiert. (Optional soll der Benutzer die Möglichkeit haben neue Sensoren anzulegen und so das System zu erweitern.)

2.5 Muss Ziele



2.6 Kann Ziele

E-Mail-Benachrichtigung

- Bei kritischen Sensorwerten
- Alarm vom Schutzsensor gegen Überschwemmungen

Füllstand Wassertank

Sensoren interagieren

- Wenn der Bodenfeuchtigkeitssensorwert unter den Schwellwert fällt, dann wird die Bewässerung aktiviert
- Wird es zu heiss, soll automatisch die Lüftung aktiviert werden

Bewässerungsanlage aktivieren

- Bodenfeuchtigkeitssensor kann nach Bedarf Bewässerungsanlage aktivieren

Schutzsensor gegen Überschwemmung

- Am Boden des Gewächshauses soll ein Sensor installiert werden, welcher auf Wasserkontakt reagiert

Notabschaltung

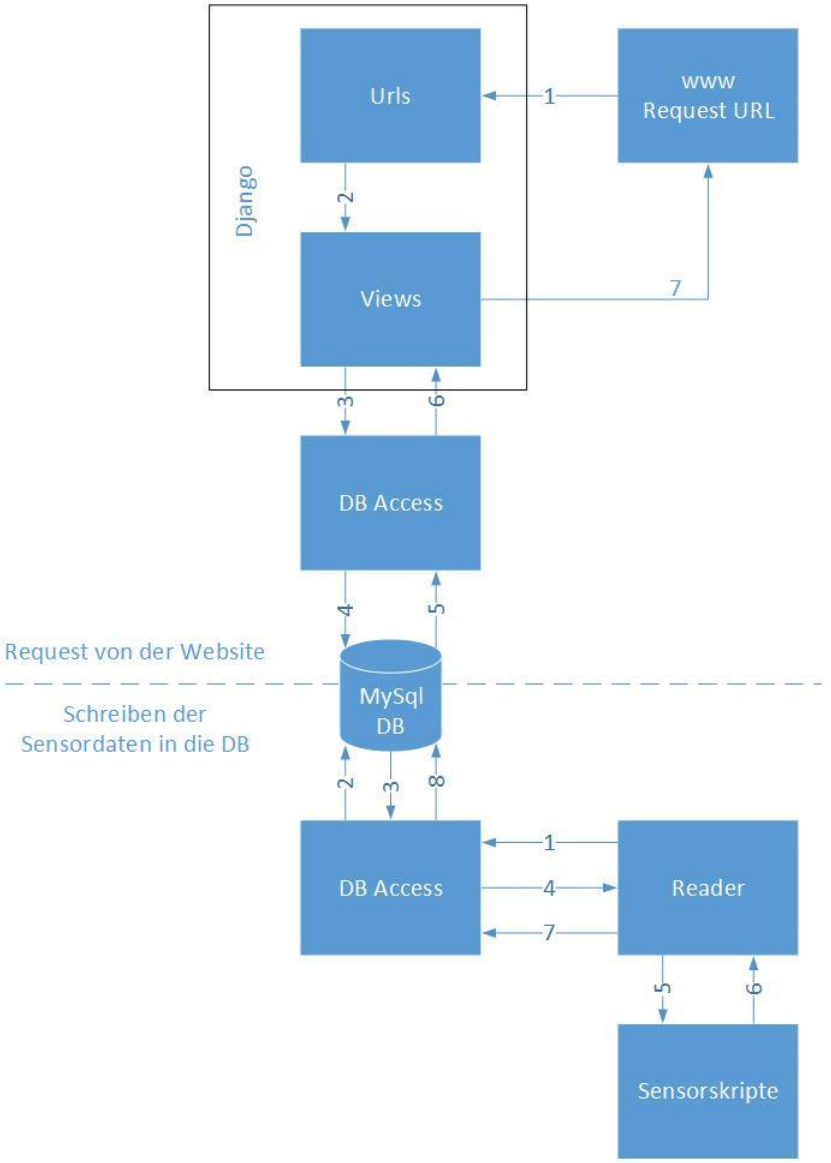
- Wenn der Schutzsensor gegen Überschwemmung anschlägt, wird das System heruntergefahren
- Bei unkontrolliertem Temperaturanstieg soll die Notabschaltung ebenfalls erfolgen

Neue Sensoren anlegen

- Es sollen weitere Sensortypen erfasst werden, welche dann bei den Projekten eingesetzt werden können

3. Funktionsbeschreibung

3.1 Detaillierte Funktionsbeschreibungen



Client Seite

1. Im Browser wird z.B. der URL von der Startseite eingegeben. Es wird ein Request an den Server gesendet.
2. Über die entsprechenden URL-Pattern wird auf eine entsprechende View verwiesen.
3. In den Views wird der Request verarbeitet und auf die Logik der Website zugegriffen, welche in separate Klassen verkapselt ist.
4. Über das DB-Access APP wird eine Verbindung zur MySQL Datenbank aufgebaut.
5. Die aufgerufene Klasse im DB-Access liest nun die angefragten Daten aus der Datenbank.
6. Die aufgerufene Klasse gibt die angefragten Daten an die Views zurück.
7. Die Rohdaten und das HTML-Template werden nun zusammengefügt und an den Browser zurückgesendet.

Server Seite

1. Der Reader ermöglicht es periodisch Sensorabfragen durchzuführen.
2. Über das DB Access APP wird eine Verbindung zur Datenbank aufgebaut.
3. In der MySQL-DB wird nachgeschaut welche Sensoren im Projekt zur Verfügung stehen. Diese Information wird an das Access Skript zurückgesendet.
4. Das DB Access Skript bereitet die Daten auf und gibt Sie an den Reader zurück.
5. Der Reader steuert die entsprechenden Sensorskripte an.
6. Die Werte werden ausgelesen und
7. über den DB-Access schliesslich in MySQL-DB abgespeichert.
8. Dort stehen die Daten für den Client bereit.

3.2 Meldungen

3.2.1 Systemmeldungen

Ein datenbankbasiertes Event-Log wird implementiert.

3.2.2 Fehlermeldungen

Es soll ein datenbankbasiertes Event-Log implementiert werden, welches systemweit die Fehler protokolliert. Zudem werden gezielt Fehlermeldungen an den Benutzer durchgereicht, wenn das System den Fehler nicht beheben kann.

3.2.3 Logdateien

Datenbankbezogene Fehler können typischerweise nicht in der Datenbank geloggt werden. Um auch diese Fehler zu protokollieren haben wir ein logfilebasiertes Event-Log vorgesehen.

4. Datenbasis

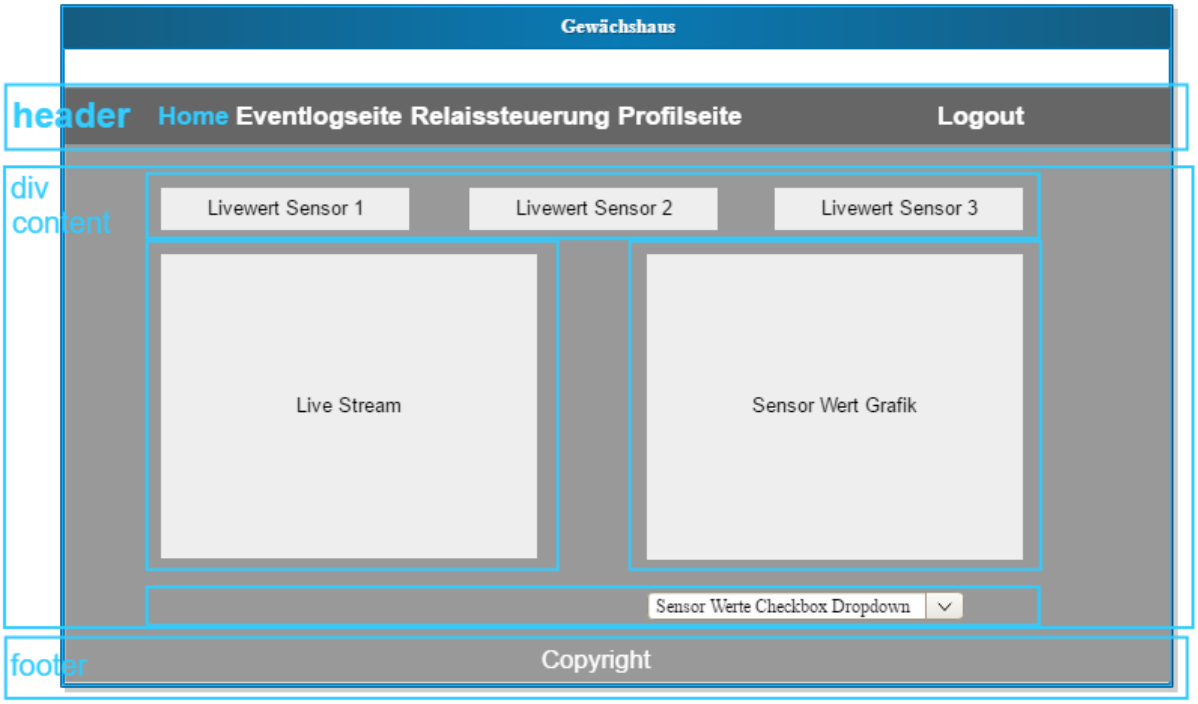
Die Daten werden über die Sensorskripte ausgelesen und mit den erforderlichen Skripten in die Datenbank geschrieben. Die Datenbanken sollen so aufgebaut sein, dass es möglich ist, den Sensoren und deren Daten Projekte zuzuweisen. Ist ein Zyklus (d.h. eine Ernte) vorüber, soll es möglich sein, die Daten und allenfalls erstellte Bilder als Projekt zu speichern. Dies soll verhindern, dass die Speicher unnötig belastet werden und das Programm an Geschwindigkeit einbüsst.

5. Externe Schnittstellen

5.1 Benutzeroberfläche

Die Benutzeroberfläche wird für Mobilgeräte optimiert, da auch von unterwegs die Sensorwerte überprüft werden können.

5.1.1 Startseite

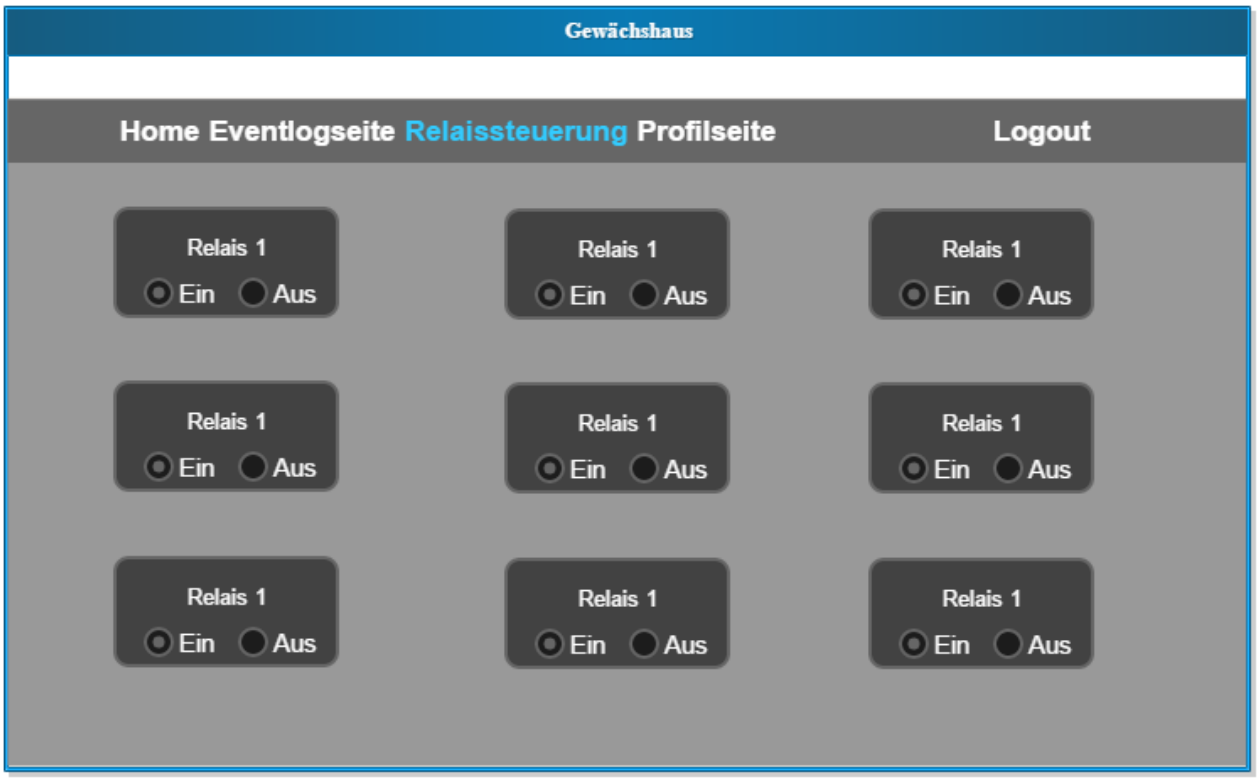


Auf der Startseite werden die wichtigsten Sensorwerte aufgeführt, welche beim Laden der Webseite aktuell abgefragt werden. Dazu kommen ein Livebild der Kamera sowie ein Liniendiagramm, welches die aufgezeichneten Sensorwerte darstellt. Über das Dropdown-Menu kann zwischen den einzelnen Sensoren hin und her gewechselt werden.

5.1.2 Eventlogseite

Die Eventlogseite beinhaltet eine einfache Auflistung der geloggten Ereignisse. Die Detailansicht öffnet sich in einem modalen Popup. Es ist ebenfalls möglich, einzelne Logeinträge zu löschen oder gegebenenfalls das gesamte Log zu löschen.

5.1.3 Relaissteuerung



Die einzelnen Relais werden auf der Seite aufgelistet. Es ist ersichtlich, ob sie ein- oder ausgeschaltet sind. Zudem ist eine Schaltfläche angedacht, um weitere Relais zu erfassen.

5.1.4 Sensorseite

Auf dieser Seite können Sensoren mutiert werden. Es ist auch ersichtlich, welche Sensoren dem aktiven Projekt zugewiesen sind.

5.2 Hardwareschnittstellen

- Die Sensoren und Relais werden über die GPIOs auf dem Raspberry Pi verbunden und angesteuert.
- Für gewisse Sensoren ist ein Analog/Digital Konverter notwendig.

5.3 Kommunikationsschnittstellen

- Der Raspberry Pi soll über WLAN angesteuert werden. Er soll ebenfalls von extern erreichbar sein. Dazu werden wir bei www.noip.com den DNS-Dienst aktivieren. Beim Modem müssen die Ports 22 (SSH) und 3389 (Remotedesktop) weitergeleitet werden.

5.4 Interne Schnittstellen

- Für die analogen Bodensensoren wird die I2C-Schnittstelle auf dem Raspberry Pi verwendet.

6. Leistungsanforderungen

Es soll möglich sein, die Sensordaten einmal pro Minute in die Datenbank abzuspeichern.
Die Datensätze sollen über das Webinterface verwaltbar sein.
Die Anzahl der Transaktionen ist nicht relevant.

7. Kontrollfunktionen

7.1 Fehlerdokumentation

Es wird ein Event-Log implementiert, welches Informationen, Felder und Warnungen photokopieren soll.