



## Diplomarbeit

# Entomophagie

### Untertitel der Arbeit

Imst, 15. März 2018

Eingereicht von

Leonid Hammer

Verantwortlich für IT: HTML, CSS, BWL: Kaufvertrag

Kevin Glatz

Verantwortlich für IT: SQL, C# BWL: Kaufvertrag

Tobias Haslwanter

Verantwortlich für IT: HTML, CSS, BWL: Kaufvertrag

Florian Tipotsch

Verantwortlich für IT: SQL, C# BWL: Kaufvertrag

Eingereicht bei

Stefan Stolz und Nina Margreiter

# Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbst verfasst und keine anderen als die angeführten Behelfe verwendet habe. Alle Stellen, die wörtlich oder inhaltlich den angegebenen Quellen entnommen wurden, sind als solche kenntlich gemacht. Ich bin damit einverstanden, dass meine Arbeit öffentlich zugänglich gemacht wird.

---

Ort, Datum

---

Leonid Hammer

---

Kevin Glatz

---

Tobias Haslwanter

---

Florian Tipotsch

# Abnahmeerklärung

Hiermit bestätigt der Auftraggeber, dass das übergebene Produkt dieser Diplomarbeit den dokumentierten Vorgaben entspricht. Des Weiteren verzichtet der Auftraggeber auf unentgeltliche Wartung und Weiterentwicklung des Produktes durch die Projektmitglieder bzw. die Schule.

---

Ort, Datum

---

Thorsten Schwerte

# Vorwort

z. B. Hinweise, wie das bearbeitete Thema gefunden wurde oder Dank für die Betreuung (Kooperationspartner/in, Betreuer/innen, Sponsoren) etc.

# **Abstract (Deutsch)**

(ca.  $\frac{1}{2}$  bis max. 2 Seiten) Kurzbeschreibung von Aufgabenstellung und Problemlösung.

# **Abstract (Englisch)**

(ca. ½ bis max. 2 Seiten)

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>11</b>
<b>Tabellenverzeichnis</b>	<b>12</b>
<b>Quelltexte</b>	<b>13</b>
<b>1 Einleitung</b>	<b>16</b>
<b>2 Projektmanagement</b>	<b>17</b>
2.1 Metainformationen . . . . .	17
2.1.1 Team . . . . .	17
2.1.2 Betreuer . . . . .	17
2.1.3 Partner . . . . .	17
2.1.4 Ansprechpartner . . . . .	17
2.2 Vorerhebungen . . . . .	17
2.2.1 Projektzieleplan . . . . .	17
2.2.2 Projektumfeld . . . . .	18
2.2.3 Risikoanalyse . . . . .	18
2.3 Pflichtenheft . . . . .	18
2.3.1 Zielbestimmung . . . . .	18
2.3.2 Produkteinsatz und Umgebung . . . . .	18
2.3.3 Funktionalitäten . . . . .	19
2.3.4 Testszenarien und Testfälle . . . . .	19
2.3.5 Liefervereinbarung . . . . .	19

2.4	Planung . . . . .	20
2.4.1	Projektstrukturplan . . . . .	20
2.4.2	Meilensteine . . . . .	20
2.4.3	Gant-Chart . . . . .	20
2.4.4	Abnahmekriterien . . . . .	20
2.4.5	Pläne zur Evaluierung . . . . .	20
2.4.6	Ergänzungen und zu klärende Punkte . . . . .	20
<b>3</b>	<b>Vorstellung des Produktes</b>	<b>21</b>
<b>4</b>	<b>Eingesetzte Technologien</b>	<b>22</b>
4.1	Technologie für Webapp . . . . .	23
4.1.1	HTML - Hypertext Markup Language . . . . .	23
4.1.2	Was ist Yii . . . . .	23
4.1.3	PureMVC . . . . .	25
4.1.4	Laravel . . . . .	25
4.2	Technologien für die Datenauslesung . . . . .	26
4.2.1	Gas Sensoren . . . . .	26
4.2.2	Wärmesensor . . . . .	28
4.2.3	Luftfeuchtigkeit . . . . .	29
4.2.4	kinetischer Motor . . . . .	29
4.2.5	Hebel . . . . .	29
4.2.6	Datenübertragung . . . . .	29
<b>5</b>	<b>Problemanalyse</b>	<b>30</b>
5.1	USE-Case-Analyse . . . . .	30
5.2	Domain-Class-Modelling . . . . .	31
5.3	User-Interface-Design . . . . .	31
<b>6</b>	<b>Systementwurf</b>	<b>32</b>
6.1	Architektur . . . . .	32
6.1.1	Design der Komponenten . . . . .	32
6.1.2	MVC - Model, View, Controller . . . . .	32
6.1.3	CRUD - Create, Read, Update, Delete . . . . .	34



6.1.4	Yii2 . . . . .	35
6.1.5	Benutzerschnittstellen . . . . .	36
6.1.6	Datenhaltungskonzept . . . . .	36
6.1.7	Konzept für Ausnahmebehandlung . . . . .	38
6.1.8	Sicherheitskonzept . . . . .	38
6.1.9	Design der Testumgebung . . . . .	38
6.1.10	Desing der Ausführungsumgebung . . . . .	39
6.2	Detaillentwurf . . . . .	39
<b>7</b>	<b>Implementierung</b>	<b>41</b>
7.1	Webapp . . . . .	42
7.1.1	Mockups . . . . .	43
7.1.2	Datenbankzugriffe . . . . .	46
7.1.3	Datenübermittlung . . . . .	46
7.1.4	Rest Schnittelle . . . . .	47
7.2	Verkablung . . . . .	49
7.2.1	Aufbau . . . . .	49
7.3	Programmierung . . . . .	50
7.3.1	DHT11 & CCS811 . . . . .	50
7.3.2	Hebel . . . . .	52
7.3.3	SG90 . . . . .	52
7.3.4	photocell . . . . .	53
<b>8</b>	<b>Deployment</b>	<b>54</b>
<b>9</b>	<b>Tests</b>	<b>55</b>
9.1	Systemtests . . . . .	55
9.2	Akzeptanztests . . . . .	55
<b>10</b>	<b>Projektevaluation</b>	<b>56</b>
<b>11</b>	<b>Benutzerhandbuch</b>	<b>57</b>
<b>12</b>	<b>Betriebswirtschaftlicher Kontext</b>	<b>58</b>

<b>13 Zusammenfassung</b>	<b>59</b>
<b>Literaturverzeichnis</b>	<b>60</b>

# **Abbildungsverzeichnis**

# **Tabellenverzeichnis**

## Quelltexte

# **Einleitende Bemerkungen**

# Notationen

Beschreibung wie Code, Hinweise, Zitate etc. formatiert werden

# **1 Einleitung**



# **2 Projektmanagement**

## **2.1 Metainformationen**

### **2.1.1 Team**

### **2.1.2 Betreuer**

### **2.1.3 Partner**

### **2.1.4 Ansprechpartner**

## **2.2 Vorerhebungen**

### **2.2.1 Projektzieleplan**

Projektziele-Hierarchie - SMART

### **2.2.2 Projektumfeld**

- Identifikation der Stakeholder
- Charakterisierung der Stakeholder
- Maßnahmen
- Grafische Darstellung des Umfeldes

### **2.2.3 Risikoanalyse**

- Risikomatrix

## **2.3 Pflichtenheft**

### **2.3.1 Zielbestimmung**

- Projektbeschreibung
- IST-Zustand
- SOLL-Zustand
- NICHT-Ziele (Abgrenzungskriterien)

### **2.3.2 Produkteinsatz und Umgebung**

- Anwendungsgebiet
- Zielgruppen

- Betriebsbedingungen
- Hard-/Softwareumgebung

### **2.3.3 Funktionalitäten**

- MUSS-Anforderungen
  - Funktional
  - Nicht-funktional
- KANN-Anforderungen
  - Funktional
  - Nicht-funktional

### **2.3.4 Testszenarien und Testfälle**

- Beschreibung der Testmethodik
- Testfall 1
- Testfall 2
- ...

### **2.3.5 Liefervereinbarung**

- Lieferumfang
- Modus
- Verteilung(Deployment)

## **2.4 Planung**

### **2.4.1 Projektstrukturplan**

### **2.4.2 Meilensteine**

### **2.4.3 Gant-Chart**

### **2.4.4 Abnahmekriterien**

### **2.4.5 Pläne zur Evaluierung**

### **2.4.6 Ergänzungen und zu klärende Punkte**

# **3 Vorstellung des Produktes**

Vorstellung des fertigen Produktes anhand von Screenshots, Bildern, Erklärungen.

## 4 Eingesetzte Technologien

- Kurzbeschreibung aller Technologien, die verwendet wurden.
- Technologien die aus dem Unterricht bekannt sind, nur nennen und deren Einsatzzweck im Projekt beschreiben, nicht die Technologien selbst.
- Technologien die aus dem Unterricht nicht bekannt sind, im Detail beschreiben incl. deren Einsatz im Projekt
- Fokus auf eingesetzten Frameworks

## **4.1 Technologie für Webapp**

- PHP - Für Webapp
- Html - Für Webapp
- MySql - Für Datenbanken
- Yii2 - Für Webapp
- MVC - Model, View, Controller - Für Webapp
- CRUD - Create, Read, Update, Delete - Für Webapp
- REST - Für Datenbank, Webapp und Datenspeicherung

### **4.1.1 HTML - Hypertext Markup Language**

HTML sind die Grundlagen für jede Website. Sie bilden die Grundstruktur jeder Webseite, und werden von Browsern dargestellt. HTML wird von dem World Wide Web Consortium (W3C) (3) und dem Web Hypertext Application Technology Working Group (WHATWG) (2) weiterentwickelt.

Gute Ressourcen zum lernen und schreiben von HTML findet man auf [w3schools.com](http://w3schools.com) (1)

### **4.1.2 Was ist Yii**

Yii ist ein high Performance PHP Framework welches vor allem für die Entwicklung im Web2.0 eingesetzt wird. Web 2.0 fördert die User aktiv im Web mitzumachen. Diese können eigenen Beiträge erstellen und diese auf der Website anzeigen lassen. Mehr dazu im Kapitel Yii2(12)

## **Alternative zu Frameworks**

Yii kann sehr weitreichend eingesetzt werden. Mit dem richtigen Wissen und Fähigkeiten kann man alles was mit einer PHP Seite möglich ist ganz einfach in Yii2 umsetzen.

Allerdings sind Frameworks nicht Administratoren freundlich da sie sehr viel Vorwissen erfordern um diese richtig zu implementieren. Einfacher zu implementieren sind CMS Systeme. Es gibt sehr viele Große CMS Systeme zum Beispiel:

- Joomla
- Wordpress
- Drupal
- Contao

Diese haben wir auch schon im Unterricht besprochen und damit Websites erstellt. Vorteile sind vor allem die einfache Implementierung und rasche Einrichtung einer Website. Auch SEO wird von den CMS Systemen vereinfacht. Nachteile sind allerdings oft eingeschränkte Möglichkeiten und Grenzen welches das CMS setzt.

## **Warum haben wir uns für Yii2 entschieden**

Der Hauptgrund warum wir uns gegen CMS Systeme entschieden haben sind die eingeschränkten Möglichkeiten die wir damit hätten. Bei Yii2 können wir die gesamte Website nach unseren Bedarf zusammenstellen und auch so bearbeiten wie wir es wollen. Es war uns auch wichtig das wir nach modernen Entwurfsmustern arbeiten. Siehe MVC - Model, View, Controller.

Wir hätten uns auch für andere Frameworks entscheiden können allerdings war



uns Yii2 schon bekannt und wir haben damit schon einige Websites erstellt.

Alternativen für Yii2 sind:

- PureMVC
- Laravel

### **4.1.3 PureMVC**

PureMVC ist seit dem Release in 2008 unverändert. Das hat den Vorteil, dass der administrative Aufwand sehr gering ist aufgrund nicht vorhandener Updates. Außerdem muss man das Framework nur einmal lernen und kann dieses dann meistern, ohne irgendwelche Änderungen zu befürchten. Es gibt auch Best-Practice Beispiele in vielen verschiedenen Sprachen. Diese findet man auf der Website (7)

### **4.1.4 Laravel**

Laravel

## **4.2 Technologien für die Datenauslesung**

- Gas Sensor - Für Luftqualität
- Wärmesensor - Für Raumtemperatur
- Luftfeuchtigkeit - Für Luftqualität
- Schalter - Für Futtermenge
- kinetischer Motor - für geregelte Luftzufuhr
- WLAN - Für Datenübertragung

### **4.2.1 Gas Sensoren**

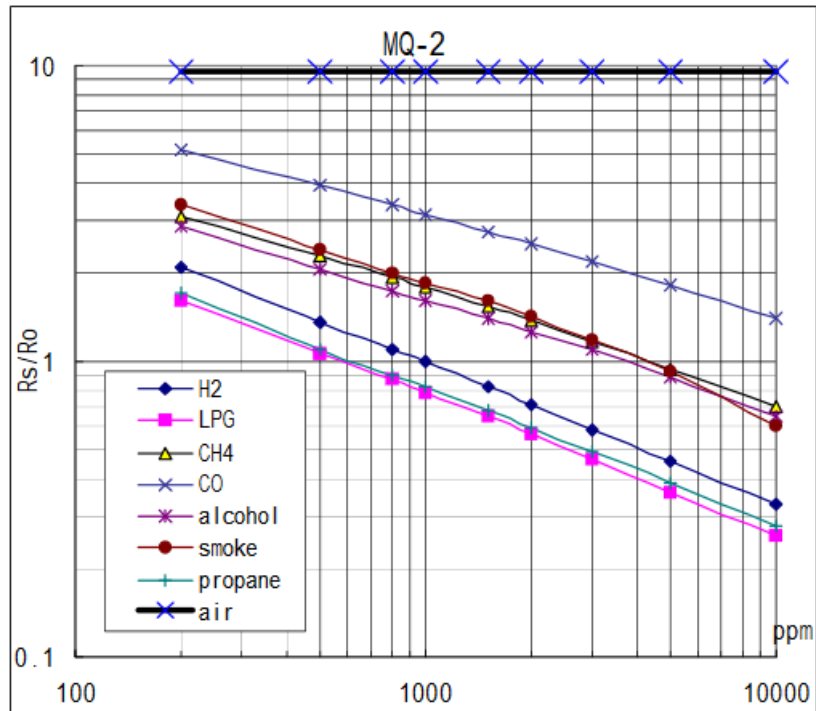
Für die Auswertung der Gaswerte stehen uns mehrere Module zur Verfügung. Zum einen stand uns die vielfältige MQ-Serie zur Verfügung oder der spezialisierte CCS811 von Adafruit.

In der Schule haben wir nicht nur den MQ2 Sensor zur Verfügung bereitgestellt bekommen, sondern auch den Adafruit CCS811. Wir bedanken uns dafür vielmals.

#### **MQ Gas Sensoren**

- MQ2 Methane, Butane, LPG, smoke
- MQ3 Alcohol, Ethanol, smoke
- MQ4 Methane, CNG Gas
- MQ5 Natural gas, LPG
- MQ6 LPG, butane gas

- MQ7 Carbon Monoxide
- MQ8 Hydrogen Gas
- MQ9 Carbon Monoxide, flammable gasses
- Mehr gibt es auf der Website: (4)



(5)

(4)

Im Datasheet (5) kann man herauslesen das der Sensor MQ2 (4) H2, LPG, CH4, CO, Alkohol, Rauch und Propan in einem Bereich von 200 bis 10000 Parts per million (Anteil pro Million) messen kann. Wie empfindlich der Sensor ist, hängt von den RS und RO werten ab.

- RS: Sensor Widerstand bei verschiedenen Konzentrationen von Gas
- RO: Sensor Widerstand bei 1000ppm von H2 bei sauberer Luft.

Da der MQ-2 auf so viele Gase außer Co2 reagiert haben wir uns entschlos-

sen eine Alternative zu suchen. Glücklicherweise hat uns die Schule auch den CCS811 von Adafruit zur Verfügung stellen können,

### **Adafruit CCS811**

In unserem Projekt haben wir uns letztendlich für den CCS811 von Adafruit entschieden. Im Gegensatz zu dem MQ-Sensoren erlaubt der CCS811 eCO<sub>2</sub> in einem Bereich von 400 bis 8192 ppm (parts per million) auszulesen. ( ? )

Dadurch können wir uns komplizierte mathematische Formelrechnungen sparen und direkt den gewünschten Datentypen verwenden.

### **4.2.2 Wärmesensor**

#### **DHT11**

In dem Arduino Uno Set war der DHT 11 Wärme und Temperatur Sensor von Adafruit direkt mitgeliefert. Dieses Modul erlaubt es uns digitale Daten in einem zwei Sekunden Takt auszulesen. Es liest die Wärme in einem Bereich von 0 bis 50°C, die Daten können allerdings um etwa 2°C variieren.

#### **CCS811**

Der CCS811 besitzt auch einen Thermistor, mit welchem wir direkt die Raumtemperatur auslesen können. Da der DHT11 allerdings auch die Luftfeuchtigkeit anzeigen kann, verwendeten wir diesen anstatt den CCS811. ( ? )

### **4.2.3 Luftfeuchtigkeit**

Da der DHT direkt bei dem Arduino Kit mit dabei war, gab es keinen Grund einen anderen Sensor zu suchen. Das Modul erlaubt uns die Luftfeuchtigkeit in einem Bereich von 20 bis 80 Prozent zu messen. Diese Daten können allerdings um beinahe 5% variieren.

### **4.2.4 kinetischer Motor**

Für eine automatisierte Luftzufuhr verwenden wir den Servo-Mikrocontroller SG90. Sowie der DHT 11 war dieser Motor schon im Arduino Uno Set beigelegt. Der 3-polige Servo wird für die Luftzufuhr verwendet. Der Propeller am drehenden Motor öffnet und schließt die Luftklappe des Kastens. Das ermöglicht eine höhere Kontrolle und Regulierung des CO2 Wertes. (<https://servodatabase.com/servo/towerpro>)

### **4.2.5 Hebel**

Um zu sehen wie viel Futter für die Insekten vorhanden ist, verwendeten wir einen Joystick. Dieser Joystick war beim Arduino Uno Set mitgeliefert und kann die X- und Y-Achse ablesen sowie auf Druck zu reagieren. Anhand der Y-Achse kann man den Joystick als Hebel verwenden um den momentanen Futterstand abzulesen.

### **4.2.6 Datenübertragung**

# 5 Problemanalyse

## 5.1 USE-Case-Analyse

- UseCases auf Basis von Benutzerzielen identifizieren:
  - Benutzer eines Systems identifizieren
  - Benutzerziele identifizieren (Interviews)
  - Use-Case-Liste pro Benutzer definieren
- UseCases auf Basis von Ereignissen identifizieren:
  - Externes Event triggert einen Prozess
  - zeitliches Event triggert einen Prozess (Zeitpunkt wird erreicht)
  - State-Event (Zustandsänderung im System triggert einen Prozess)
- Werkzeuge:
  - USE-Case-Beschreibungen (textuell, tabellarisch)
  - USE-Case-Diagramm
  - Aktivitätsdiagramm für den Use-Case (Interaktion zwischen Akteur und System abbilden)
  - System-Sequenzdiagramm (Spezialfall eines Sequenzdiagramms: Nur 1 Akteur und 1 Objekt, das Objekt ist das komplette System, es geht um die Input/Output Requirements, die abzubilden sind)

## **5.2 Domain-Class-Modelling**

- "Dinge" (Rollen, Einheiten, Geräte, Events etc.) identifizieren, um die es im Projekt geht
- ER-Modellierung oder Klassendiagramme
- Zustandsdiagramme (zur Darstellung des Lebenszyklus von Domain-Klassen darstellen)

## **5.3 User-Interface-Design**

- Mockups
- Wireframes

# 6 Systementwurf

## 6.1 Architektur

### 6.1.1 Design der Komponenten

Unsere Webapp ist in Yii2 programmiert und ist deshalb nach dem MVC Muster aufgebaut.

### 6.1.2 MVC - Model, View, Controller

#### Was ist MVC?

MVC, auch Model, View, Controller ist ein modernes Entwurfsmuster das meist für Anwendungen die ein User Interface beinhalten. Zum Beispiel für PHP, JAVA, C# und Ruby Anwendungen. (11)



## **Vor- und Nachteile**

### Vorteile

Gleichzeitiges Programmieren

Hohe Kohäsion (9)

Lose Kopplung (10)

### Nachteile

Schlechte Übersicht

Konsistente Programmierung notwendig

Steile Lernkurve

Das MVC Entwurfsmuster ist sehr weit verbreitet und hoch angesehen, es wird deshalb in den meisten Anwendungen mit User Interface angewandt. Der Grund für die Verwendung dieses Musters sind die vielen Vorteile die es bietet. Außerdem können viele Nachteile über Frameworks behoben werden.

## **Aufbau von MVC**

Im Grunde arbeitet MVC mit 3 Klassen:

- Modell
- View
- Controller

Die unterschiedlichen Klassen haben unterschiedliche Aufgaben.

## **View**

Die View Klasse ist nur zum Anzeigen der einzelnen Teile des User Interfaces. Bei Webseiten zum Beispiel die Index Seite oder das Login Formular. Außerdem nimmt es Benutzer Eingaben entgegen.

## **Controller**

Die Controller Klasse Steuert die ganze Anwendung, sie nimmt die User eingaben von der View Klasse entgegen und verarbeitet diese. Außerdem ändert sie auch die View Klasse um andere Seiten anzuzeigen. Sie nimmt auch die Daten von der Modell Klasse entgegen, verarbeitet diese und gibt sie an die View Klasse weiter.

## **Modell**

Die Modell Klasse enthält die darzustellenden Daten, sie ist unabhängig von der View und der Controller Klasse

### **6.1.3 CRUD - Create, Read, Update, Delete**

#### **Was ist CRUD?**

CRUD sind die 4 grundlegenden Aufgaben einer Datenbankbindung:

- Create - Erstellen neuer Datensätze
- Read - Auslesen der Datensätze
- Update - Aktualisieren von vorhandenen Datensätze
- Delete - Löschen von Datensätzen

(8)

## **Vor- und Nachteile**

Vorteile    Nachteile

### **6.1.4 Yii2**

#### **Was ist Yii**

Yii ist ein high Performance PHP Framework welches vor allem für die Entwicklung im Web2.0 eingesetzt wird. Web 2.0 fördert die User aktiv im Web mitzumachen. Diese können eigenen Beiträge erstellen und diese auf der Website anzeigen lassen.(12)

#### **Gii**

Gii ist der Yii eigene Model, Crud, Controller, Form, Module und Extension Generator. Mit ihm kann man sehr einfach eine Model Klasse mit einer Unterliegenden Datenbanktabelle erstellen. Aus dieser Model klasse kann man dann wiederum CRUD befehle erzeugen. Mit Gii kann man die gesamte Grund MVC Struktur für Yii2 erzeugen und im weiter verlauf dann nach eigenen Wünschen verändern.

## **Vor- und Nachteile**

Yii hat sehr viele Vorteile, allerdings auch einige Nachteile: Vorteile sind:

- CRUD-Creator (Gii)
- Model Generator (Gii)

- Einfache Implementierung von HTML Formulare
- Einfach Datenbankzugriffe

Nachteile ist der Hohe Setup aufwand und die benötigten Kenntnisse in PHP und SQL.

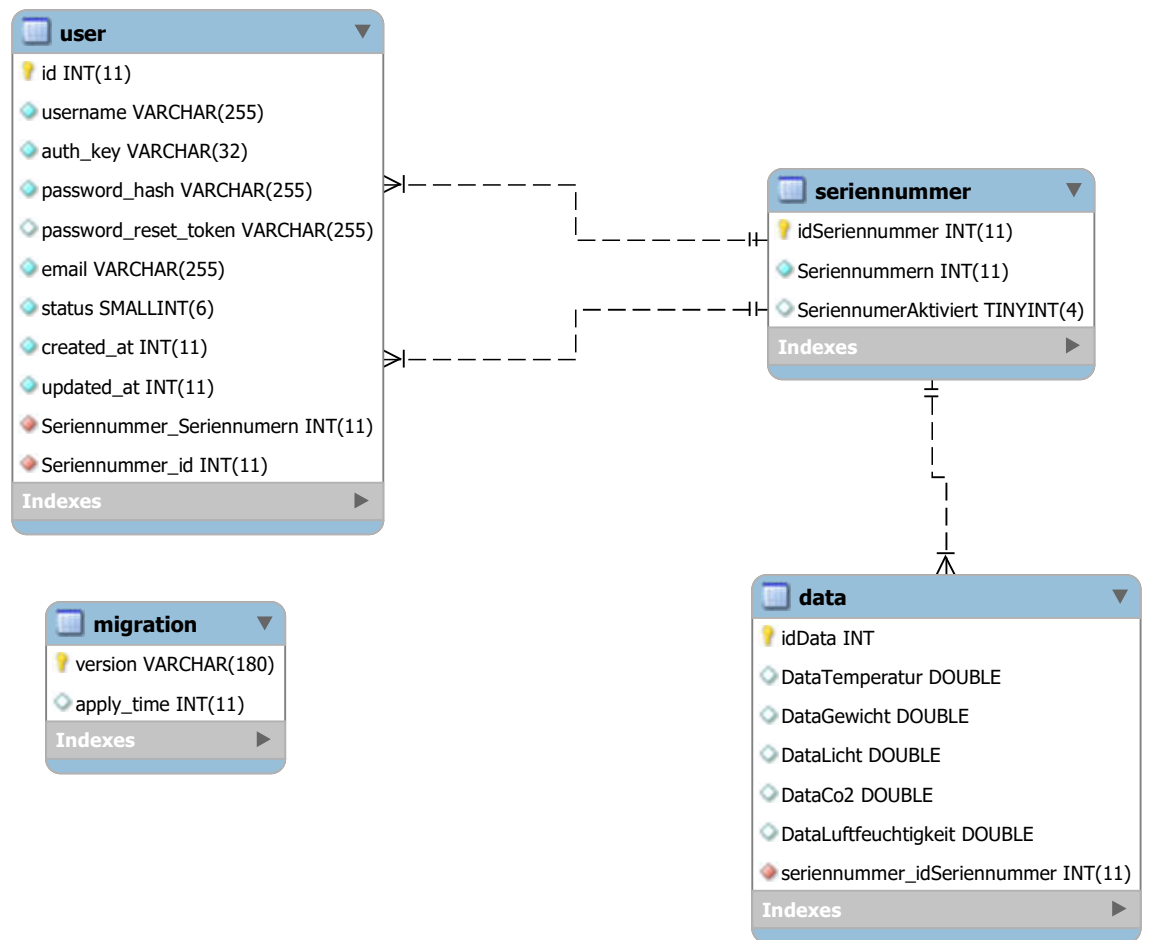
### **6.1.5 Benutzerschnittstellen**

Die Benutzerschnittstelle unseres Brutkasten besteht hauptsächlich über die Webapp. Dort kann man sich anmelden und die Daten für den eigenen Brutkasten auslesen. Um sicherzugehen das man nur die eigenen Daten sieht, muss man sich auf der Website mit der Seriennummer auf dem Kasten Registrieren, und dadurch auf die Daten zugreifen welche für die jeweilige Seriennummer gespeichert sind zugreifen.

Wenn man noch nicht registriert wird kann man auf der Website einen Link zu unserem Webshop sehen, den wir allerdings nicht in diesem Projekt erstellen werden.

### **6.1.6 Datenhaltungskonzept**

Hier ist unser ER-Diagramm



ten bis zu 7 Tage Speichern. Daten sollen alle 5-10 Minuten von unserem Brutkasten übermittelt werden. Zur Übermittlung werden wir ein WLAN Modul an unserem Brutkasten anbringen. Mehr dazu im Kapitel Datenübermittlung. Für Prototyp Zwecken werden wir die Daten vorerst auf einen Raspberry Pi 3 speichern.

### **6.1.7 Konzept für Ausnahmebehandlung**

- Systemweite Festlegung, wie mit Exceptions umgegangen wird
- Exceptions sind primär aus den Bereichen UI, Persistenz, Workflow-Management

### **6.1.8 Sicherheitskonzept**

Beschreibung aller sicherheitsrelevanten Designentscheidungen

- Design der Security-Elemente
- Design von Safety-Elementen (Fehlertoleranz, Verfügbarkeit etc.)

### **6.1.9 Design der Testumgebung**

- wie wird getestet (Unit-Testing, Integrationstesting, Systemtests, Akzeptanztests)
- Testumgebung, Testprozess, Teststrategie, Testmethoden, Testfälle

### **6.1.10 Desing der Ausführungsumgebung**

- Deployment (DevOps)
- Betrieb (besonders Hoch- und Hertunerfahren der Anwendung)

## **6.2 Detailentwurf**

Design jedes einzelnen USE-Cases

- Design-Klassendiagramme vom Domain-Klassendiagramm ableiten (incl. detaillierter Darstellung und Verwendung von Vererbungshierarchichen, abstrakten Klassen, Interfaces)
- Sequenzdiagramme vom System-Sequenz-Diagramm ableiten
- Aktivitätsdiagramme
- Detaillierte Zustandsdiagramme für wichtige Klassen

Verwendung von CRC-Cards (Class, Responsibilities, Collaboration) für die Klassen

- um Verantwortlichkeiten und Zusammenarbeit zwischen Klassen zu definieren und
- um auf den Entwurf der Geschäftslogik zu fokussieren

Design-Klassen für jeden einzelnen USE-Case können z.B. sein:

- UI-Klassen
- Data-Access-Klassen
- Entity-Klassen (Domain-Klassen)
- Controller-Klassen
- Business-Logik-Klassen

- View-Klassen

Optimierung des Entwurfs (Modularisierung, Erweiterbarkeit, Lesbarkeit):

- Kopplung optimieren
- Kohäsion optimieren
- SOLID
- Entwurfsmuster einsetzen



# 7 Implementierung

Detaillierte Beschreibung der Implementierung aller Teilkomponenten der Software entlang der zentralsten Use-Cases:

- GUI-Implementierung
- Controllerlogik
- Geschäftslogik
- Datenbankzugriffe

Detaillierte Beschreibung der Teststrategie (Testdriven Development):

- UNIT-Tests (Funktional)
- Integrationstests

Zu Codesequenzen:

- kurze Codesequenzen direkt im Text (mit Zeilnnummern auf die man in der Beschreibung verweisen kann)
- lange Codesequenzen in den Anhang (mit Zeilennummer) und darauf verweisen (wie z.B. hier)

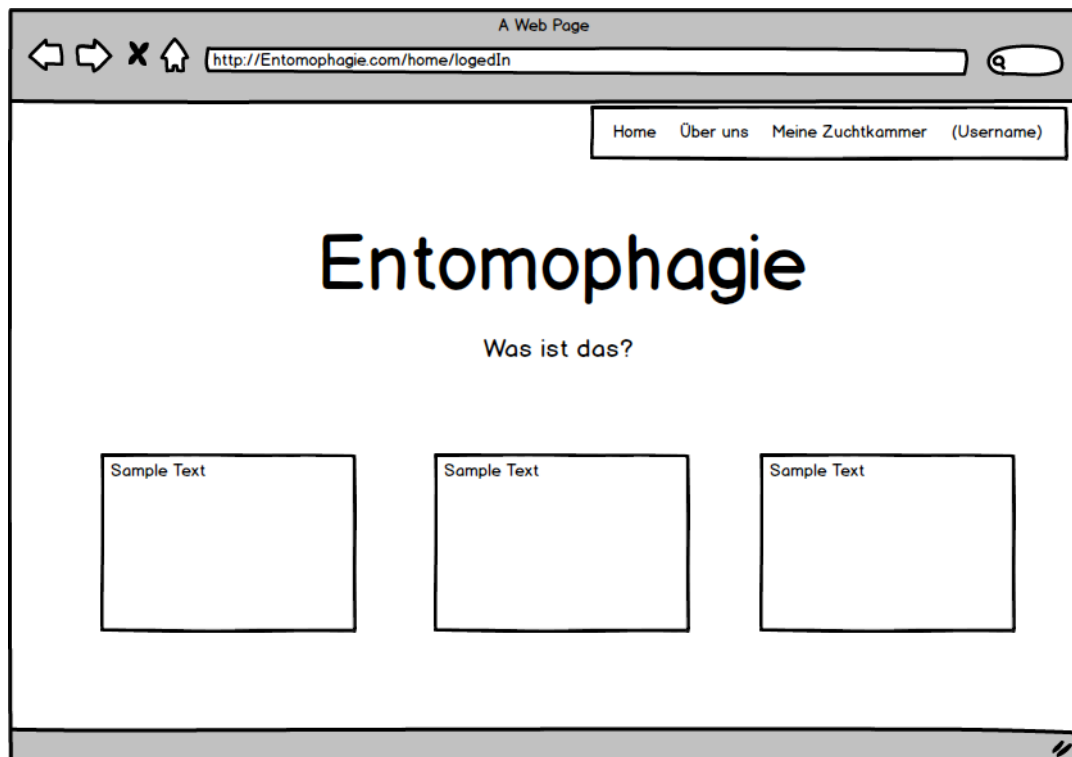
## **7.1 Webapp**

Für unser Projekt erstellen wir eine Webapp mit der man die Daten seiner eigenen Zuchtkammer anzeigen lassen kann. Wir haben geplant das man sich mit der Seriennummer der Box Registrieren kann und dann am Handy über eine Webapp alle Daten anzeigen lassen kann. Folgende Daten sollte man auslesen können:

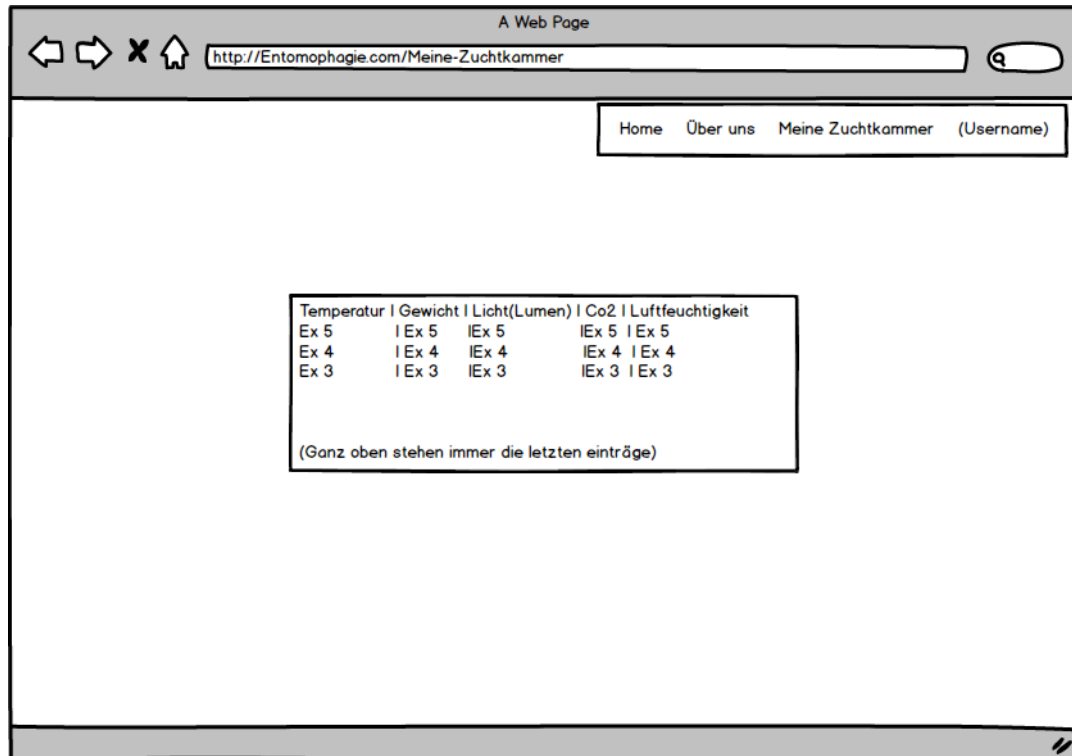
- Sauerstoff
- Luftfeuchtigkeit
- Gewicht
- Temperatur
- Futtermenge
- ungefähre Zeit bis zu Reife

Als Grundlage für die Website haben wir das Framework Yii2 verwendet. Mehr dazu im Kapitel Yii2.

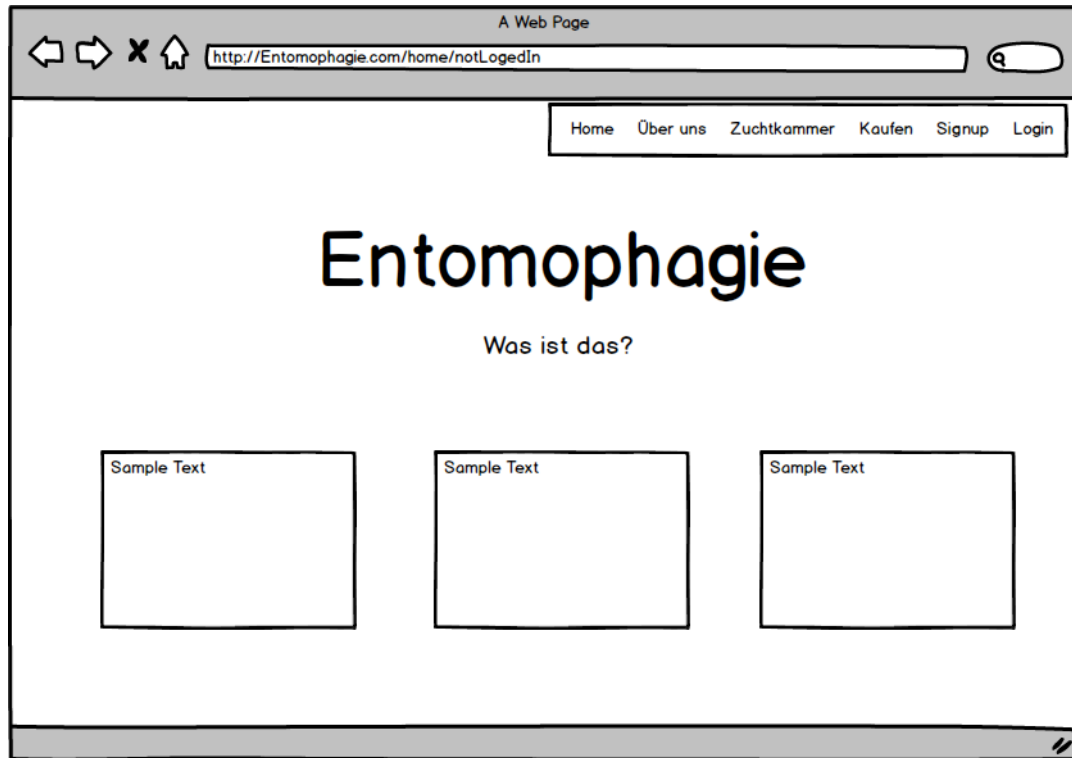
### 7.1.1 Mockups



Hier sieht man die Ansicht wenn man auf unserer Website Angemeldet ist. Man kann auf seine eigene Zuchtammer zugreifen indem man auf den Button "Meine Zuchtammer" klickt und dort die Daten auslesen.



Auf der Seite "Meine Zuchtkammer" sieht die letzten 10 Daten die meine Zuchtkammer wiedergegeben hat. Diese sind in Absteigender Reihenfolge geordnet, dass heißt, dass der letzte Eintrag ganz oben steht.



Hier Sieht man die Ansicht wenn man auf unserer Website nicht Angemeldet ist. Man kann über den Button "Zuchtkammer" mehr über unsere Zuchtkammern herausfinden, weiter kann man über den Button "Kaufen" eine eigene Zuchtkammer kaufen. Bei Signup kann man sich als neuer Benutzer registrieren, um sich allerdings zu registrieren braucht man zuerst eine Seriennummer die man beim Kauf einer Zuchtkammer erhält.

### **7.1.2 Datenbankzugriffe**

Unserer Datenbank zugriffe werden von unserem Framework verarbeite, dabei verwendet das Framework CRUD befehle und nach dem MVC Muster. Das heißt es gibt ein unterliegendes Modell welches Daten unserem Controller mitgibt welche wiederum die View erzeugt.

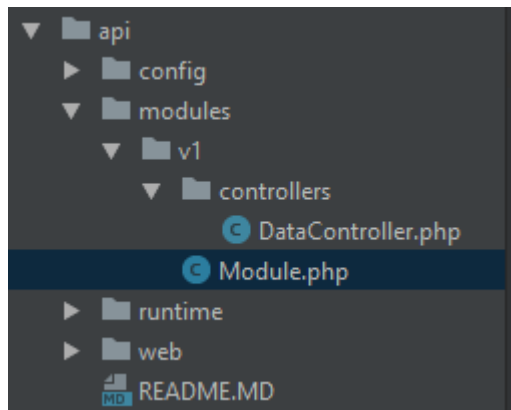
Der gesamte Datenbankzugriff kann mittels Yii2 sehr einfach erstellt werden. Mehr dazu siehe Gii

### **7.1.3 Datenübermittlung**

Um die Daten von unsrem Arduino an unsere Datenbank zu senden werden wir in den Brutkasten ein WLAN Modul einbauen und die Daten alle 5-10 Minuten als JSON-Format an unsere REST-Schnittstelle unserer Webapp senden. Das hat den Vorteil das unsere Daten direkt von unserem Brutkasten an die Datenbank gegeben werden kann.

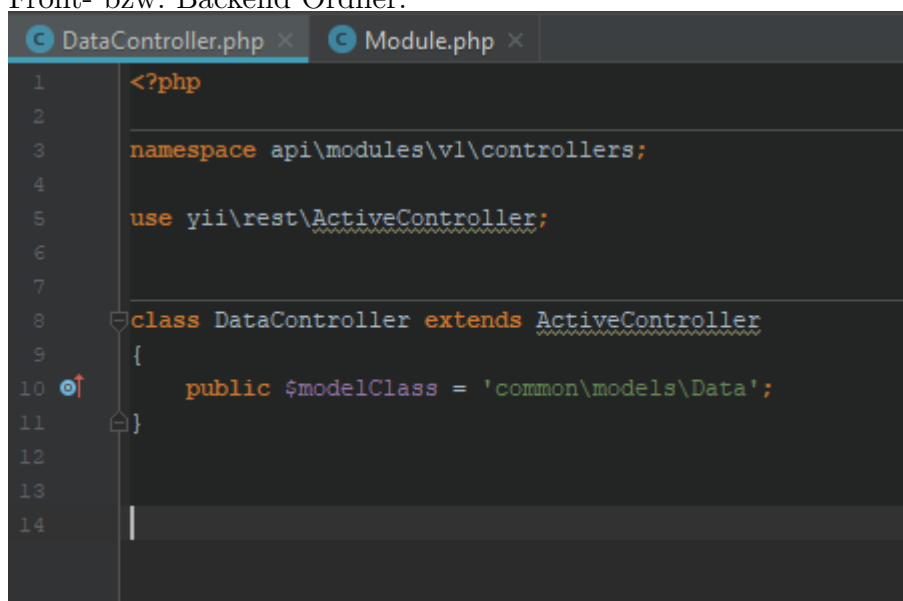
Für diese Lösung brauchen wir eine REST Schnittelle in unserer Webapp. Dies kann über YII2 sehr einfach realisiert werden. Yii2 hat eine vor implementierte REST Schnittelle welche man nur noch aktivieren muss. Die Daten werden aus der Modell Klasse ausgelesen und über den Controller an die REST-View weitergeleitet. Die ganze Implementierung sind 5-10 Zeilen Code. Die Code Sequenz sehen sie im Kapitel Rest Schnittelle.

## 7.1.4 Rest Schnittelle



Hier sieht man die Ordnerstruktur. Wir erstellen im Obersten Verzeichnis einen neuen Ordner namens 'api' um im Web über folgende URL: 'www.entomophagie/api/web/v1/data' auf unsere Restschnittelle Zugreifen können. Dabei müssen wir darauf achten das wir Groß- und Kleinschreibung beachten. Um dieses Problem zu lösen ist es am einfachsten alle Ordner in unserem Verzeichnis immer klein zuschreiben.

Weiters Kopieren wir 'web', 'config' und den 'runtime' Ordner aus unserem Front- bzw. Backend Ordner.



Hier sieht man die Controller Klasse. In der Controller Klasse müssen wir den

Yii eigenen ActiveController einbinden und die Modell Klasse festlegen welche die Daten ausliest.

```
1 <?php
2 namespace api\modules\v1;
3
4 class Module extends \yii\base\Module
5 {
6     public $controllerNamespace = 'api\modules\v1\controllers';
7
8     public function init()
9     {
10         parent::init();
11     }
12 }
13
```

Hier sieht man die Module Klasse. In der Module Klasse wird unser neuer Rest Controller initialisiert, damit unsere Anwendung diese Verwenden kann. Dort müssen wir unseren Controller den wir zuvor erstellt haben festlegen.

Wir haben den Rest Controller mit folgendem Online Tutorial erstellt (6).



## **7.2 Verkablung**

Die Verkablung der einzelnen Module ist aus verschiedenen Gründen wichtig. Zum einen braucht jedes Modul genügend Strom um seine Aufgabe zu erfüllen, gleichzeitig würde es aber ohne die Erdung zu einem Kurzschluss kommen. Des Weiteren ist es wichtig, dass die Module Daten auslesen und weiterleiten können.

### **7.2.1 Aufbau**

(Bild einfügen)

In Abbildung x können wir nun den Arduino und das Breadboard sehen. Dort kann man direkt die drei Farben erkennen, die jeweils für Stromzufuhr (blau), Datenauslesung (grün) und Erdung (braun) stehen.

(Bild mit Stromfluss einfügen)

Hier sieht man, wie der Strom im Breadboard fließt. Das heißt, wenn man in der untersten Reihe das Board mit 5 Volt versorgt, sind alle weiteren Steckplätze verwendbar. Wichtig hierbei ist, dass das Kabel für die Stromzufuhr auch auf der richtigen Reihe platziert wurde. Das erkennt man bei dieser Leiterplatte an dem Plus und Minus Zeichen. Plus steht hierbei für Strom und Minus für die Erdung.

Falls man das falsch verkabelt, funktioniert das Modul nicht, es werden allerdings keine Schäden angerichtet.

(Bild mit analog und digitalwerten einfügen)

Der Arduino hat zwei Möglichkeiten, Daten auszulesen. Einmal spricht man hier

von digitaler und analogen Datenauslesung. (internet suche) Digitale Werte bedeutet hier allerdings nur das der ausgelesene Wert als 0 oder 1 abgespeichert werden kann, analoge Daten sind bei diesem Punkt anders, da diese einen Wert zwischen 0 und x haben können (zitat)

## **7.3 Programmierung**

Um die Daten der einzelnen Sensoren verwenden zu können, müssen diese ausgelesen und sinnvoll wiedergegeben werden. Nur sobald die Sensoren richtig programmiert und verkabelt sind, kriegt man lesbare Information. Diese können anschließend weiter interpretiert und verwendet werden.

Die verwendeten Sensoren:

- DHT11
- CCS811
- Hebel
- Servo SG90
- photocell

### **7.3.1 DHT11 & CCS811**

Für den DHT 11 sowie dem CCS811 gibt es eine von Adafruit frei verwendbare Bibliothek. Mithilfe von dieser kann man ein Objekt erstellen und es mithilfe dieses Objektes die Momentane Temperatur/Luftfeuchtigkeit auslesen.

```
#include "DHT.h"
#define DHTPIN 2      // what digital pin we're connected to
#define DHTTYPE DHT11 // DHT 11
DHT dht(DHTPIN, DHTTYPE);
```

Das `#include` fügt die Klasse in unser Projekt ein und erlaubt es uns das Objekt `dht` sowie `ccs` zu erstellen.

Das `#define` legt des weiteren fest, dass relevante Daten, im Falle des DHT Moduls, von Pin 2 kommen.

```
//Adafruit
Serial.println("CCS811 test");
if(!ccs.begin()){
    Serial.println("Failed to start sensor! Please check your wiring.");
    //while(1);
}
delay(10);
```

Die Setup Funktion, wird für einmalige Dinge wie die Kalibrierung, dem Starten des Serial Monitors oder der Verbindung mit dem Raspberry verwendet. Der CO2 Sensor benötigt das Setup, da es zuerst kalibriert werden muss um die Luftqualität zu messen.

```
if(ccs.available()){
    if(!ccs.readData()){
        Serial.print("CO2: ");
        Serial.println(ccs.geteCO2());
    }
    else{
        Serial.println("ERROR!");
        //while(1);
    }
}
```

Die Loop Funktion, wiederholt sich im vorgegebenen Rhythmus immer wieder und gibt keine Variablen zurück. Dort können werden alle Daten ausgelesen und an das WLAN Modul weitergeschickt bzw. in diesem Fall wird es auf den Serial Monitor ausgegeben.

Die WENN abfrage prüft als erstes ob der CO2 Sensor kalibriert wurde oder nicht. Falls es gelungen sein sollte gibt es die Werte mit dem Befehl `Serial.println(ccs.getCO2());` im Serial Monitor aus.

### **7.3.2 Hebel**

Das Joystick bzw. der Hebel konnte direkt abgelesen werden. Wichtig hierbei ist es das man nicht nur eine Achse im Setup verwendet sondern beide, ansonsten kommt es bei dem Loop zu einem Fehler und der Wert der benötigten Achse verändert sich nicht.

### **7.3.3 SG90**

Die vom verwendeten Bibliotheken waren bei der Installierung der Arudino eigenen IDE direkt dabei. Die Servo kontrolliert die Luftzuvor und muss daher Werte vom CCS811 wiederverwenden

```
//Servo
  if(ccs.getCO2() <= co2Min){

    //move the micro servo from 0 degrees to 180 degrees
    for(;servoAngle < 180; servoAngle++) {
      servo.write(servoAngle);
      delay(10);
    }
  } if (ccs.getCO2() > co2Min && servoAngle != 0){
    servo.write(45);
    servoAngle = 0;
    Serial.println("RETURN");
  }
```

Die If Abfrage prüft ob CO2 einen Mindestwert (co2Min) unterschreitet. Falls das passiert wird eine Schleife ausgeführt die den Servo 180° dreht. Diese 180° würde die Lüftungsklappe aufhalten. Falls der CO2 Wert wieder in einem akzeptablen Bereich liegt und die Position des Motors nicht null beträgt, wird die zweite Schleife aktiviert die den Motor zur Position 0 zurückbringt

#### 7.3.4 photocell

# 8 Deployment

- Umsetzung der Ausführungsumgebung
- Deployment
- DevOps-Thema

# 9 Tests

## 9.1 Systemtests

Systemtests aller implementierten Funktionalitäten lt. Pflichtenheft

- Beschreibung der Teststrategie
- Testfall 1
- Testfall 2
- Testfall 3
- ...

## 9.2 Akzeptanztests

# **10 Projektevaluation**

siehe Projektmanagement-Unterricht



# **11 Benutzerhandbuch**

falls im Projekt gefordert

# **12 Betriebswirtschaftlicher Kontext**

BW-Teil

# 13 Zusammenfassung

- Etwas längere Form des Abstracts
- Detaillierte Beschreibung des Outputs der Arbeit

# Literaturverzeichnis

- [1] W3schools. URL: <https://www.w3schools.com/>.
- [2] Web hypertext application technology working group. URL: <https://whatwg.org/>.
- [3] World wide web consortium. URL: <https://www.w3.org/>.
- [4] Arduino. Mqgassensor. URL: <https://playground.arduino.cc/Main/MQGasSensors>.
- [5] LTD HANWEI ELETRONICS CO. Technical data mq-2 gas sensor. URL: <https://www.mouser.com/ds/2/321/605-00008-MQ-2-Datasheet-370464.pdf>.
- [6] Budi Irwan. Setup restful api in yii2, July 2014. URL: <http://budiirawan.com/setup-restful-api-yii2/>.
- [7] PureMVC. Implementation idioms and best practices. URL: <http://puremvc.org/>.
- [8] Wikipedia. Crud. URL: <https://de.wikipedia.org/wiki/CRUD>.
- [9] Wikipedia. Kohäsion. URL: [https://de.wikipedia.org/wiki/Koh%C3%A4sion\\_\(Informatik\)](https://de.wikipedia.org/wiki/Koh%C3%A4sion_(Informatik)).

- [10] Wikipedia. Lose kopplung. URL: [https://de.wikipedia.org/wiki/Lose\\_Kopplung](https://de.wikipedia.org/wiki/Lose_Kopplung).
- [11] Wikipedia. Model view controllers. URL: [https://de.wikipedia.org/wiki/Model\\_View\\_Controller](https://de.wikipedia.org/wiki/Model_View_Controller).
- [12] Wikipedia. Web 2.0. URL: [https://en.wikipedia.org/wiki/Web\\_2.0](https://en.wikipedia.org/wiki/Web_2.0).