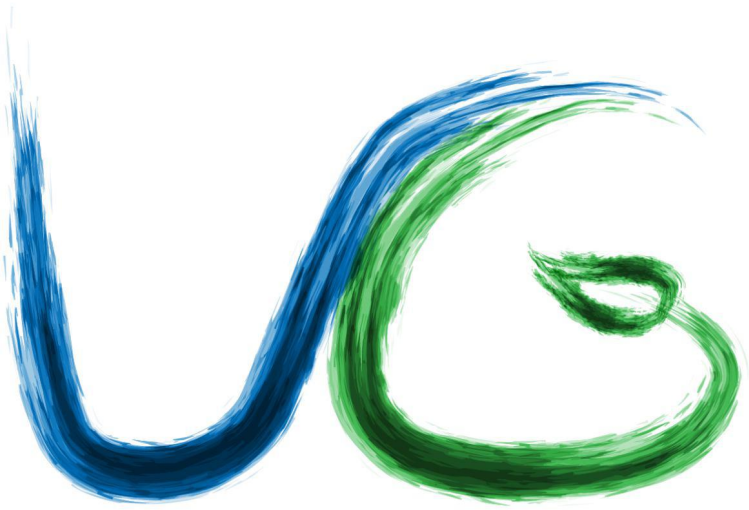


TGM - HTBLuVA Wien XX
IT Abteilung

Diplomarbeit
Urban Green

Ramin Bahadoorifar
Matthias Schwebler
Samuel Schober
Konrad Kelc

Version: January 30, 2017 at 16:40

Contents

Abstract

Abstract DE auf enlisch übersetzen

Kurzfassung

Dieses Diplomprojekt, wird in Kooperation mit der Firma "Ponix Systems" durchgeführt und handelt über die Entwicklung eines Low-Cost Prototypen eines Aquaponik Systems für den Heimgebrauch um dem globalen Trend der Umweltverbesserung zu folgen. Bei einem Aquaponik System handelt es sich um ein in sich geschlossenes Ecosystem, in dem Nutzpflanzen mit Wasser aus einem Aquarium bewssert und zustzlich mit spezieller Beleuchtung behandelt werden, um eine stark reduzierte Wachstumszeit zu erzielen. Hauptziel unseres Projektes ist es, die derzeitige Marktlücke von Aquaponik Systemen in Mitteleuropa zu füllen. Dafür wird ein pflegeleichtes Ökosystem - bestehend aus einem Aquarium und einem Beet - entwickelt und auf die Überwachung und Automatisierung optimiert. Dazu werden diverse Sensoren, Aktoren und Single Board Computer verbaut, welche die Regulierung der Parameter des Systems übernehmen. Die Daten des Aquariums bzw. der Pflanzen werden an einen zentralen Server geleitet und dort persistiert um gegebenenfalls gewünschte Diagramme und Graphen zu erstellen. Für die Persistierung wird auf Serverseite eine NoSQL Datenbank (MongoDB) verwendet und auf Seiten der Single Board Computer eine lightweight und schnelle In-Memory-Datenbank verwendet (RedisDB). Das Aquaponiksystem kann von einer Webapplikation aus beobachtet und gesteuert werden. Die dafr bentigten Daten werden an einem Zentralserver durch ein Socket in einem bestimmten Interval gesendet. Diese werden zeitlich angeordnet in der NoSQL Datenbank gespeichert und dem jeweiligen Benutzer zugeordnet, aus welchen man dann eine Zeitanalyse erstellen kann in Form eines Diagramms. Bei der Entwicklung der Webapplikation wurde das innovative Framework Angular 2 verwendet. Die Verbindung des Ökosystems zum Internet wird von einem Single Board Computer übernommen, der über einen kleinen Touchscreen bedient werden kann. Selbst bei Internetunterbrechungen zeichnet das System weiterhin Daten auf und sendet sie bei erneutem Internetzugriff mit korrektem Zeitstempel an den Server

Acknowledgements

We want to thank everyone.
- Schabel (Idee, Grundstein)
- Ponix Systems (Hardware, Softwareunterstützung)
- Koppensteiner (Bereitstellung des Arbeitsraums, Platz fr Aquarium usw.)
TODO

1 Einführung

1.1 Einleitung

1.2 Aufgabenstellung

1.3 Ziele und Zielgruppen

1.4 Konzept

2 Grundlagen und vorhandene Technologien

2.1 Methodiken zur Verbreitung

2.1.1 Citizen Science

2.1.2 Blidungssektor

2.2 Vorhandene Aquaponiksysteme

2.2.1 Grove Ecosystem

2.2.2 EcoQube C

2.2.3 Ponix Systems

3 Projektmanagement

3.1 Projektmanagement Methode

3.2 Teamstruktur

3.3 Aufgabenteilung

3.4 Terminplanung

3.5 User Stories

3.6 Sprint Dokumentation

4 Evaluierung der bentigten Technologien und Komponenten

4.1 Single Board Computer - User Story 1817

Um die Sensoren und Aktoren anzusprechen wird ein SBC (Single Board Computer) verwendet. Aufgrund der Anzahl an erhältlichen SBCs, wird der Fokus auf populäre Produkte mit einer großen Community gelegt. Ideal wäre ein SBC, der auf Messgeräte, wie sie im Aquarium verwendet werden, spezialisiert ist und ein WLAN-Interface, für die Datenübermittlusng hat. Des Weiteren sollte er einen internen Speicher bieten (zustzlich zum Betriebssystem 200MB fr Redis Datenbank + Daten) um im Falle eines Netzwerkausfalles, für einige Stunden alle Daten zwischenspeichern zu können.

4.1.1 Arduino

Der Arduino Uno ist ein SBC, welcher im Bereich von 20 - 25 zu erhalten ist. Er hat einen internen Speicher der groß genug ist, um einen Netzwerkausfall überbrücken zu können. Da er aber selbst keine Netzwerkschnittstelle hat, muss er mit einem WiFi-Shield (90) ausgestattet werden. Ein Shield ist eine Hardware-erweiterung, die dem Arduino situationsbezogene Funktionen hinzufügen kann. Für dieses Projekt ist das "Open Aquarium Board" (50) optimal, da es speziell entwickelt wurde um ein Aquarium zu überwachen bzw. zu steuern und eine Open Source API an-bietet.
Das Problem dabei ist, dass nur ein Shield auf dem Arduino angebracht werden kann. Das heißt, entweder das Open Aquarium Board wird verwendet, oder ein WLAN Shield zur Datenübermittlung. Er müsste mit einem anderen SBC, welcher Netzwerktauglich ist, verbunden werden um die Daten auf einem zentralen Server zu persisiteren.

4.1.2 Raspberry Pi B

Der Raspberry Pi B, ist einer der energieeffizienteren Computer seiner Art (240mA im Leerlauf). Er kann mithilfe einer SD Karte mit einem adäquaten Speicher ausgestattet und einfach mit der eingebauten RJ45 Buchse bzw. einem WLAN-Stick mit dem Internet verbunden werden. Ein solcher WLAN-Stick ist mit 10 im Vergleich zu dem WiFi-Shield billiger. Da aber der Raspberry Pi B keine API für jegliche Sensoren und Aktoren bietet, erfordert die Implementierung der Überwachung bzw. Steuerung des Aquaponic Systems einen Arbeitsaufwand der mit anderer Hardware umgangen werden kann.

4.1.3 Raspberry Pi Zero

Ebenso wie der Raspberry Pi B, hat der Pi Zero einen erweiterbaren Speicher und kann mit einem WLAN-Stick mit einem Lokalen Netzwerk verbunden werden. Der Unterschied ist, dass er keine RJ45 Buchse besitzt und um einiges weniger Rechnleistung. Der Vorteil dabei ist, dass er sehr wenig Strom verbraucht (65mA im Leerlauf).

4.1.4 Lösung

Aus Erfahrungswerten der Firma "Ponix Systems", ist das Open Aquarium Board für den Arduino unabdinglich. Die optimale Lösung ist also, ein Arduino mit entsprechendem Shield, der die Daten an einen Raspberry Pi übermittel, welcher diese dann an einen Server schickt.

4.2 Sensoren - User Story 1975

Die Sensoren für den Prototypen, werden von der Firma Ponix Systems zur verfügung gestellt. Diese müssen allerdings erst auf Funktionalität untersucht werden.
Für die Entwicklung, des Codes, zum Ansprechen der Sensoren, wird "Arduino IDE" verwendet. Diese IDE bietet die Möglichkeit, den Code auf einem rechenstarken PC zu kompilieren und ihn dann auf dem Arduino auszuführen. Zusätzlich kann sie verwendet werden, um alle eingehenden Daten des Arduinos zu überwachen. Der kompilierte Code wird beim Endprodukt auf dem Arduino gespeichert und sendet kontinuierlich Daten über die USB Schnittstelle an den Raspberry Pi.
Es werden Libraries von Arduino indirekt über die API von OpenAquarium verwendet.

4.2.1 Temperatur

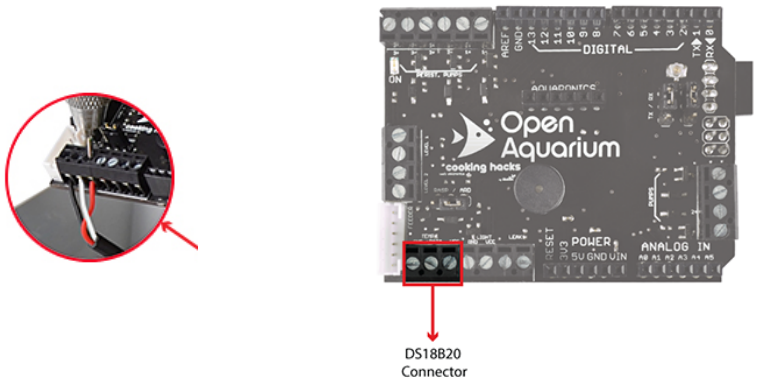
Zum Messen der Temperatur wird der Sensor von www.cooking-hacks.com verwendet (DS18B20):
Input: 3.0-5.5V
Arbeitsbereich: -55°C bis +125°C
Genauigkeit: ± 0.5°C von -10°C bis +85°C

Die Temperatur kann mit nur einem Befehl abgelesen werden. Dieser liefert einen Wert in Grad Celsius zurück.

```
OpenAquarium.init();
float temperature = OpenAquarium.readtemperature();
```

Die Testergebnisse des Temperatur Sensors stimmten im Bereich von 24°C, unter Berücksichtigung der Abweichung, mit denen eines handelsüblichen Thermometers überein.

Angeschlossen werden die Sensoren bzw. der Temperatursensor auf dem OpenAquarium Board:



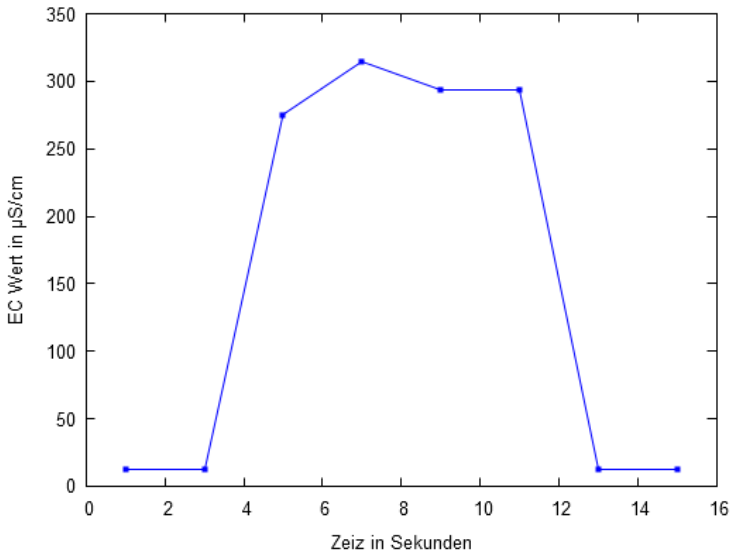
4.2.2 EC Wert

Der EC Wert (Electrical Conductivity) gibt ber die Leitfhigkeit des Wassers auskunft. Mit diesem Wert kann auf die Menge der im Wasser gelsten Minerale/Salze geschlossen werden. Desto mehr Nährstoffe also im Wasser vorhanden sind, umso höher ist auch der EC Wert.
Um diesen zu Messen wird der Sensor von www.cooking-hacks.com verwendet.

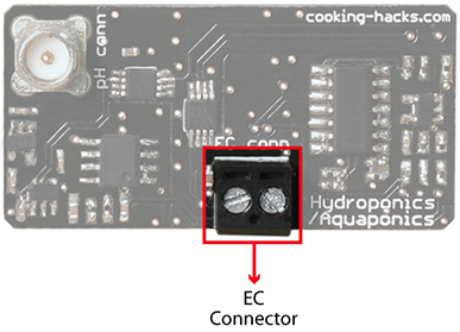
```
OpenAquarium.init();
OpenAquarium.calibrateEC(point_1_cond,point_1_cal,
                         point_2_cond,point_2_cal);

float resistanceEC = OpenAquarium.readResistanceEC();
float EC = OpenAquarium.ECConversion(resistanceEC); //EC Value in uS/cm
```

Die Tests beim Eintauchen des Sensors in Leitungswasser ergaben folgende Ergebnisse:



Wie erwartet, steigt der EC Wert beim Eintauchen ins Wasser, da es natürlich um einiges leitfähiger ist, als Luft. Angeschlossen wird der Sensor auf dem OpenAquarium Board:



4.2.3 PH Wert

4.2.3.1 Allgemein

Der PH Wert ist einer der wichtigsten Werte, wenn es um das Überwachen eines Aquaponik Systems geht. Dafür gibt es ein großes Spektrum an erhältlichen Sensoren. Die der unteren Preisklasse, benötigen eine regelmäßige Wartung. Die der oberen Preisklasse heben den Preis des Produktes, über die Zielgruppe hinaus, und sind deshalb unmbrauchbar.
Da der Sensor ein analoges Signal liefert, muss die gemessene Spannung erst umgerechnet werden. Dies übernimmt die Klasse OpenAquarium.

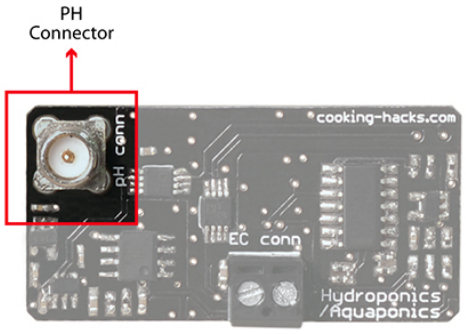
```
OpenAquarium.calibratepH(cp4,cp7,cp10);
int mvpH = OpenAquarium.readpH(); //Value in mV of pH
float pH = OpenAquarium.pHConversion(mvpH);
```

cp4, cp7, cp7 ... Calibrationpoints (Konstanten)

4.2.3.2 Calibrationpoints

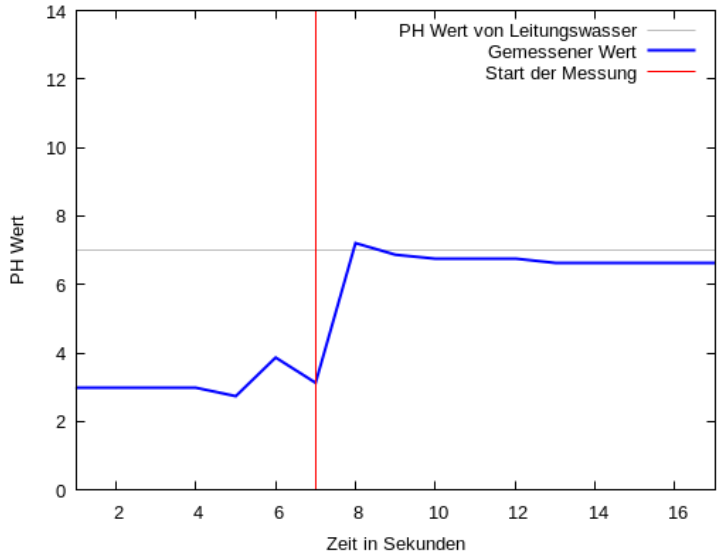
Calibrationpoints (Kalibrierungswerte), werden angepasst, wenn der gemessene PH Wert zu stark vom eigentlichen Wert abweicht. Um die Abweichung des Sensors zu beheben, sind Kalibrierungskits vorgesehen, die allerdings bis zu ein mal im Monat verwendet werden müssen. Dies belastet den Kunden mit zusätzlichen Kosten und Aufwand und soll daher umgangen werden.

Angeschlossen wird der Sensor auf dem OpenAquarium Board.



4.2.3.3 Tests

Beim testen eines bereits länger in Betrieb (2 Monate) gewesenem PH Sensor brachte dieser folgende Ergebnisse:



Wie zu sehen ist, weicht der gemessene Wert bereits stark von dem eigentlichen Wert ab. In einem Aquaponik System hat eine Abweichung dieser größe, starke Auswirkungen auf das Gleichgewicht des Ecosystems.

4.2.3.4 Fazit

Da es keine PH Sensoren gibt, die billig sind und zugleich einen niedrigen Warungsaufwand aufweisen, wird je nach Möglichkeiten versucht eine eigene Lösung zu etablieren. Dies soll mithilfe einiger Teststreifen und einer Kamera passieren, bei der, der Benutzer per Knopfdruck über die Website den PH Wert messen kann. Die Vorteile bei dieser Lösung sind, dass solch ein System einzigartig auf dem Markt wäre, und dass der Benutzer selbst bestimmen kann, ob er den PH Wert messen will und wie oft. Wie genau dieses System realisiert wird, wird im nächsten Sprint (Sprint 1) geplant.

4.3 Aktoren - User Story 1979

Es werden insgesamt vier Aktoren von Ponix Systems zur Verfügung gestellt. Darunter befinden sich: 2 Pumpen (Wasser/Mineralien), 1 Futterautomat und eine Pflanzenbeleuchtung.
Um dem Kunden die Arbeit abzunehmen die Zimmertemperatur so zu regulieren um ein Klima für die Fische zu schaffen, wird zusätzlich ein Hitzestrahler verbaut. Dieser wird im nächsten Sprint (Sprint 1) geplant und eingesetzt.

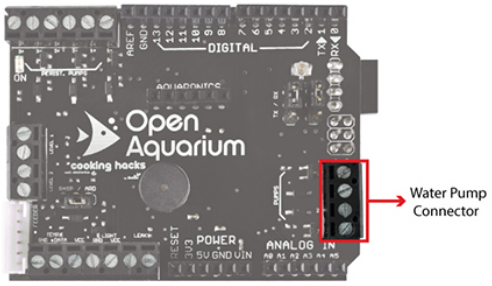
4.3.1 Pumpe

4.3.1.1 Wasser

Die Wasserpumpe kann mit einer Zeile Code ein bzw. ausgeschalten werden. Die Steuerung, die kontrolliert wann genau die Pumpe eingeschalten wird, muss erst implementiert werden (Sprint 1).

```
OpenAquarium.pumpON(1);
OpenAquarium.pumpOFF(1);
```

Zum Anschließen der Pumpe sind auf dem AquariumBoard Steckpltze vorgesehen.



Pumpleistung: 100-350

l/h
Spannung: 3.5-12V DC
Leistungsbereich: 0.5W-5W

4.3.1.2 Mineralien

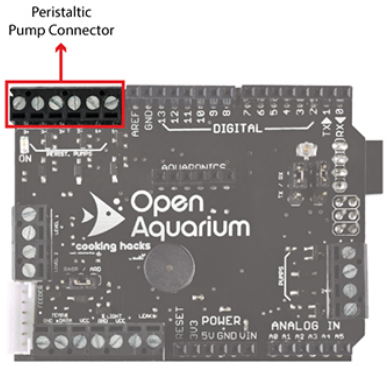
Bei dieser Pumpe handelt es sich um eine Schlauchpumpe, was den Vorteil bringt genau Dosierungen mit der Anzahl an Tropfen festzulegen. Wie bei der Wasserpumpe ist der Code zum Aktivieren und Deaktivieren der Pumpe kurz.

```
OpenAquarium.perpumpON(1);
OpenAquarium.perpumpOFF(1);
```

Die Dosierung der Mineralien muss nicht manuell vorgenommen werden sondern kann ber das OpenAquarium.h File angepasst werden.

```
#define dosingdrop1 5 // 5 Tropfen
#define dosingdrop1 10 // 10 Tropfen
#define dosingdrop1 15 // ...
```

Zum Anschlieen der Pumpe sind wieder Steckpltze vorhanden.



12V DC input

voltage
Flow: 20-60 ml/min

4.3.2 Futterautomat

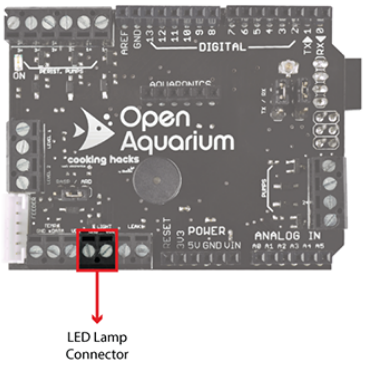
Um den Wartungsaufwand zu minimieren, ist ein Futterautomat notwendig, der über eine Weboberfläche konfiguriert werden kann. Es kann entweder ein teurer Futterautomat gekauft werden, der diese Option anbietet, oder ein durchschnittlicher Futterautomat wird so bearbeitet, dass er auf ein Stromsignal reagiert indem er Futter ausgibt.
Der Futterautomat von Ponix Systems gehört zur letzteren Art. Sobald 5 Volt angelegt wird, gibt er Futter aus. So kann später über die WebApp das Intervall und die Menge gesteuert werden.

4.3.3 Pflanzenbeleuchtung

Die Beleuchtung wird mit einem LED Streifen von www.cooking-hacks.com (bzw. von Ponix Systems gesponsort) realisiert. Er besteht aus 72 LEDs (60 Weie und 12 Blaue) mit einer Leuchtkraft von 500 Lumen. Der Streifen kann mit Hilfe der OpenAquarium Klasse der aktuellen Zeit automatisch angepasst werden. Der folgende Code, steuert die Lampe so, dass sie fr die Pflanze eine Sonne simuliert.

```
now = OpenAquarium.getTime();
OpenAquarium.printTime(now);
OpenAquarium.lighting(now);
```

Wie bei den beiden Pumpen kann der LED Streifen am OpenAquarium Board angeschlossen werden.



Leistung: 5.5 watts

Beleuchtung: 72 LEDs
Farbe: 60 Weie + 12 Blaue

4.4 Datenbankmanagementsystem

4.4.1 Evaluierung RDBMS vs. NoSQL

4.4.1.1 Einleitung

Zur Speicherung der Sensordaten soll ein geeignetes Datenbankmanagementsystem verwendet werden. Zu Beginn wird evaluiert, ob ein relationales oder ein NoSQL-Datenbanksystem verwendet wird. Dabei werden die jeweiligen Vor-und Nachteile von SQL und NoSQL Datenbanken aufgelistet. Besonders wert gelegt wird auf die Performance, die Konsistenz sowie dem einfachen Zugriff auf die Daten.

4.4.1.2 Kriterien

- Performance:
  Gibt an, wie schnell die Lese- und Schreibleistung ist.
- Konsistenz:
  Gibt an, wie gut das DBMS Maßnahmen zur Konsistenzerhaltung umsetzt.
- einfacher Zugriff auf Daten:
  Gibt an, wie Umfangreich die API des DBMS ist.

4.4.1.3 Optionen

4.4.1.4 SQl-Datenbanken

Abfragesprache

Die Abfragesprache SQL wir auf die drei Teile DDL(Data Definition Language), DML(Data Manipulation Language) und DCL (Data Control Language) aufgeteilt.

- DDL:
  ist die Sprache für das Anlegen, Ändern und Löschen von Datenstrukturen.
- DML:
  ist die Sprache für das Abfragen, Einfügen, Ändern oder Löschen von Nutzdaten.
- DCL:
  ist die Sprache für die Zugriffskontrolle.

Konsistenz

SQL-Datenbanken verwenden das ACID Prinzip, daher besitzen sie eine bessere Konsistenz als NoSQL-Datenbanken. NoSQL Datenbanken verwenden das BASE Prinzip, die Daten hier sind daher nur "schlussendlich" konsistent. Dadurch kann es passieren das nach einem Update nicht immer die selben Werte zurückgeliefert werden.

4.4.1.5 NoSQl-Datenbanken

NoSQL-Datenbanken besitzen wegen ihres einfachen Schemas eine hohe Skalierbarkeit, welche trotz eines hohen Datenvolumens gewährt wird.

4.4.1.6 Fazit

SQL-Datenbanken weisen eine bessere Konsistenz auf, jedoch besitzen NoSQL-Datenbanken eine bessere Performance und Skalierbarkeit. Da wir besonderen Wert auf die Performance legen, haben wir uns entschieden eine NoSQL-Datenbank zu verwenden.

4.4.2 Evaluierung eines NoSQL Datenbanksystems

4.4.2.1 Einleitung

Die Folgenden Datebankmanagementsysteme wurden evaluiert:

- MongoDB
- Redis

4.4.2.2 Kriterien

- Performance:
  Gibt an, wie schnell die Lese- und Schreibleistung ist.
- einfacher Zugriff auf Daten:
  Gibt an, wie Umfangreich die API des DBMS ist.
- Dokumentation:
  Gibt an, wie gut das DBMS dokumentiert ist.
- Verbreitung:
  Gibt an, wie weit das DBMS verbreitet ist.

4.4.2.3 Optionen

4.4.2.4 MongoDB

MongoDB ist eine Schema-freie, dokumentenorientierte NoSQL-Datenbank, die in der Programmiersprache C++ geschrieben ist. Da die Datenbank dokumentenorientiert ist, kann sie Sammlungen von JSON-ähnlichen Dokumenten verwalten. So können viele Anwendungen Daten auf natürlichere Weise modellieren, da die Daten zwar in komplexen Hierarchien verschachtelt werden können, dabei aber immer abfragbar und indizierbar bleiben.

Außerdem ist MongoDB eine Allzweckdatenbank mit offenem Quellcode (OpenSource). Sie hat u.a. folgende Merkmale:

- Dokumentdatenmodell mit dynamischen Schemata
- Umfassende, flexible Indexunterstützung
- Aggregation-Framework und MapReduce
- Auto-Sharding für horizontale Skalierbarkeit
- Eingebaute Replikation für Hochverfügbarkeit
- Textsuche
- Erweiterte Sicherheit (z.B. Kerberos Unterstützung)
- Speicherung großer Dateien mit GridFS

4.4.2.5 Redis

Redis ist eine In-Memory-Datenbank mit einer einfachen Schlüssel-Werte-Datenstruktur. Nach einer Erhebung von DB-Engines.com ist Redis der verbreitetste Schlüssel-Werte-Speicher.
Die einfache Struktur der Datenbank eignet sich weniger für komplexe Datenstrukturen, die überwiegend in der Datenbank selbst abgebildet werden soll, dafür ist der große Vorteil von Redis, dass es schneller ist als relationale Datenbanken wie z.B. MySQL. Bis zu ca. 100.000 Schreibvorgänge und ca. 80.000 Lesevorgänge pro Sekunde sind auf herkömmlicher Hardware möglich.
Redis bietet Persistenz durch automatisiertes regelmäßiges Abspeichern oder per Protokolldatei, wodurch bei entsprechender Konfiguration auch eine ACID-konforme Dauerhaftigkeit erreichbar ist.

4.4.2.6 Fazit

Redis schneidet bei den Kriterien, wie z.B. der Performance und den einfachen Zugriff auf Daten durch seine Kompaktheit, besser als MongoDB ab, hat jedoch den großen Nachteil, dass es bei serverseitiger Verwendung, extrem viel Arbeitsspeicher verbraucht, wenn einige Aquaponik Systeme ihre Daten über den Server persistieren.

4.4.3 Gesamtfazit

Es müssen sowohl auf dem lokalen Raspberry Pi, als auch auf dem Server Echtzeitdaten vorhanden sein, um einerseits auf dem Touchdisplay eine Übersicht anzuzeigen und andererseits über den Browser alle Daten zu betrachten. Da auf dem Raspberry Pi allerdings nicht viel Speicherplatz aber genügend Arbeitsspeicher vorhanden ist, wird dafür Redis verwendet und für die serverseitige Persistierung MongoDB.

4.5 Web Framework

Bei Web Frameworks, werden durch vordefinierte und vorgefertigte Klassen, häufig gebrauchte Funktionen wie Mailversand, sichere Authentifizierung, Sicherheitsfunktionen, Lokalisierung, Performance (z.B. HTTP Caching) oder grundlegende Funktionen für Webformulare vereinfacht.
Web Frameworks sind darauf ausgelegt, sehr schnell Ergebnisse zu erzielen und lauffähige Webanwendungen zu erstellen. Dazu bieten sie einen Datenbankzugriff, Templating-Mechanismen, eine saubere Trennung von Präsen-tation und Code durch Verwendung des Model-View-Controllers.
Da eine eigenständige Implementierung all dieser Mechanismen den Arbeitsaufwand übersteigen würde, ist eine Verwendung eines solchen Web Frameworks unabdinglich. Welches Framework dabei gewählt wird hängt von folgenden Komponenten ab:

- Schwierigkeitsgrad
- Lernkurve
- Sicherheit
- Dokumentation

4.5.1 Django

Django ist ein in Python geschriebenes opensource Web Application Framework, das einem Model-View-Presenter-Schema folgt.

4.5.1.1 Schwierigkeitsgrad/Lernkurve

Da Python selbst leicht zu erlernen ist, ist es keine Herausforderung sich in die Sprache einzuarbeiten. Das Framework ansich, gilt als eines der einfachereren (aber auch funktionsarmeren) seiner Art und bietet daher eine flache Lernkurve. Für jemanden der bereits mit Webframeworks gearbeitet hat, ist Django schnell zu erlernen.

4.5.1.2 Sicherheit

Django ist so konzipiert, dass die standartmäßig implementierten Sicherheitsmaßnahmen, für den größten Teil der Webentwicklung ausreicht. Da zwischen dem Raspberry Pi und dem Server wenig sicherheitsrelevante Daten ausgetauscht werden, genügt die Sicherheit von Django.

4.5.1.3 Dokumentation

Die Dokumentation von Django ist gerade für Einsteiger optimal, da sie einige nützliche Tutorials bietet. Aber auch für erfahrene Entwickler ist die Dokumentation großartig.

4.5.2 Node.js

Node.js ist eine Event-basierte Plattform, welche nicht blockierend ist. Das bedeutet, dass der Server nicht wartet bis ein Ereignis vollendet wurde, um mit den nächsten Aufgaben fortfahren zu können. Stattdessen werden Callbacks verwendet, welche dann erst aufgerufen werden, wenn ein bestimmtes Ereignis fertig ausgeführt wurde, bei dem von einer längeren Dauer ausgegangen wird. Währenddessen kann der Server andere Aufgaben bearbeiten.

An sich läuft Node.js nicht Multi-Threaded, wie man vielleicht erwartet, sondern bearbeitet jede eingehende Verbindung mit einem Thread. Jedoch werden bestimmte Funktionen, wie das Auslesen einer Datenbank oder das Rendern von Templates, intern durch mehrere Threads erledigt, welche dann bei Vollendung den zugewiesenen Callback aufrufen. Durch diese Architektur ist Node.js in vielen Bereichen viel schneller als andere Webserver, die für jede Verbindung einen Prozess erstellen würden.

Node.js ist sehr abhängig durch Third-Party Module, welche durch NPM (Node Package Manager) verwaltet und installiert werden können. Diese Module machen das Entwickeln von Applikationen viel einfacher oder geben auch mehr Funktionen. Für uns interessante Module wären zum Beispiel: Express.js oder Socket.io. Ersteres gibt uns die Möglichkeit die Applikation im MVC-Pattern zu entwickeln. Mit Socket.io könnte man Daten dem Client senden und gleichzeitig empfangen, ohne dabei davor eine Anfrage vom Client erwarten zu müssen. Es genügt damit nur noch eine Anfrage vom Client bei dem eine Verbindung (Socket) zum Server geöffnet wird, sodass man Daten bidirektional senden kann.

4.5.2.1 Schwierigkeitsgrad/Lernkurve

Wie der Name schon verrät verwendet Node.js JavaScript als Programmiersprache. Falls man die Sprache schon kann, wie in unserem Fall, ist eine Lernkurve jeweils in den einzeln verwendeten Modulen sichtbar, die sehr unterschiedlich sind. Somit ist der Schwierigkeitsgrad nicht wirklich einschätzbar. Der Schwierigkeitsgrad in den oben genannten Modulen liegt im mittleren Bereich.

4.5.2.2 Sicherheit

Node.js hat eine relativ große Community. Das ist insofern wichtig, da Sicherheitslücken dadurch schneller gefunden und behoben werden können. Jedoch unterscheidet sich die Sicherheit auch in diesem Punkt in den einzelnen Modulen. Somit muss beachtet werden, dass der zu verwendende Modul eine hohe Bekanntheit hat.

4.5.2.3 Dokumentation

Node.js ist an sich sehr gut dokumentiert und bietet durch eine Vielzahl von Tutorials einen einfachen Einstieg. Von den bekannten Modulen, wissen wir, dass die Dokumentation sehr gut ist, wie bei Socket.io oder Express.js. Außerdem wird der Einstieg in

diesen Modulen mit verschiedenen Tutorials, die im Internet zu finden sind, sehr einfach gemacht.

4.5.3 Laravel

Laravel ist ein PHP-Framework und wird von vielen Entwickler derzeit als eines der besten Frameworks für PHP bezeichnet. Laravel bietet viele Funktionen an, wie Routing, Templating, Unit Testing, usw.

Die Architektur von PHP unterscheidet sich von Node.js sehr stark:

- Pro Verbindung wird ein neuer Prozess erstellt. Dies hat bei hoher Nutzzahl zur Folge, dass der Arbeitsspeicher des Servers sehr schnell voll werden kann. Node.js verwendet nur einen Thread und erstellt, falls erforderlich, mehrere Threads für eine jeweilige Aufgabe, welche dann bei erfolgreicher Ausführung (also Aufrufen des Callbacks) beendet werden. Wird aber ein Fehler hervorgerufen stürzt Node.js komplett ab, wogegen bei PHP nur der Prozess abstürzt.
- Jedesmal, wenn ein PHP-Skript ausgeführt wird, wird das Skript neu interpretiert. Dies hat den Vorteil, dass bei einer Codeänderung nicht der Server neugestartet werden muss. Der Nachteil liegt jedoch in der Performance. Bei Node.js wird der Code beim Starten kompiliert. Eine Codeänderung wirkt sich jedoch solange nicht am Server aus bis Node.js neugestartet wurde. Dies dauert deutlich länger als ein PHP-Server. Dagegen ist die Performance in der Laufzeit deutlich schneller als bei PHP.
- Der Code in PHP läuft, anders als in Node.js, synchron ab. So muss zum Beispiel für eine Datenbankverbindung solange gewartet werden, bis diese auch erfolgreich erstellt worden ist. Dies würde die Antwortzeit einer Anfrage sehr beeinträchtigen.

4.5.3.1 Schwierigkeitsgrad/Lernkurve

Die Skriptsprache PHP ist sehr einfach zu lernen, genauso wie das Framework Laravel. Jedoch haben die Teammitglieder nicht viel Erfahrung mit PHP, sodass etwas Zeit für das Lernen vergehen würde.

4.5.3.2 Sicherheit

Die Standart implementationen und Sicherheitsmanahmen sind fr den grten Teil der Webentwicklung ausreichend und decken die Ansprche des Projekts vollstndig.

4.5.3.3 Dokumentation

PHP ansich bietet eine hervorragende Dokumentation mit einigen Beispielen, was den Einstieg in die Sprache erleichtert. In die Dokumentation von Laravel muss man allerdings viel Zeit investieren um sich einzulesen.

4.5.4 Fazit

4.5.4.1 Django

Für Einsteiger ist Django definitiv empfehlenswert. Die gute Dokumentation ermöglicht es jeden Entwickler mit - egal mit welchen Kenntnissen - sich einen guten Überblick, was für dieses Projekt bereits ausreichend ist.

4.5.4.2 Node.js

Node.js kann mit seinen Modulen innovative Lösungen bieten, wie das schon erwähnte Socket.io-Modul. Außerdem besitzt Node.js eine sehr große Community, die ständig wächst, wie auch der derzeitige Trend zeigt. Große Unternehmen, wie Netflix oder Paypal, basieren auf Node.js, aufgrund seiner Schnelligkeit und hohen Skalierbarkeit.

4.5.4.3 Laravel

PHP bzw. Laravel ist für unsere Anforderungen nicht besonders gut geeignet, da unser Produkt ein schnelles System erfordert, sowie eine hohe Zahl von Verbindungen. Dies würden wir mit PHP nicht erreichen können.

Bewertung der Frameworks nach Schulnotensystem (1 - Sehr gut bis Nicht genügend - 5)

Framework	**Schwierigkeit**	**Sicherheit**	**Dokumentation**	**Erfahrung**
Django	2	3	2	2
NodeJS	1	2	1	1
Laravel	3	2	2	3

4.6 Art der Datenübermittlung

Bei der Datenübertragung sind mehrere Parameter zu beachten. Dazu gehören hauptsächlich das Format der Daten und das Intervall in denen diese übertragen werden.

4.6.1 Arduino zu Raspberry Pi und vice versa

Um die Sensordaten dem Raspberry Pi mitzuteilen, eignet sich am besten ein Key-Value Format. Dabei gibt es verschiedene Möglichkeiten, wie z.B. JSON und CSV.
Das Intervall der Datenübertragung kann frei, vom Benutzer gewählt werden.

4.6.1.1 JSON

JSON (JavaScript Object Notation) hat den großen Vorteil, der leichten Server-seitigen Auswertung und Konvertierung in ein Javascript Object. Da die Daten aber vorübergehend auf dem Raspberry Pi persistiert werden sollen, und Redis kein automatisches speichern von JSON Objekten unterstützt erzeugt diese Methode der Übertragung mehr Aufwand als Nutzen.

4.6.1.2 CSV

Ähnlich wie bei JSON, müssten die Daten erst bearbeitet und gefiltert werden, um in die Datenbank gespeichert werden zu können.

4.6.1.3 Eigenes Format

Um die Daten mit Redis zu persistieren wird folgende Syntax verwendet: "Key1 Value1 Key2 Value2 ...". Um Mehraufwand durch Verwendung eines standartisierten Datenformats zu vermeiden, können die Daten gleich so gesendet werden, dass sie ohne Bearbeitung in Redis gespeichert werden können.

4.6.1.4 Fazit

Der Datenaustausch wird in folgendem Format stattfinden:

```
temperature 23.4
ec 285.7
```

4.6.2 Raspberry Pi zu Server und vice versa

Wenn die Daten in Redis gespeichert worden sind, müssen diese dem Zentralserver gesendet werden. Der Sender auf dem Raspberry Pi ist Node.js, wie auf dem Empfänger bzw. Server.

Die Daten schauen bei Auslesen der Datenbank wie folgt aus:

```
1) temperature
2) 23.4
3) ec
4) 285.7
```

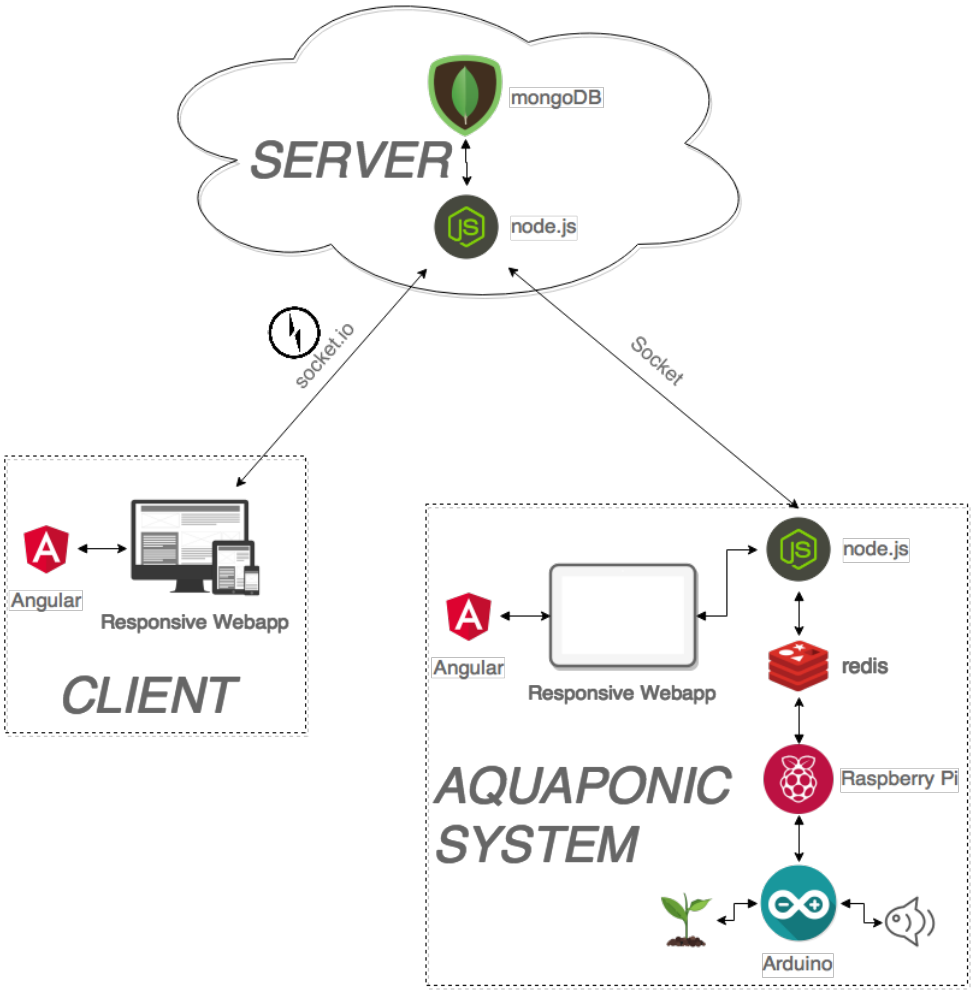
Wie wir in Kapitel 10.1 festgestellt haben, offenbart sich die Übertragung in JSON nicht nur wegen der Einfachheit der Konvertierung zu einem Javascript Objekt als Vorteil, sondern auch aus drei anderen Gründen:

- **Einfache Speicherung in MongoDB:** Da MongoDB JSON als Speicherformat verwendet, ist es deswegen für uns einfacher JSON-Objekte in die Datenbank zu speichern.
- **Übertragung der Daten durch Socket.io:** Daten werden standardmäßig in Socket.io als JSON-Objekt gesendet.
- **Oberfläche der Webapplikation durch Angular:** Angular bietet Funktionen an, JSON-Objekte in der Weboberfläche anzeigen zu lassen. Falls sich das JSON-Objekt verändert, wird automatisch auch die Oberfläche dementsprechend angepasst.

Deswegen werden die Daten wie folgt ausschauen:

```
{
  temperature: "23.4",
  ec: "285.7"
}
```

4.7 Fazit des kompletten Systems



5 Projekt Umsetzung - Designunterlagen

5.1 Use-Case Diagramm

5.2 Datenakquisition

5.2.1 Technologie

5.2.2 Ablauf

5.3 Datenverarbeitung

5.3.1 Technologie

5.3.2 Umsetzung des Ablaufs

6 Zugriff auf das Aquaponik Systems

6.1 Userinterface - Frontend

6.1.1 User Experience

6.1.2 Browserkompatibilitt

6.1.3 Technologien

6.1.4 Funktionen

6.1.5 Optimierungen

6.2 Zentralserver

6.2.1 Authentifikation

6.2.2 Kommunikation

7 Ausfallsicherheit und Konsistenzsicherung der Daten

7.1 Zwischenspeicherung beim Aquaponik System

7.1.1 Ausfall der Internetverbindung

7.1.2 Ausfall der Stromversorgung

8 Prototyp testing

8.1 Hardware

8.2 Prototyp

8.3 Aussichten

9 Ausblick

10 Appendix

10.1 Glossaries

10.2 Figures

10.3 Listings

10.4 Sources

[BVH15] Scholz Dominik Belinic Vennesa, Haidn Martin and Siegel Hannah. Robonav. https://github.com/TGM-HIT/RoboterNavigation/, 2015.

[Git15] GitHub. Downloading git. https://www.git-scm.com/download/win, 2015.

[Pro14] The Apache Ant Project. Current release of ant. http://ant.apache.org/bindownload.cgi, 2014.

[Ver15] Dr. Christian Verbeek. Downloads. http://wiki.openrobotino.org/index.php?title=Downloads#API2, 2015.