

# **Sensorbasierte Pflanzenüberwachung in Echtzeit**

## **Eine IoT-Lösung für eine intelligente Vase mit Sprachsteuerung**

**Studienarbeit**

des Studienganges Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

**Tobias Kapp**

Juni 2024

**Bearbeitungszeitraum**

2 Semester

**Matrikelnummer, Kurs**

1985124, TINF21B

**Dualer Partner**

diconium digital solutions GmbH

**Betreuer**

Prof. Dr. Monika Kochanowski

# **Abstract**

Smart Devices dringen auch im privaten Bereich immer weiter vor und sollen verschiedenste Tätigkeiten automatisieren und vereinfachen.

In der landwirtschaftlichen Industrie werden bereits verschiedenste Arten von intelligenten Systemen benutzt, um den Pflanzenanbau zu automatisieren und effizienter zu gestalten. Ähnliche Systeme können auch im privaten Bereich genutzt werden, um die Pflanzenpflege zu vereinfachen.

In der folgenden Arbeit wird ein erster sprachgesteuerter Prototyp erstellt, welcher dynamisch die Kultivierung von Pflanzen unterstützt und automatisch die Pflanze bewässert. Dieses System soll für weitere AI-Anbindungen offen gehalten werden, um sich selbstständig verbessern zu können.

## **Erklärung**

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Aus den benutzten Quellen, direkt oder indirekt, übernommene Gedanken habe ich als solche kenntlich gemacht. Diese Arbeit wurde bisher in gleicher oder ähnlicher Form oder auszugsweise noch keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Stuttgart, Juni 2024

---

Tobias Kapp

# Inhaltsverzeichnis

<b>Abstract</b> . . . . .	<b>I</b>
<b>Erklärung</b> . . . . .	<b>II</b>
<b>Inhaltsverzeichnis</b> . . . . .	<b>III</b>
<b>Glossar</b> . . . . .	<b>VI</b>
<b>1 Einleitung</b> . . . . .	<b>1</b>
1.1 Abgrenzung zu anderen Projekten und Zielstellung . . . . .	1
1.2 Aufbau der Arbeit . . . . .	2
<b>2 Stand der Technik</b> . . . . .	<b>2</b>
2.1 Existierende Produkte für private Endverbraucher . . . . .	3
2.2 Agrikulturelle Forschung . . . . .	6
2.3 Theoretische Grundlage des Machine LearningMachine Learning Verfahrens	11
2.3.1 . . . . .	13
2.3.2 Artificial Neural Networks . . . . .	14
2.3.3 Pattern Recognition . . . . .	15
2.3.4 Zusammenfassung . . . . .	16
2.4 Bilderkennung . . . . .	16
<b>3 Methodik</b> . . . . .	<b>17</b>
3.1 Anforderungsanalyse . . . . .	18
3.1.1 Funktionale Anforderungen . . . . .	18
3.1.2 Nicht-funktionale Anforderungen . . . . .	19
3.2 Grobkonzept . . . . .	20
3.3 Feinkonzept . . . . .	22
3.3.1 Auswahl des Mikrocontrollers . . . . .	22
3.3.2 Auswahl der Sensoren . . . . .	23
3.3.3 Nachrichtenprotokoll . . . . .	26
3.3.4 Programmiersprache . . . . .	27
3.3.5 Auswahl der Datenbank . . . . .	28
3.3.6 Sprachsteuerung . . . . .	29
3.3.7 Testing . . . . .	31
3.3.8 Deployment Verfahren . . . . .	32
3.3.9 Serverinfrastruktur . . . . .	33
3.4 Theoretisches Konzept . . . . .	33
<b>4 Implementierung</b> . . . . .	<b>36</b>

4.1	Projektarchitektur . . . . .	36
4.2	I/O der Hardware . . . . .	37
4.3	Mikrocontroller . . . . .	37
4.3.1	Sensormessung und Umwandlung der Daten . . . . .	39
4.3.2	Senden und Empfangen von Daten . . . . .	39
4.3.3	Einschalten der Pumpe . . . . .	39
4.3.4	Zusammenfassung . . . . .	39
4.4	Main Controller . . . . .	40
4.5	Nachrichten Controller . . . . .	41
4.6	Datenbankcontroller . . . . .	41
4.6.1	Schreiben der Daten . . . . .	41
4.6.2	Lesen der Daten . . . . .	41
4.7	Reaktionscontroller . . . . .	42
4.8	Sprachsteuerungs Controller . . . . .	43
4.9	Testing . . . . .	44
4.10	Deployment . . . . .	44
<b>5</b>	<b>Evaluation . . . . .</b>	<b>45</b>
5.1	Anforderungen . . . . .	45
5.2	Umgesetzte Funktionen . . . . .	47
5.3	Qualität des Codes . . . . .	47
5.4	Tauglichkeit des Mikrocontrollers und der Sensoren . . . . .	49
5.5	Skalierung . . . . .	49
<b>6</b>	<b>Fazit . . . . .</b>	<b>50</b>
6.1	Zusammenfassung . . . . .	50
6.2	Ausblick . . . . .	51
<b>Abbildungsverzeichnis</b>	<b>52</b>	
<b>Tabellenverzeichnis</b>	<b>53</b>	
<b>Quellcodeverzeichnis</b>	<b>54</b>	
<b>Anhang</b>	<b>55</b>	
<b>Literatur</b>	<b>78</b>	

V

# Glossar

**Entscheidungsbaum** Hierarchische Baumstruktur, um Entscheidungen zu fällen.

**Fuzzy Logic** Alternative zu Boolean Entscheidungen, indem Entscheidungsspannen festgelegt werden. Welche Temperatur kann als warm oder kalt anerkannt werden.

**Gamification** Eine Erfahrung mit verschiedenen Techniken von Spielen auszubauen, um Nutzuende zu motivieren.

**Greedy Algorithmus** Einfache Art von problemlösenden Algorithmen, sie suchen nicht die beste Lösung, sondern eine akzeptable in einem annehmbaren Zeitraum.

**Künstliche Neuronale Netzwerke** Machine Learning Algorithmen, welche sich auf die Funktionsweise des menschlichen Gehirns berufen.

**Machine Learning** Wissenschaft, welche sich mit der Erstellung von stochastischen Algorithmen befasst, welche ohne explizite Instruktionen Daten auswertet.

**Morphologische Analyse** Aus der Produktentwicklung stammendes System, bei dem mehrere Herangehensweisen benutzt werden, um alle Seiten eines Konzeptes vorurteilslos zu betrachten.

**Selective Search** Wenig aufwendige Methode zur Objekterkennung, basierend auf der Forschung von Felzenszwab und Huttenlocher.

**Trainingsset** Beispiele für ein Machine-Learning-Algorithmus, von dem er Regeln ableiten kann.

# 1 Einleitung

Intelligente Maschinen übernehmen schon lange die Pflanzenpflege für landwirtschaftliche Unternehmen. Dies soll Kosten senken, Ressourcen schonen, den Ablauf produktiver gestalten und die Qualität der Ernte erhöhen. Bereits jetzt nutzen über 20 Prozent aller landwirtschaftlichen Betriebe Farm-Management-Software, rund fünf Prozent setzen auf vollautomatisierte Technologien.[1] Die Technologien sind hierbei weitreichend: Roboter, automatisierte Gewächshäuser und Bewässerungsanlagen und Datenauswertungen tragen dazu bei, gesunde Pflanzen effizient zu kultivieren.

Derartige Ansätze sind auch im privaten Gebrauch sinnvoll. Eine Studie aus den USA schätzt den Marktwert der amerikanischen Hauspflanzindustrie auf 16 Milliarden Dollar ein[2]. Die Pflege von Hauspflanzen ist jedoch anspruchsvoll; in den Niederlanden sterben jedes Jahr 16 Millionen Hauspflanzen, hauptsächlich aufgrund mangelnden Wissens über die Bedürfnisse verschiedener Pflanzen.[3] Diese Werte sind repräsentativ für den Großteil der westlichen Länder. Automatisierte Pflanzenpflege könnte helfen, Pflanzen auch ohne spezielles Fachwissen gesund zu halten. Zudem kann sie Menschen Arbeit abnehmen, da die Pflege zeitaufwendig ist und aktive Beteiligung erfordert. Es ist notwendig, Wissen über Pflanzenpflege zu erwerben, die Pflanzen zu beobachten, ihre Reaktionen auf Umwelteinflüsse zu verstehen, Krankheiten zu identifizieren und regelmäßige Tätigkeiten wie Düngen und Gießen durchzuführen. Durch die teilweise Automatisierung dieser Aufgaben bleibt den Nutzenden mehr Zeit für andere Aktivitäten.

Automatisierung und Unterstützung im privaten Bereich könnten somit nicht nur den Aufwand für weniger engagierte Pflanzenbesitzende reduzieren, sondern auch die Verschwendungen gesunder Pflanzen verringern, die sonst neu gekauft werden müssten..

## 1.1 Abgrenzung zu anderen Projekten und Zielstellung

Dieses Projekt unterscheidet sich von anderen Produkten durch seine vollständige Sprachsteuerung und die Berücksichtigung verschiedener Nutzergruppen. Es soll sowohl für Enthusiasten relevante Informationen und Zusammenhänge zwischen Umweltparametern aufdecken als auch die Pflanzenpflege nahezu vollständig übernehmen. Es wird lediglich ein Sprachassistent benötigt, visuelle Eingaben sind nicht erforderlich. Das Gerät soll den Nutzenden Zeit ersparen und keine zusätzliche Aufmerksamkeit erfordern. Interessierten bietet es wertvolle Informationen, ohne sie in redundanten Informationsflüssen zu verstricken.

Durch den Einsatz verschiedener KI-Techniken soll die Maschine optimale Ergebnisse erzielen und die Pflege an die Bedürfnisse jeder speziellen Pflanze anpassen. Kleine Veränderungen der Umweltparameter werden genutzt, um das Wachstum der Pflanzen zu optimieren. Nutzende sollen nur bei Interesse in diesen Prozess eingreifen.

Um diesen Ansprüchen gerecht zu werden, reicht ein lokales System, das isoliert von der

Außenwelt agiert, nicht aus. Berechnungen und Erfahrungswerte von vielen verschiedenen Geräten sollen zusammengetragen und gespeichert werden. So können Optimalwerte für neue Pflanzengattungen erstellt und jederzeit angepasst werden. Werte wie die Wasserausgabe oder die Zeitabstände zwischen Berechnungen können flexibel verändert werden, um das System zu verbessern. Die gesammelten Informationen können auch zur Erstellung weiterer Projekte und Studien genutzt werden.

## 1.2 Aufbau der Arbeit

In diesem Projekt wird ein Prototyp für eine intelligente Vase erstellt.

Zunächst werden andere Produkte und Projekte gesichtet, um die Anforderungen zu definieren und aus bereits gemachten Erfahrungen zu lernen. Mithilfe eines Mikrocontrollers und Sensoren werden verschiedene Umweltparameter gesammelt und in einer zentralen Datenbank gespeichert. Aus diesen Daten werden Berechnungen durchgeführt, um sie mit optimalen Wachstumswerten zu vergleichen und sinnvolle Anweisungen zu generieren. An den Mikrocontroller wird eine Pumpe angeschlossen, die die Pflanze automatisch bewässert.

Anschließend wird das Projekt hinsichtlich der Anforderungen, Codequalität und Funktionsweise evaluiert.

Abschließend werden die Ergebnisse zusammengefasst und ein Ausblick auf das weitere Vorgehen gegeben.

## 2 Stand der Technik

Dieses Kapitel beleuchtet bereits existierende Produkte auf dem Markt und setzt die Grundlage für weiterführende Überlegungen. Es werden sowohl Vasen mit Sensoren für private Endverbrauchende untersucht als auch kommerziell eingesetzte Maschinen und akademische Forschungen erläutert. Ziel ist es, diese Erfahrungswerte auf das neue Produkt „Intelligente Vase“ übertragen. Zusätzlich werden verschiedene Machine-Learning-Verfahren aufgezeigt, um geeignete Werkzeuge für das Projekt zu identifizieren.

Um vorhandene Geräte besser zu kategorisieren, muss der Begriff „intelligent“ definiert werden. Viele Produkte verwenden den Begriff „intelligent“, wenn das System lediglich automatisch abläuft. Folgende Systeme können voneinander unterschieden werden:

Systeme der dritten Art erfassen Änderungen in der Umgebung, wie etwa die Stärke des Sonnenlichts pro Tag, und dokumentieren deren Auswirkungen auf die Pflanze. Mithilfe von Lampen und einer Pumpe können die Umweltbedingungen auch aktiv verändert werden, um die Präferenzen der Pflanze zu testen und optimale Wachstumsbedingungen zu erzielen.

Art des Systems	Beschreibung	Beispiel
Automatisch	Automatische Abläufe, ohne Rücksicht aus Umwelt	Ein System gießt alle 4 Stunden 20ml
Intelligent 1. Art	Reagieren auf unterschiedliche Umwelteinflüssen	Künstlichen Sonnenlicht wird nur aktiviert, wenn die Sonne nicht scheint
Intelligent 2. Art	Rücksicht auf Art der Pflanze	Kakteen benötigen weniger Wasser als Tulpen. Erkennung der Pflanze muss nicht automatisch ablaufen.
Intelligent 3. Art	Anpassung der Werte	Pflanzen sind unterschiedlich. Wachstum wird erkannt und Umwelteinflüsse dynamisch angepasst, um möglichst gute Ergebnisse zu erzielen.

Tabelle 1: Einteilung der Intelligenten Systeme

## 2.1 Existierende Produkte für private Endverbraucher

Produkte, welche die Pflanzenpflege für private Nutzende teilweise automatisieren, werden häufig unter den Bezeichnungen SSmart Garden und SSmart Vase angeboten. Es gibt jedoch keine einheitliche Definition, und Funktionen, Aussehen sowie Technologien variieren je nach Produkt und Hersteller.



Abbildung 1: Ein Hydroponisches Anbausystem

[4]

**Hydroponische Systeme** Viele Produkte konzentrieren sich auf die Anzucht. Diese kosten zwischen 90 Euro Emsa [5] und 450 Euro Geberioz [6]. Sie nutzen überwiegend ein hydroponisches System, bei dem Pflanzen ohne Erde, lediglich durch einen konstanten, langsamem Wasserzufluss mit einer Nährstofflösung, aufgezogen werden. Diese Technik ist sehr effizient und ermöglicht eine genaue Kontrolle des Pflanzenwachstums. Pflanzenfabrik [7]

Die Geräte sind mit einem Wassertank und einem Platz für produktspezifische Pads ausgestattet, die die Pflanzen mit den richtigen Nährstoffen versorgen. Über Schläuche wird ein konstanter Wasserzufluss gewährleistet. Zusätzlich sind mehrere LED-Lampen angebracht, um den jungen Pflanzen ausreichend Licht zu geben, wobei auf Sonnenlicht meist nicht eingegangen wird. Diese Geräte erfordern normalerweise keine externe Steuerung, da Wasserzufluss und Licht vom Hersteller vorgegeben sind. Unterschiede zwischen den Pflanzen werden dabei nicht berücksichtigt. Hochpreisige Alternativen bieten manchmal eine Steuerung über Smartphone-Apps oder Fernbedienungen an, wobei Pumpe und Licht separat kontrolliert werden können, jedoch keine vollständig automatisierte Anpassung stattfindet.

Diese Systeme agieren automatisch und nicht intelligent.



Abbildung 2: Eine Smart Vase mit Display

[8]

**Smart Vases** Wesentlich geringer ist das Angebot bei Smarten Vasen. Diese kombinieren verschiedene Sensoren mit dem Verhalten eines digitalen Haustiers. Bei Kosten von etwa 150€[9] besitzen sie ein Display, auf dem die verschiedenen 'Gemütszustände' der Pflanze angezeigt werden. Über Emojis wird beispielsweise angezeigt, ob die Pflanze gegossen werden muss oder es an Sonnenlicht fehlt. Andere Produkte integrieren weitere 'Haustierfunktionen', wie ein Verlangen nach Aufmerksamkeit oder haptischer Berührung. Pflanzenpflege wird auf diese Weise gamifiziert. Gamification und durch das Display wird Aminismus erzeugt, das Zuschreiben von Persönlichkeit an ein lebloses Objekt. Die aufgebaute Beziehung sorgt für ein erhöhtes Verantwortungsbewusstsein[10] und soll der

Pflanzenpflege mehr Bedeutung geben, sowie die Nutzenden motivieren.

Diese Maschinen besitzen in der Regel keine Aktoren; die Nutzenden müssen sich weiterhin selbst um die Pflanze kümmern. Zudem können sie nicht auf spezifische Pflanzen zugeschnitten werden, da die Richtwerte vom Hersteller vorgegeben und allgemein gehalten sind.

Diese Systeme gehören zu den intelligenten Systemen erster Art.

**Automatische Bewässerungssysteme** Im Freien existieren verschiedene Geräte zur Bewässerung von Pflanzen. Eine Sprenkleranlage wird automatisch durch eine kontinuierliche Wasserversorgung aktiviert, wodurch die Nutzenden keine manuelle Bedienung vornehmen müssen. Einige Geräte integrieren intelligente Funktionen. Durch einen WLAN-Adapter können Wettervorhersagen eingebunden werden, um das Bewässern bei Regen zu vermeiden oder durch Bodenfeuchtesensoren übermäßige Bewässerung zu verhindern und besonders sonnenintensive Tage auszugleichen. Zudem können mittels verschiedener Programme, häufig über Smartphone-Apps zugänglich, individuelle Bewässerungszeiten festgelegt werden, beispielsweise ausschließlich nachts statt tagsüber. Intelligente Varianten dieser Geräte tragen dazu bei, den Wasserverbrauch zu reduzieren. Die Preisspanne für derartige Produkte liegt zwischen ca. 70€ und 200€.[11]

Obwohl diese Maschinen primär für größere Gärten konzipiert sind, lässt sich ihr Ansatz auch auf die Intelligente Vase anwenden. Der Hersteller Rainpoint veranschaulicht die Funktionsweise dieses Geräts. Der Controller verbindet sich dabei mit der Cloud, auf die die Nutzenden über ihr Smartphone zugreifen können. Diese Geräte können entweder der Automatischen oder der Intelligenten Kategorie der 1. Art zugeordnet werden.

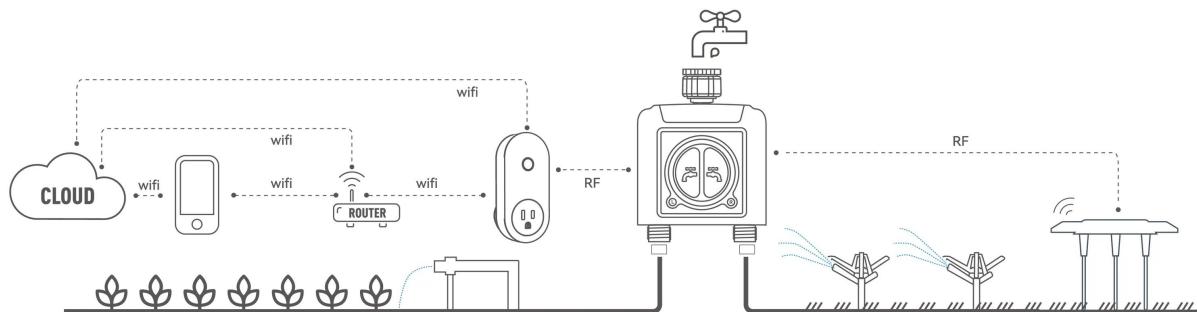


Abbildung 3: Diagramm des Herstellers Rainpoint

[11]

Diese Geräte können zur automatischen oder zur intelligenten Kategorie der 1. Art zählen.

## 2.2 Agrikulturelle Forschung

”Die Landwirtschaft ist bis heute die wichtigste Erwerbsquelle und der größte Wirtschaftszweig der Welt. Ein Drittel aller arbeitenden Menschen ist in der Landwirtschaft beschäftigt.”[12]

Mit zunehmendem Bevölkerungswachstum und gesteigertem Konsum trägt die Landwirtschaft vier Prozent zum gesamten weltweiten Bruttoinlandsprodukt bei. Die Möglichkeit, durch Automatisierung Arbeitskräfte einzusparen, die Natur zu schonen und nachhaltig zu nutzen sowie profitablere Ernten zu erreichen, hätte einen erheblichen globalen Einfluss. Die Europäische Kommission betrachtet Innovation und Fortschritt als eine ihrer wichtigsten Aufgaben in den kommenden Jahren .[13]

Ein großer Fokus der agrarischen Automatisierung liegt auf der Bewässerung. Mit Hilfe verschiedener Algorithmen und KI-Modelle wird versucht, Wasser möglichst effizient einzusetzen und Verschwendungen zu vermeiden. Die Grundlage für eine automatische Bewässerung besteht aus drei Schritten:[14]

- Aufnahme der Bodenfeuchtigkeit mittels Sensoren
- Vergleich mit internen Datenbanken bezüglich des optimalen Feuchtigkeitswerts
- Reaktion durch Ein- oder Ausschalten der Pumpe

Wahlweise können Daten auch für weitere Forschungszwecke gespeichert werden. In größeren Anbaugebieten, in denen nicht jeder Quadratmeter vermessen werden kann und verschiedene Pflanzen angebaut werden sowie komplexere Pumpensysteme eingesetzt werden, können auch komplexere Systeme wie Fuzzy-Controller verwendet werden, um die gesamte Fläche effektiv zu bewässern .[15]

Gewächshäuser ermöglichen den Anbau kälteempfindlicher Pflanzen auch in kalten Regionen. In den meisten Konzepten kann erwärmede Luft von Sonnenstrahlen nicht nach außen entweichen, was die Temperatur innerhalb des Gewächshauses erhöht. Um optimales Pflanzenwachstum zu gewährleisten, darf die Temperatur jedoch nicht zu hoch sein. Verdunstungskälte, Ventilatoren oder Raumöffnungen können hier Abhilfe schaffen. Auch hier werden seit Langem automatisierte Verfahren verwendet. Das Grundprinzip ähnelt dem der Bewässerung:[16]

- Aufnahme der Lufttemperatur mittels Sensoren
- Vergleich mit internen Datenbanken bezüglich des optimalen Temperaturwerts
- Reaktion durch Ein- oder Ausschalten der Kühlungsmethoden

Auch hier könnten sich Fuzzy-Controller eignen, falls mehrere Möglichkeiten zur Kühlung existieren. Oft wird die Temperaturregelung in Gewächshäusern mit automatischer Bewässerung kombiniert.[17]

Inzwischen werden auch komplexere Machine Learning Techniken verwendet, um Pflanzen optimal zu überwachen.

**Integration von Machine Learning Systemen in der Champignon Zucht[18]** In einer Champignon-Zucht in Litauen wurde über die letzten Jahre ein Klimamanagementsystem entwickelt. Dabei sammelten Umweltsensoren Daten wie die Komposttemperatur, Luftfeuchtigkeit, CO<sub>2</sub>-Gehalt in der Luft und weitere Informationen während der verschiedenen Wachstumsphasen der Pilze. Diese Informationen wurden über eine Benutzeroberfläche ausgegeben und konnten verschiedene Werte anhand vorgegebener Richtlinien anpassen (intelligentes System zweiter Art). Forschende kamen nun auf die Idee, ein System zu integrieren, das anhand des aktuellen Zustands der Pilze Empfehlungen für Veränderungen gibt. Das Modell verknüpft hierbei Umweltdaten mit visuellen Informationen. Verschiedene Kameras nehmen in regelmäßigen zehnminütigen Abständen Fotos vom Zustand der Zucht auf. Erkennt der Computer Verbesserungsmöglichkeiten, werden diese dem Team mitgeteilt.

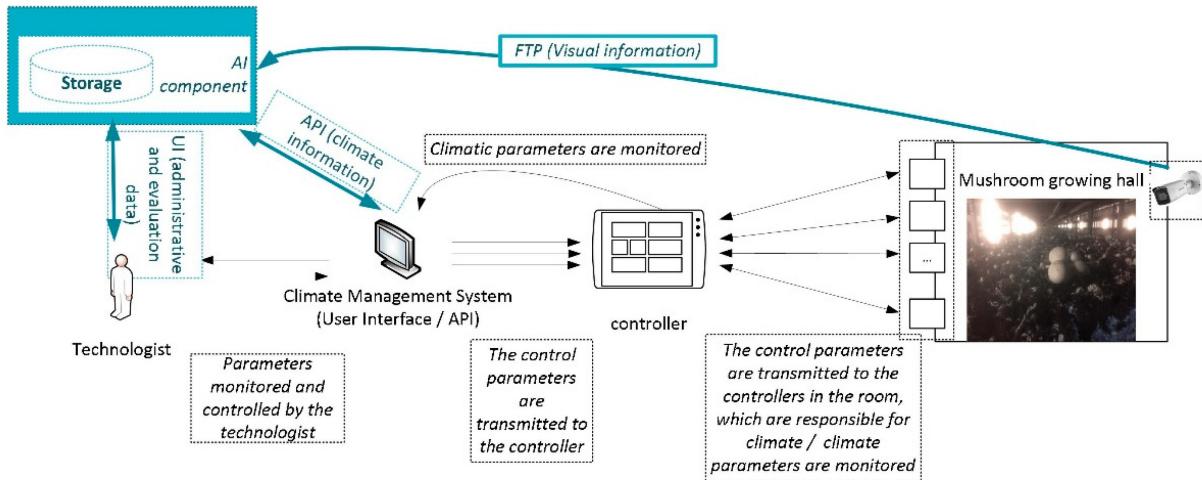


Abbildung 4: Diagramm der AI-Einbindung in der Champignon Aufzucht  
[11][18]

Das Projekt bietet Einblicke, wie verschiedene KI-Tools in die Pflanzenaufzucht integriert werden können. Es werden auch die genauen Technologien beschrieben. So wurden aufgenommene Bilder nicht nur auf die Größe der Pilze untersucht, sondern auch die des Myzels, der fädenförmigen Zellen eines Pilzes. Zur genaueren Analyse wurde die Fourier-Analyse genutzt. Dabei werden komplexe Signale in einzelne Sinus- und Kosinuswellen aufgespalten und unerwünschtes Rauschen entfernt, um Muster zu erkennen.[19] Hier wurden drei verschiedene schwarz-weiß Bilder (a, e, i) verwendet, die innerhalb von 8 Tagen aufgenommen wurden. Diese Bilder wurden durch drei verschiedene Filter analysiert, die jeweils tiefe (b, f, j), mittlere (c, g, k) und hohe Frequenzen (d, h, l) zulassen. Tiefe Frequenzen stehen hierbei für graduelle Unterschiede in der Farbe, beispielsweise den

verschiedenen Weißtönen des Pilzes, während hohe für starke Kontraste stehen, wie etwa die Kante zwischen Pilz und Boden.

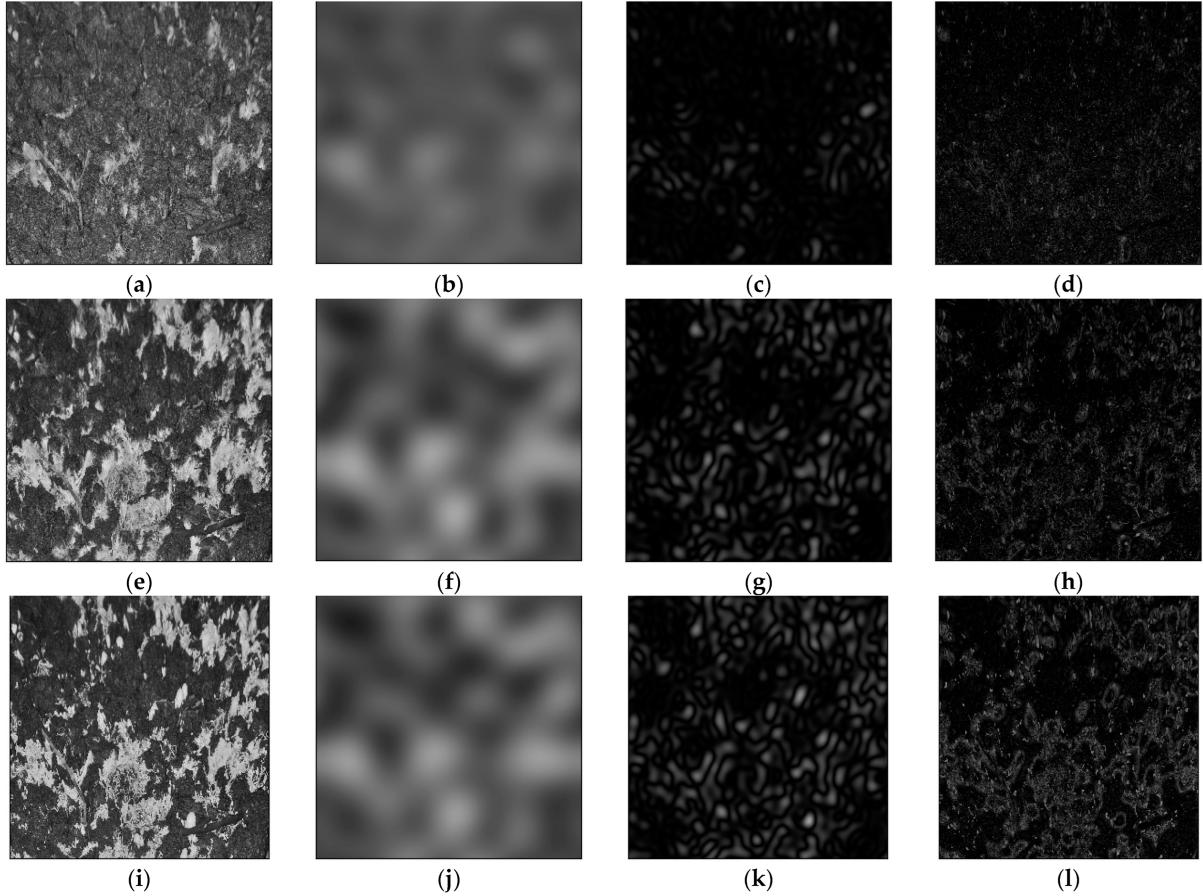


Abbildung 5: Bildverarbeitung anhand der Fourier Analyse  
[11][18]

Mithilfe der Morphologischen Analyse kann erkannt werden, wie viele einzelne Myzel-Flächen existieren und wie groß diese sind. Um die einzelnen Pilze zu erkennen, wurde das Faster R-CNN (Region-based Convolutional Neural Network) Verfahren verwendet. Regions of Interest (ROI), für den Computer relevante Abschnitte, werden hierbei erkannt, mithilfe von Selective Search und einem Greedy Algorithmus zu maximal 2000 Objekten zusammengefasst und mit Rechtecken umrandet.[20] Anschließend werden die ROIs durch ein Neuronales Netzwerk analysiert, um den Objekttyp zu bestimmen. Beim Faster R-CNN wird die Auswahl der Regions of Interest in das Neuronale Netzwerk integriert, was den Vorgang deutlich beschleunigt. Mithilfe von realen Markern, die in den Boden gesetzt wurden, konnten die Maße der Objekte in reale Daten umgesetzt und in verschiedene Kategorien gefasst werden.

Für eine sorgfältige Bildanalyse sollten also in jedem Fall verschiedene Bilder über mehrere Tage hinweg verwendet werden. Diese werden unterschiedlich aufbereitet, um

möglichst vielfältige Informationen zu bieten, damit das System das gewünschte System möglichst genau erkennen kann.

Die über mehrere Wochen gesammelten Daten konnten benutzt werden, um das Wachstum und die Gesundheit der Pilze unter verschiedenen Umweltbedingungen in den verschiedenen Wachstumsphasen einzuschätzen. Mit diesen Daten konnten Algorithmen entwickelt werden, um Empfehlungen für das Team bezüglich des Pflanzenwachstums zu geben.

Die Entscheidungsfindung basiert auf einem Entscheidungsbaum. Es werden sowohl Expertenmeinungen, die visuellen Daten als auch die Klimaparameter berücksichtigt. Die Merkmale werden in 4, 12 und 24 Stunden Intervallen betrachtet. Die Werte werden mit den am ähnlichsten vergangenen Versuchen verglichen. Daraus wird abgeleitet, ob eine Änderung der Parameter erforderlich ist. Ist dies der Fall, werden die Werte mit der besten Wachstumsprognose empfohlen. Hierbei wird das Prinzip des K-nearest-neighbors (KNN) verwendet. Über den gesamten Ablauf muss permanent geprüft werden, ob die Eingangsdaten korrekt sind und der Entscheidungsbaum anhand der Daten eine sinnvolle Empfehlung geben kann. Ist dies nicht der Fall, wird der Automationsprozess abgebrochen. Es benötigt also einige Züchtungen, bis dem System genügend Daten zur Verfügung stehen und es sinnvolle Empfehlungen geben kann.

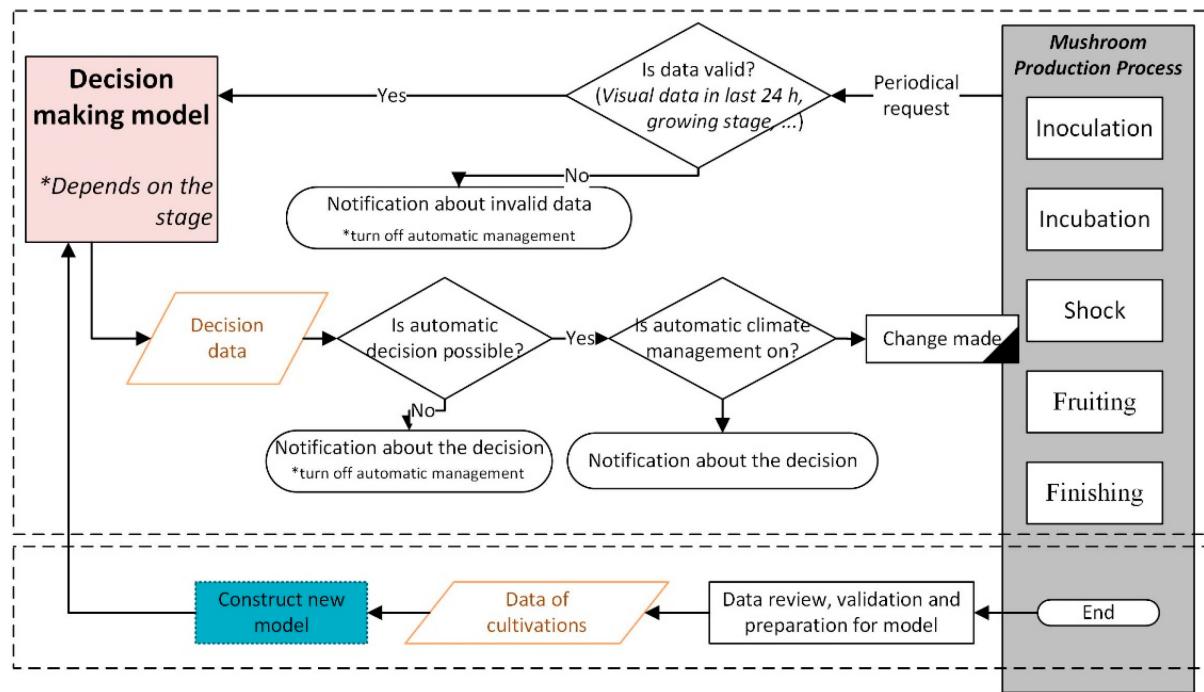


Abbildung 6: Prozess der Entscheidungsfindung[18]

Obwohl die Studie mit einer geringen Anzahl von Testdaten arbeiten musste, waren die Ergebnisse erfolgsversprechend. Der Algorithmus konnte mit überwiegender Sicherheit bestimmen, in welcher Wachstumsphase sich die Pilze befinden und ob eine Änderung des

Klimas notwendig ist. Mit einem erhöhten Datensatz werden dann auch die empfohlenen Änderungen immer sicherer.

**Werkzeuge in der Automation von Landwirtschaft mittels AI** [21] Experten Systeme wurden bereits seit den 80er Jahren in der Landwirtschaft eingesetzt. Basierend auf dem Fachwissen und den Meinungen von Expertinnen und Experten sollen diese Systeme Farmerinnen und Farmern bei allen Teilen des Anbaus unterstützen und wichtige Informationen liefern. Diese Systeme laufen meist nicht eigenständig und benötigen Input sowie die Einschätzung der Landwirtinnen und Landwirte, um genutzt werden zu können.

In einem Journal des Verlags KeAi fassten verschiedene Forschende aus Indien mögliche Anwendungen von KI in der Landwirtschaft zusammen . Künstliche neuronale NetzwerkeKünstliche Neuronale Netzwerke sind hierbei die bekanntesten und werden für verschiedene Zwecke genutzt. ANN's benötigen zwar viele Daten, können jedoch aus diesen zu großen Teilen ihre eigenen Schlüsse ziehen. So können Zusammenhänge und Denkweisen erkannt werden, die bisher nicht berücksichtigt wurden. Zudem können sie mit weiteren Daten trainiert werden. Der Output bleibt nicht konstant und kann mit dem System wachsen, ohne weitere Programmierarbeit aufzuwenden. So kann beispielsweise der Reifegrad von Getreide, Pflanzenerkennung oder die Verfügbarkeit von Wasser eingeschätzt werden. In einer Studie zur Abschätzung der Maisernte wurden Multilayered Feedforward ANN's und einem Conjugate Gradient Descent Algorithm zur Gewichtung der Verbindungen verwendet.[22] Fuzzy Logic Fuzzy Logic hingegen wurde bei der Verbesserung eines Expertensystems für Sojabohnen eingesetzt. Da viele Werte wie Pestbefall oder Bodenqualität nicht einfach als binärer Wert ausgegeben werden können, wurde sich hier für einen Fuzzy-Logik-Ansatz entschieden.[23]

Oft werden dabei sogenannte positivistische und systematische Methoden kombiniert. Positivistische Methoden sind eher "hands-on", und die Algorithmen funktionieren ähnlich wie das Vorgehen von Expertinnen und Experten. Bei systematischen Methoden hingegen ist das System mehr auf sich allein gestellt und soll eigene Wege finden, um Ergebnisse zu erzielen. Eine einzelne Methode ist in manchen Fällen ineffizient oder nicht genau genug. In solchen Fällen kann eine Kombination sinnvoll sein, um alle wichtigen Bereiche abzudecken.

**Zusammenfassung** Die verschiedenen Produkte und Studien zeigen, dass Automatisierungsprozesse in der Pflanzenpflege bereits jetzt von immenser Bedeutung sind und unter der Überwachung von Expertinnen und Experten den Prozess vereinfachen und bessere Ergebnisse erzielen können. Sensoren, regelbasierte Systeme, Fuzzy-Logik, Künstliche neuronale Netzwerke (ANN's), Mustererkennung und Bildverarbeitung/Maschinenvision werden alle im Bereich der Landwirtschaft eingesetzt, oft in Kombination. Bei der Entwicklung eines neuen Produkts kann auf eine Vielzahl von Werkzeugen und Technologien

zurückgegriffen werden, um das gewünschte Ergebnis zu erzielen. Diese bereits implementierten Technologien können zu großen Teilen auch im privaten Bereich eingesetzt werden, um die Pflanzenpflege zu erleichtern.

## 2.3 Theoretische Grundlage des Machine LearningMachine Learning Verfahrens

Im vorangegangenen Abschnitt wurde aufgezeigt, dass viele verschiedene algorithmische Verfahren verwendet werden, um die Pflanzenpflege zu automatisieren. In diesem Abschnitt wird genauer beleuchtet, wie diese Verfahren funktionieren und in welchen Anwendungsbereichen sie eingesetzt werden sollten.

Generell wird zwischen überwachtem (Supervised) und unüberwachtem (Unsupervised) Lernen unterschieden. Überwachtes Lernen bedeutet hierbei, dass ein Mensch manuell Ergebnisse überprüft und eine Richtung vorgibt, während beim unüberwachten Lernen der Algorithmus vollkommen eigenständig seine Ergebnisse erzielt. Bei komplexen Projekten wie diesem ist es von Vorteil, Technologien aus beiden Arten zu kombinieren, um sowohl Expertenwissen als auch maschinelle Erkenntnisse zu vereinen.

Im Folgenden werden drei verschiedene Machine-Learning-Ansätze vorgestellt. Alle können bei der Entscheidungsfindung in der Pflanzenpflege hilfreich sein und können unterschiedlich kombiniert werden, um bestmögliche Ergebnisse zu erzielen.

**Fuzzy Logic** Über den verschiedenen Methoden steht die Entscheidung, ob Fuzzy- oder Boolesche Logik verwendet werden sollte. Anders als bei Booleschen Entscheidungen ermöglichen Fuzzy-Logic-Algorithmen eine Entscheidung bei ungenauen Werten. Numerische Zustände wie 21 Grad Celsius können in linguistische Zustände (Sets) wie "kalt" oder "warm" umgewandelt werden.[24] Reale Szenarien, bei denen menschliche Einschätzungen benötigt werden, können so wesentlich besser behandelt werden.

Fuzzy Logic basiert immer auf einer Architektur mit vier Teilen.[25] Um es deutlicher zu machen, wird der Ablauf anhand einer automatischen Klimaanlage aufgezeigt:

1. Im Regelwerk werden alle Abläufe von Expertinnen und Experten festgesetzt. Im Normalfall werden Fälle in "stark positiv", "positiv", "neutral", "negativ" und "stark negativ" eingeteilt. Ist es sehr heiß, schalte die Klimaanlage auf maximale Leistung an. Ist es nur warm, reicht eine geringere Leistung. Ist es zu kalt, schalte die Heizung an.
2. Während der Fuzzification werden aus den eindeutigen Inputs, wie der gemessenen Gradanzahl, unscharfe Mengen gebildet. So wird beispielsweise alles zwischen drei und zehn Grad als "moderat kalt" eingestuft.

3. Die Inferenzmaschine ordnet den unsicheren Input einer Regel zu. Wenn es warm ist, schalte die Klimaanlage auf moderater Leistung an.
4. Während der Defuzzification wird die unsichere Menge und die Regel zu einem klaren Wert konvertiert. Die Klimaanlage wird mit 65 Prozent Leistung angeschaltet. Da verschiedene Inputs zu Sets zusammengefasst werden, ist ein Informationsverlust unvermeidbar.

Fuzzy Logic ist um einiges flexibler als gewöhnliche Entscheidungsbäume und ist gut geeignet, um menschliches Handeln umzusetzen. Verschiedene numerische Variablen können zu Sets verbunden werden, um ein komplexes Regelwerk zu erstellen. Bei den Inputs SSpannung und die SSchnelligkeit der Änderungsrate der Spannung", beispielsweise NM für "Negative Medium" und PL für "Positive Large", kann das Regelwerk für den Output der Spannung folgendermaßen aussehen:

		Voltage						
		NL	NM	NS	ZE	PS	PM	PL
Del	NL	PL	PL	PL	PL	PM	PS	ZE
	NM	PL	PL	PM	PM	PS	ZE	NS
	NS	PL	PM	PS	PS	NS	NM	NL
	ZE	PL	PM	PS	ZE	NS	NM	NL
	PS	PL	PM	PS	NS	NS	NM	NL
	PM	PM	ZE	NS	NM	NM	NL	NL
	PL	ZE	NS	NM	NL	NL	NL	NL

Abbildung 7: Regelwerk für eine Spannungsausgabe. [24]

Ergebnisse sind hierbei nicht zwingend optimal, aber ausreichend, und es ist einfacher, mit komplexen Situationen umzugehen. Fuzzy Logic sollte nicht benutzt werden, wenn die Ergebnisse exakt sein müssen. Auch dauern die Berechnungen im Vergleich zu anderen Machine-Learning-Algorithmen lange.

### 2.3.1

Decision Tree Decision Trees sind eines der einfachsten zu implementierenden Machine-Learning-Verfahren. Sie gehören zu den überwachten Lernalgorithmen, bei denen Daten vom Ersteller gelabelt werden, und die Maschine trifft keine eigenen Einschätzungen darüber, was richtig und falsch ist. Diese Methode kann sowohl für klassifizierte als auch für regressionsbasierte Aufgaben verwendet werden.

Wenn wir beispielsweise vorhersagen möchten, ob eine Pflanze wachsen wird, sind verschiedene Daten wichtig: Temperatur, Bodenqualität, Sonneneinstrahlung, Wasserzufuhr, etc. Alle diese Daten bilden den Root-Node, der dann gesplittet wird. Hierbei wählt der Algorithmus eine Eigenschaft aus, die den Datensatz am eindeutigsten teilt. Während einige Pflanzen auch bei niedrigen Temperaturen wachsen können, kann keine Pflanze ohne Wasserzufuhr wachsen. Diese Variable teilt den Datensatz in Sub-Nodes auf. Wenn diese Nodes weiter unterteilt werden können, bezeichnet man sie als Entscheidungs-Nodes, sonst als Blatt-Nodes. Sind bestimmte Eigenschaften nicht relevant, werden sie vom Algorithmus entfernt, das sogenannte "Pruning". So entstehen Branches, die die Entscheidung des Systems bestimmen.

Decision Trees können bei großen Datensätzen sehr komplex werden und den Hintergrundrauschen (nicht relevante Eigenschaften) nicht von den relevanten Mustern unterscheiden. Techniken wie Pruning und die Begrenzung der Tiefe können hierbei helfen. Auch können kleine Änderungen des Datensatzes zu gänzlich anderen Bäumen führen. Die Herausforderung besteht darin, die entscheidenden Attribute zu filtern, sodass nicht für jeden Fall des Trainingssets ein Blatt-Node gebildet wird, sondern das System andere Daten sinnvoll einschätzen kann.

Wann ein Node in welche Subnodes gesplittet wird, ist der wichtigste Entscheidungsfaktor für die Effektivität des Algorithmus. Das System muss hierbei entscheiden, welche Attribute eine hohe Entropie aufweisen, das Ergebnis zufällig beeinflussen, und aus welchen man einen hohen Informationsgewinn erlangen kann, um den Datensatz möglichst homogen zu splitten.[26] Diese Einschätzung muss von einem Experten vorgenommen werden. Dabei gibt es verschiedene Algorithmen, die je nach Anwendung unterschieden werden sollten, unter anderem:

- ID3: Der Algorithmus entscheidet bei jeder Iteration, welches Attribut die niedrigste Entropie und den höchsten Informationsgewinn hat, und bildet darauf ein Subset.
- C4.5: Hier wird der normalisierte Informationsgewinn verwendet und über die Entropie gestellt.
- CART: Ein Decision Tree, der für die Klassifikation optimiert ist.
- MARS: Ein Decision Tree, der für die Regression optimiert ist.

### 2.3.2 Artificial Neural Networks

Das menschliche Gehirn filtert aus einer schier unendlichen Masse an Informationen mit geringem Aufwand unwichtige Informationen heraus und führt schnelle Berechnungen durch. Dies geschieht durch einen scheinbar simplen Vorgang, bei dem Rezeptoren Signale an Effektoren senden und diese Informationen weiterleiten. Diese Einheiten werden als Neuronen bezeichnet. Wenn dieselben Wege immer wieder benutzt werden, wächst das Netzwerk an Verbindungen und wird effektiver.[27]

Künstliche neuronale Netze sind stark vom menschlichen Gehirn inspiriert und sollen aus großen und komplexen Datenmengen sinnvolle Ergebnisse erzielen. Dies geschieht in drei Schichten:[28]

- Eingabeschicht: Hier werden die eingegebenen Daten aufgenommen und an die richtigen Eingabeneuronen weitergeleitet. Diese gewichten die Eingaben und geben sie an die nächste Schicht, bestehend aus einem oder mehreren Neuronen, weiter.
- Verborgene Schicht: Diese Schicht kann aus beliebig vielen Ebenen bestehen. Die angenommene Information wird hier beliebig oft weitergewichtet und weitergeleitet.
- Ausgabeschicht: Dies ist die letzte Schicht. Die gewichtete Information wird gespeichert und ausgegeben.

Ein Neuron sammelt die Informationen aller vorangegangenen Neuronen, mit denen es verbunden ist, und multipliziert diese anhand der Gewichtung. Dann wird dieser Wert durch eine Aktivierungsfunktion auf einen Wert zwischen Null und Eins normalisiert.[29]

Die Gewichtung wird zunächst zufällig verteilt. Wenn sie den richtigen Output generiert, bleibt die Gewichtung bestehen, wenn nicht, wird sie angepasst. Es gibt verschiedene Möglichkeiten, dem System zu sagen, ob der generierte Output korrekt ist:[30]

- Beim Supervised Learning werden für jedes Beispiel die korrekten Output-Daten vorgegeben. Jede Mail im Trainingsset wird beispielsweise als Spam oder Nicht-Spam gekennzeichnet. Das System kann so erkennen, ob es die richtigen Indizes zur Erkennung verwendet.
- Beim Unsupervised Learning ist der Output oft vage und zu komplex zu beschreiben. Das System gruppiert die Daten selbst und erstellt Zusammenhänge. Beim Clustering werden Daten beispielsweise in Gruppen geordnet, um unbemerkte Muster zu erkennen.[31] Diese Methode wird oft für Empfehlungen in Online-Shops verwendet, um unbeachtete Patterns zu nutzen oder in Data Science, um Forschern zu helfen neue Zusammenhänge zu erkennen.
- Das Reinforcement Learning liegt zwischen den beiden Varianten. Das System erhält Informationen darüber, ob der Output zielführend war oder nicht. [32] Mit jedem

Trainingsdurchlauf soll die KI so besser werden. Bei einem Rennspiel entscheiden Gas und Lenkradrichtung über den Spielverlauf. Kann das Fahrzeug schneller ans Ziel gelangen, hat es die richtigen Verbindungen geschaffen. Die Reward-Funktion, welche Parameter belohnt werden und wünschenswert sind, kann über den Verlauf des Trainings verändert werden, um bestimmtes Verhalten zu fördern. Ähnlich wie ein Tennistrainer, der zur Aufgabe gibt lediglich Rückhand-Schläge zu üben, verliert auch die AI einmal erlerntes Wissen nur langsam und kann so neue Techniken lernen, im Realfall sehen, wann diese hilfreich sind, und so gezielt einsetzen.

ChatGPT Das Konzept der Generalisierung ist von entscheidender Bedeutung in der künstlichen Intelligenz. Ähnlich wie ein Mensch kann eine KI bessere Ergebnisse erzielen, wenn der Rahmen enger gesteckt ist. Eine KI, die beispielsweise nur Bilder von Hunden in einem bestimmten Album erkennen soll (das Trainingsset) wird dieses Ziel verhältnismäßig schnell und zuverlässig erreichen. Wenn jedoch das Ziel darin besteht, auf zufälligen Bildern Hunde zu erkennen, ist eine andere Gewichtung und ein breiterer Rahmen nötig. Die erstgenannte KI wird zwar im Trainingsset nahezu perfekt sein, aber in anderen Situationen kaum zu gebrauchen sein. Eine eher auf Generalisierung fokussierte KI wird generell gute Ergebnisse liefern, aber nie so genau wie die spezialisierte im Trainingsset. Hier liegt es am Entwickler, den richtigen Grad an Generalisierung zu wählen, um die gewünschten Ergebnisse zu erzielen.[33]

Es gibt viele verschiedene Formen von Künstlichen Neuronalen Netzen, und je nach Anwendung muss die Art und die genaue Implementierung spezifiziert werden, um optimale Ergebnisse zu erzielen. Künstliche Neuronale Netze sollten verwendet werden, wenn die Trainingsdaten zu komplex sind oder die Datenstruktur nicht vollständig verstanden wurde. Sie bilden die Grundlage für viele weitere komplexe KI-Technologien.

### 2.3.3 Pattern Recognition

Pattern Recognition ist eine Technik des maschinellen Lernens, die das automatische Klassifizieren von Daten in Objekte und Klassen ermöglicht. Der Lernprozess kann dabei sowohl überwacht (supervised) als auch unüberwacht (unsupervised) erfolgen. Es gibt verschiedene Möglichkeiten, Pattern Recognition zu implementieren:[34]

- Statistical Pattern Recognition: Hier werden aufgezeichnete statistische Daten verwendet, um ein Modell zu erstellen.
- Syntactic Pattern Recognition: Diese Methode wird vor allem in der Texterkennung und -generierung eingesetzt. Einzelne Teile, wie Buchstaben oder Wörter, werden mit ihren Vorgängern und Nachfolgern verglichen, um ein Muster zu erkennen.
- Neural Pattern Recognition: Hierbei wird ein künstliches neuronales Netzwerk (ANN)

verwendet, um Muster zu identifizieren. Diese Methode ist wesentlich aufwendiger als die anderen Technologien..

- Template Matching: Dies ist die einfachste Methode, bei der Testdaten mit einem Beispielsatz verglichen werden. Es wird lediglich die Ähnlichkeit zum Vorlagenmuster bewertet.
- Fuzzy Based: Hier werden einzelne ähnliche Attribute zu Mustern zusammengefasst. Diese können dann verwendet werden, um Testdaten zu klassifizieren und zuzuordnen.

Verschiedene Techniken können kombiniert werden, um genauere Ergebnisse zu erzielen oder die jeweiligen Vorteile zu nutzen.

Die Anwendungen von Pattern Recognition sind vielseitig und reichen von Bild-, Text- und Spracherkennung bis hin zur Analyse des Börsenmarktes oder seismischer Aktivitäten. Besonders hilfreich sind diese Techniken bei großen, unübersichtlichen Datenmengen, bei denen Zusammenhänge nur schwer erkennbar sind.

#### 2.3.4 Zusammenfassung

Für optimale Ergebnisse sollten verschiedene Ansätze ausprobiert und getestet werden, im besten Fall in Kombination miteinander.

Die Verwendung von Decision Trees für grundlegende Entscheidungen wie das Gießen basierend auf dem Wasserstand ist eine gute Möglichkeit, die Pflanzen am Leben zu erhalten.

Um Wachstum zu optimieren und die Entscheidungen dynamisch an die Umgebung anzupassen, könnte ein Ansatz des Reinforcement Learning effektiv sein. Dieser Ansatz ermöglicht es dem System, aufgrund von Rückmeldungen aus der Umgebung selbstständig Entscheidungen zu treffen, um das gewünschte Ziel zu erreichen, ohne ein detailliertes Verständnis des zugrunde liegenden Prozesses zu benötigen.

In späteren Iterationen, insbesondere wenn mehr Sensordaten verfügbar sind, kann es sinnvoll sein, weitere Machine-Learning-Modelle zu implementieren, wie beispielsweise neuronale Netzwerke. Diese könnten komplexere Muster in den Daten erkennen und möglicherweise genauere Entscheidungen treffen, um das Pflanzenwachstum zu optimieren. Es ist wichtig, die verschiedenen Ansätze zu testen und zu evaluieren, um diejenigen zu identifizieren, die die besten Ergebnisse liefern.

### 2.4 Bilderkennung

Die Verwendung maschinelner Bilderkennung, die sich an der Funktionsweise des menschlichen Sehsinns orientiert, hat sich als äußerst effektiv erwiesen. Durch die hierarchische

Verarbeitung werden zunächst irrelevante Informationen herausgefiltert, und die Aufmerksamkeit auf grobe Merkmale gelenkt.[35] Diese Abstufung der Priorität erleichtert die Arbeit der Bilderkennung erheblich, da Informationen nur aus wenigen Teilen eines Bildes gewonnen werden müssen, und Details nur bei Bedarf und Interesse berücksichtigt werden.

Zuverlässige Bilderkennungssoftware, welche tatsächlich auf Merkmale achtet und Anomalien ignorieren können, benötigen enorme Datensätze. Eine der größten Datenbanken, ImageNet, besteht aus über 14 Millionen genauestens beschrifteter Bilder. [36]

Zuverlässige Bilderkennungssoftware benötigt jedoch enorme Datensätze. Eine der größten Datenbanken, ImageNet, besteht aus über 14 Millionen genau beschrifteten Bildern. Neuronale Netzwerke, insbesondere Convolutional Neural Networks (CNNs), sind die am häufigsten verwendete Technik für die Bilderkennung.[37] Diese Netze verbessern den Prozess, indem sie die Pixel in Bezug zu ihren Nachbarn setzen können, und so ein besseres Verständnis für die Struktur des Bildes ermöglichen.[38]

Für das Beobachten und Einschätzen des Pflanzenwachstums verwenden Forscher aus Korea eine Mischung aus RGB- und Tiefenbildern.[39] Zur Objekterkennung wurde das Datenset von Imagenet benutzt. Durch verschiedene Filter und Bildfusionstechniken können aus diesen Bildern sieben Bilder mit unterschiedlichen Informationsgehalten generiert werden. Durch Objekterkennung und Bildverarbeitung können wichtige Merkmale wie der Stamm-Durchmesser oder Abstände zwischen den Pflanzenteilen bestimmt werden.

Unternehmen wie Roboflow bieten spezialisierte Dienste für die Einschätzung des Pflanzenwachstums an.[40] Mithilfe von Trainingsdaten, die aus beschrifteten RGB-Bildern bestehen, können Algorithmen entwickelt werden, die das Pflanzenwachstum anhand neuer Bilder schätzen können. Dies ermöglicht eine automatisierte und präzise Überwachung des Pflanzenwachstums über die Zeit hinweg.

### 3 Methodik

Durch die wenigen beteiligten Personen ist ein simplifiziertes Wasserfallmodell ausreichend, um das Projekt durchzuführen.

Nach einer Analyse der Anforderungen und der Konzeption, bei der verschiedene Technologien verglichen werden und die grobe Struktur des Projektes festgehalten wird, geschieht die Umsetzung und daraufhin die Evaluation. In der Evaluation werden Faktoren wie das Einhalten der Anforderungen und die Codequalität überprüft.

### 3.1 Anforderungsanalyse

Da es sich um ein Forschungsprojekt handelt, können wirtschaftliche Aspekte einer Anforderungsanalyse zunächst vernachlässigt werden. Die Struktur der Analyse wird sich auf die Arbeit von Pandey, Suman und Ramani [41] und dem Figma Template[42] für Use-Cases berufen. Sie bieten eine klare Struktur zur Identifizierung und Beschreibung der verschiedenen Anwendungsfälle für das Forschungsprojekt zur automatisierten Pflanzenpflege. Die definierten Use Cases ermöglichen es, die Bedürfnisse und Ziele der potenziellen Nutzenden sowie die erforderlichen Funktionen des Systems genau zu erfassen.

#### 3.1.1 Funktionale Anforderungen

##### Use Case 1: Automatische Bewässerung

- Kunde: Pflanzenneuling
- System: Automatische Wasserpumpe
- Ziele des Kunden: Automatisches Gießen für eine gesunde Pflanze, er spart Zeit und muss sich keine Gedanken machen müssen
- Trigger: Neue Pflanze

Basic Flow: Der Kunde erhält eine neue Pflanze. Er weiß den Namen, hat aber sonst keine weiteren Informationen. Er schließt die Pflanze, an das System an. Es wird automatisch die Bodenfeuchtigkeit ausgemessen und auf einem geeigneten Level gehalten.

Alternate Flow: Der Kunde kennt bereits die gewünschte Bodenfeuchtigkeit. Das System nimmt diese auf und hält die Erde auf diesem Level.

##### Use Case 2: Überwachung der Pflanze

- Kunde: Interessierter Gärtner
- System: Automatische Sensoren
- Ziele des Kunden: Detaillierte Informationen über seine Pflanze, um diese besser zu pflegen
- Trigger: Neue oder ungesunde Pflanze, reines Interesse

Basic Flow: Der Kunde schließt eine Pflanze an das System an. Das System misst regelmäßig alle nötigen Daten und gibt diese auf Anfrage aus. Der Kunde kann verschiedene Anfragen wie letzter Wert, Durchschnitte oder generelle Informationen über die Pflanze, Sonneneinstrahlung, Luftfeuchtigkeit, Bodenfeuchtigkeit und Temperatur, im Vergleich zum Optimum abfragen. Auch soll es mithilfe von Bildern Pflanzenwachstum über die letzten Wochen anzeigen können.

### Use Case 3: Optimieren

- Kunde: Ergebnisorientierter Gärtner
- System: Automatische Sensoren und Wasserpumpe
- Ziele des Kunden: Die Pflanze soll möglichst schnell wachsen und blühen oder Früchte tragen.
- Trigger: Nicht optimale Werte der Pflanze bremsen das Wachstum aus.

Basic Flow: Das System erkennt, dass bereits festgelegte Werte wie gewünschte Sonneneinstrahlung nicht optimal sind, da die Pflanze langsamer als gewöhnlich wächst. Es gibt Tipps um die Umweltsituation der Pflanze zu verändern und variiert die Frequenz des Gießens. Die neuen Daten werden verwendet, um weitere genauere Verbesserungsmöglichkeiten zu erkennen.

#### 3.1.2 Nicht-funktionale Anforderungen

Die Anforderungen, die sich aus dem Konzept des Projekts, der technischen Umsetzung und verschiedenen Erfahrungen ergeben, bilden eine solide Grundlage für die Entwicklung des Systems zur automatisierten Pflanzenpflege. Im Folgendenden finden sich die nicht-funktionalen Anforderungen zusammengefasst:

- Sprachgesteuerte Funktion: Das System soll vollständig durch Sprachbefehle gesteuert werden können, um eine intuitive Benutzererfahrung zu bieten.
- Reaktionszeit: Die Abfragen sollen in unter drei Sekunden geschehen, um dem Nutzer ein reibungsloses Erlebnis zu bieten. Anders als für Webseiten gibt es keine Studien, wie lange Nutzende gewillt sind bei einem Sprachassistenten auf eine Antwort zu warten. Aus Erfahrung mit verschiedenen Apps, sind drei Sekunden eine annehmbare Zeit für eine Reaktion.
- Regelmäßige Überprüfung des Pflanzenzustands: Das System soll den Zustand der Pflanze alle fünf Minuten überprüfen, um potenzielle Probleme schnell zu erkennen und zu korrigieren.
- Echtzeit-Datenerfassung und -speicherung: Das System soll jede Minute Werte der Pflanze erhalten und speichern, um Messfehler auszugleichen und zuverlässige Daten für Entscheidungen bereitzustellen.
- 24/7 Verfügbarkeit: Das System soll dem Benutzer jederzeit zur Verfügung stehen, um eine kontinuierliche Überwachung und Pflege der Pflanze zu gewährleisten.

- Automatische Geräteverbindung: Ein neues Gerät soll sich automatisch mit allen Diensten verbinden, ohne dass der Benutzer zusätzliche Eingaben vornehmen muss.
- Maximaler Ausfallzeitraum: Ein einzelnes Gerät darf (bei bestehender Internetverbindung und Stromzufuhr) maximal fünf Minuten ausfallen. So soll verhindert werden, dass über längere Zeit die Pflanze in nicht-optimalen Bedingungen wächst.

Später soll das System einmal die Woche Tipps zur Verbesserung der Umweltsituation geben. Die schnellsten Pflanzen zeigen jede Woche sichtbares Wachstum.[43] Ein späterer Machine Learning Ansatz kann so in sinnvollen Intervallen Änderungen zum Pflanzenwachstum geben und die Ergebnisse auswerten.

### 3.2 Grobkonzept

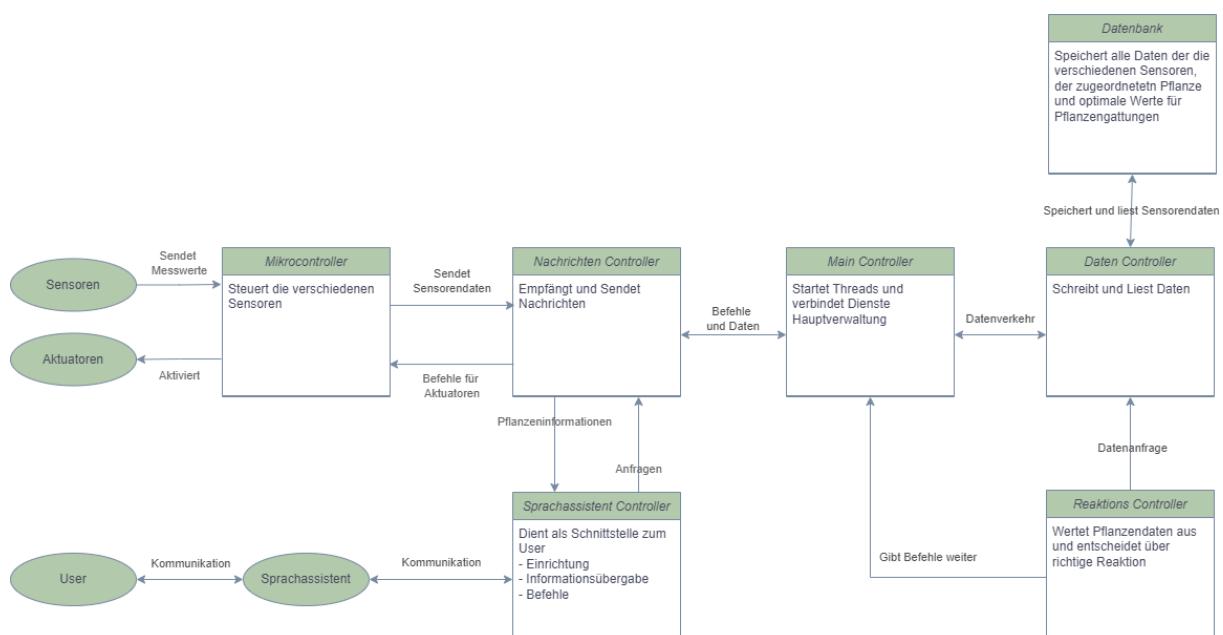


Abbildung 8: Flowchart der allgemeinen Software-Architektur

[11]

Das Grobkonzept der Architektur sieht neun verschiedene Teile vor.

- Sensoren
- Mikrocontroller
- Main Controller
- Nachrichten Controller
- Daten Controller

- Datenbank
- Reaktions Controller
- Sprachassistent Controller
- Sprachassistent

An einem Mikrocontroller sind verschiedene Sensoren und Aktuatoren angeschlossen. In regelmäßigen Abständen sammeln diese Umweltdaten und geben diese an den Controller weiter. Bei Bedarf kann dieser die Aktuatoren aktivieren. Drahtlos sendet der Mikrocontroller die Daten an den Nachrichten Controller weiter. Dieser sendet diese an den Daten Controller, welcher sie in die Datenbank schreibt. In regelmäßigen Abständen holt der Main Controller über den Daten Controller die letzten Daten ein und gibt diese an den Reaktions Controller weiter. Dieser entscheidet, ob die Umweltdaten von den optimalen Werten abweichen. Ist dies der Fall wird der Befehl an den Mikrocontroller oder den Sprachassistenten weitergegeben. Der Nutzende interagiert mit der Maschine ausschließlich über einen Sprachassistenten. Über diesen können Anfragen gestellt werden, und diese teilt dem Nutzer automatisch wichtige Informationen zum Zustand der Pflanze zu oder empfiehlt eine Änderung des Umfelds.

Als Sensoren benötigen wir zunächst Messapparate für Licht, Temperatur, Boden- und Luftfeuchtigkeit. Als Aktuatoren momentan lediglich eine Pumpe. Die Auswahl der Sensoren beruht sich auf die wichtigsten Umgebungsfaktoren für eine Pflanze.[44]

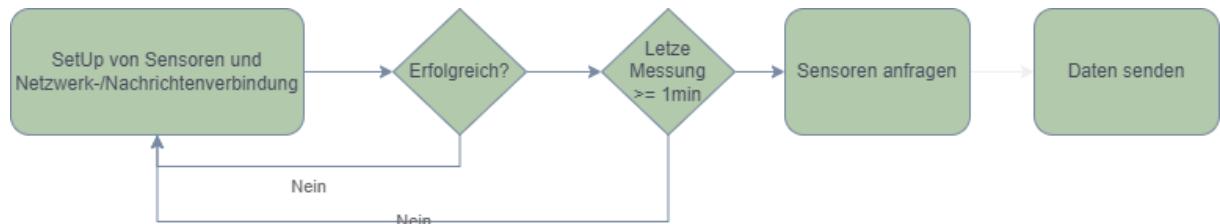


Abbildung 9: Flowchart des Mikrocontrollers

[11]

Der Mikrocontroller soll jede Minute die Daten einholen. Um die Funktionstüchtigkeit zu garantieren wird vor jeder Messung die Verbindung zu den Sensoren, der Netzwerkverbindung und dem Nachrichtendienst getestet. Funktioniert alles werden die Daten an den Nachrichten Controller gesendet. Erhält dieser diese speichert er sie in die Datenbank.

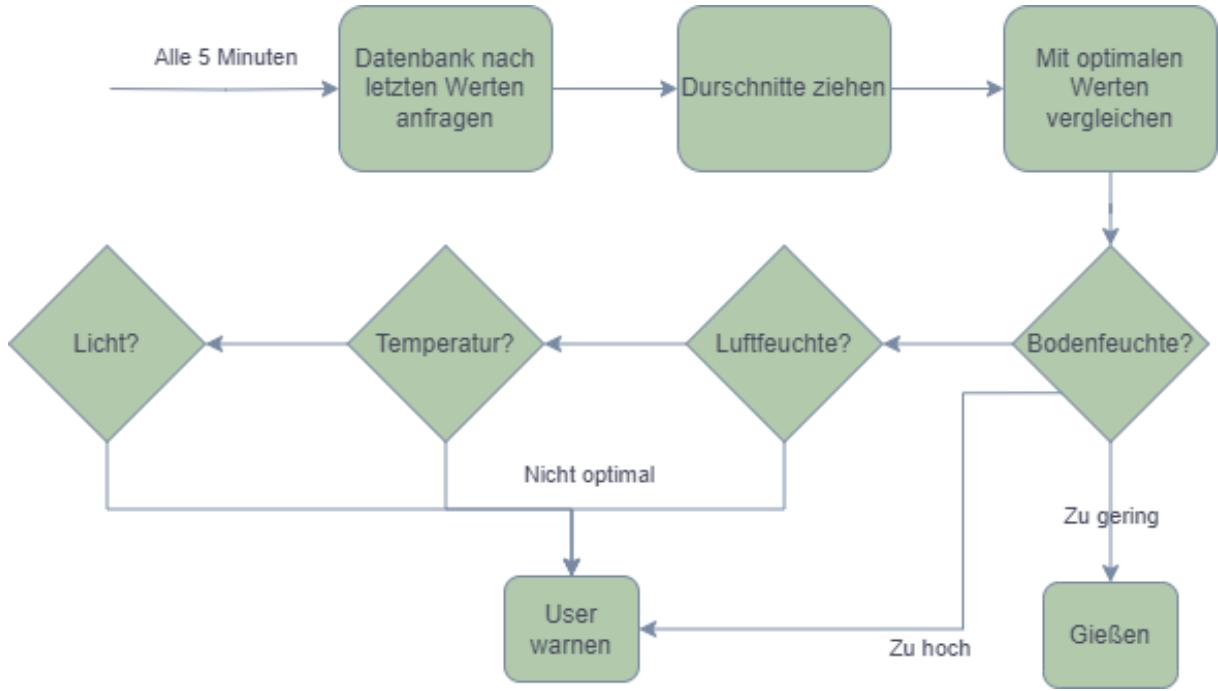


Abbildung 10: Flowchart des Main Controllers

[11]

Alle fünf Minuten überprüft der Reaktions Controller, ob die gesendeten Daten mit den optimalen Werten für diese Pflanze übereinstimmen. Ist dies nicht der Fall wird der Nutzende darauf hingewiesen oder die Pumpe aktiviert.

### 3.3 Feinkonzept

#### 3.3.1 Auswahl des Mikrocontrollers

Für die vorliegende Arbeit wurden spezifische Anforderungen identifiziert, die für das erfolgreiche Gelingen des Projekts von entscheidender Bedeutung sind. Zunächst ist eine zuverlässige Internetverbindung von höchster Priorität. Diese soll es ermöglichen, eine direkte Kommunikation mit dem Nachrichten Controller herzustellen, und so variabel für eine breite Palette von Anwendungsmöglichkeiten zu sein.

Des Weiteren war die Kompatibilität mit etablierten Bibliotheken ein wesentlicher Aspekt. Da verschiedene Sensoren benötigt werden, erleichtert die Verfügbarkeit einer Vielzahl von Bibliotheken die Entwicklung erheblich und ermöglicht eine einfache Softwaregestaltung, ohne aufwendige Workarounds.

Ein weiterer entscheidender Faktor war die Anzahl und Vielfalt der Anschlüsse. Um den Anforderungen des Projekts gerecht zu werden, war es wichtig, über ausreichend digitale und analoge I/O-Pins sowie verschiedene Kommunikationsschnittstellen zu verfügen.

Im Hinblick auf diese Anforderungen erwies sich der Arduino UNO R4 WiFi als optimale Lösung für das Projekt.[45] Durch die Integration des ESP32-S3 Moduls bietet er

die Möglichkeit, eine stabile drahtlose Verbindung herzustellen und somit eine Vielzahl von drahtlosen Anwendungen zu realisieren.

Darüber hinaus bietet der UNO R4 WiFi eine breite Unterstützung für gängige Bibliotheken und eine aktive Entwicklergemeinschaft, was die Entwicklung und Implementierung von Funktionen erleichtert und beschleunigt.

Die umfangreiche Anzahl an digitalen und analogen I/O-Pins sowie die Vielfalt der Kommunikationsschnittstellen ermöglichen eine flexible Konfiguration und Erweiterung des Systems, um den Anforderungen des Projekts gerecht zu werden.

Insgesamt erfüllt der Arduino UNO R4 WiFi die definierten Anforderungen für das Projekt und stellt somit die ideale Plattform für die erfolgreiche Umsetzung der gestellten Aufgaben dar.

### 3.3.2 Auswahl der Sensoren

Im Folgenden werden die Anforderungen und die darauffolgende Auswahl der Sensoren getroffen. Zu einer zuverlässigen Arbeitsweise und einer einfachen Integrierung in die Arduino-Umgebung kommen für den Typ des Sensors spezielle Anforderungen hinzu.

**Licht** Für die Lichtintensitätserkennung in diesem Projekt war es entscheidend, einen Sensor zu verwenden, der in der Lage ist, zwischen verschiedenen Lichtverhältnissen zu unterscheiden. Der LIGHT Unit Sensor erfüllt diese Anforderung durch die Integration eines Fotowiderstands und eines einstellbaren 10K-Widerstands.[46]



Abbildung 11: Lichtsensor  
[11]

Der Sensor ist in der Lage, die Lichtintensität zu erfassen und einen Schwellenwert für

die Lichtintensität festzulegen. Dies ermöglicht die differenzierte Erkennung von Sonnenlicht. Durch die Erfassung der Spannungsänderung erhält der Sensor Lichtintensitätsdaten durch Analog-Digital-Umwandlung.

Um eine präzisere Lichtintensitätserkennung zu erreichen, integriert dieser Sensor auch einen LM393 Dual-Differenzverstärker. Dieser wird verwendet, um die differentielle Spannung zwischen dem Fotowiderstand und dem druckempfindlichen Widerstand zu vergleichen.

Die Kompatibilität mit verschiedenen Softwareentwicklungsplattformen wie Arduino erleichtert die Integration des Sensors in das Projekt und ermöglicht eine einfache Programmierung und Steuerung.

**Temperatur und Luftfeuchtigkeit** Für das Projekt war es von großer Bedeutung, genaue Werte für Temperatur und Luftfeuchtigkeit zu erhalten, ohne auf abstrakte Messungen angewiesen zu sein. Die Verwendung eines Sensors mit einer echten I2C-Schnittstelle war entscheidend, um eine einfache Integration in das Projekt zu ermöglichen und die Anforderungen an die Messgenauigkeit zu erfüllen.

Der Sensirion SHT40 Sensor erfüllt diese Anforderungen vollständig. Mit einer typischen relativen Feuchtigkeitsgenauigkeit von  $\pm 1,8$  Prozent und einer typischen Temperaturgenauigkeit von  $\pm 0,2^\circ\text{C}$  liefert der Sensor zuverlässige Messwerte über einen breiten Bereich von Temperaturen und Luftfeuchtigkeiten. [47]

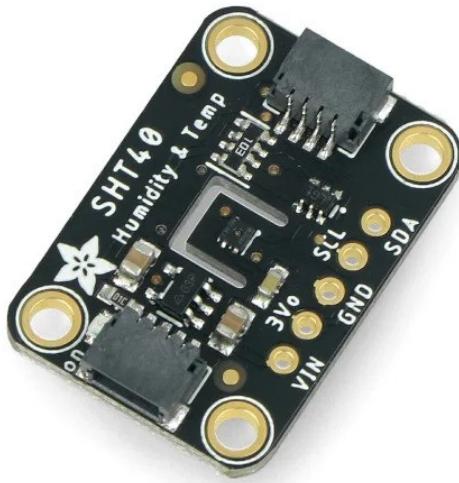


Abbildung 12: Temperatur- und Luftfeuchtigkeitssensor  
[11]

Durch seine I2C-Schnittstelle ermöglicht der Sensor eine einfache Kommunikation mit anderen Geräten und Mikrocontrollern. Die Unterstützung des Sensors durch Arduino erleichtert die Programmierung und Integration in verschiedene Projekte und Plattformen.

Zusätzlich bietet das Breakout-Board des Sensors eine einfache Handhabung und Integration in das Projekt, einschließlich SparkFun Qwiic-kompatibler STEMMA QT-Anschlüsse für den I2C-Bus.

**Bodenfeuchtigkeit und Pumpe** Die Watering Unit von M5Stack konnte die Feuchtigkeitsmessung durch Sensoren und eine Pumpe in einem Gerät vereinen. [48]



Abbildung 13: Pumpe und Bodenfeuchtigkeitssensor  
[11]

Die Wassereinheit nutzt die Kapazitätsmessungstechnologie für die Bodenfeuchteerkennung. Die Sensorplatten sind kapazitiv gestaltet, was im Vergleich zu anderen Methoden eine höhere Korrosionsbeständigkeit bietet. Dadurch wird die Langzeitgenauigkeit der Feuchtigkeitsmessung gewährleistet.

Mit der integrierten Wasserumwälzpumpe kann die Einheit automatisch die Bewässerung basierend auf den gemessenen Bodenfeuchtigkeitswerten steuern.

**Kamera** Für das Projekt ist es wichtig, eine Kamera zu finden, die ausreichend gute Bilder liefert, ohne außergewöhnliche Bildqualität zu erfordern. Da die Kamera statisch installiert ist, immer dieselbe Pflanze überwacht und lediglich die Maße des Objektes erkennen soll, nicht einzelne Zweige oder Farben, sind hohe Anforderungen an die Bildqualität nicht notwendig.

Die AZ Delivery Kamera erfüllt diese Anforderungen.[49] Mit einer Auflösung von 640 x 480 Pixeln liefert sie ausreichend scharfe Bilder für die Projektanforderungen. Mit einer Größe von 3,5 cm Seitenlänge und 3 cm Höhe lässt sie sich einfach an der Pflanze platzieren.



Abbildung 14: Kamera

[11]

Der CMOS-Bildsensor bietet die volle Funktionalität einer VGA-Kamera und eines Bildprozessors in einem kompakten Gehäuse. Die Kamera ist mit Arduino kompatibel und kann Bilder in verschiedenen Formaten liefern, die über die Serial Camera Control Bus (SCCB)-Schnittstelle gesteuert werden. Sie kann mit bis zu 30 Bildern pro Sekunde in VGA-Auflösung aufnehmen, wobei der Benutzer die volle Kontrolle über die Bildqualität, das Format und die Datenübertragung hat.

### 3.3.3 Nachrichtenprotokoll

Für das Nachrichtenprotokoll zwischen dem Controller und dem Arduino war es wichtig, einen Nachrichtendienst zu wählen, der einfach zu handhaben ist, gut auf Ausfälle reagiert und stabil ist. Hierbei standen unter anderem MQTT, HTTP/REST und CoAP zur Auswahl.

MQTT ist der de-facto Standart für IoT-Projekte. Ausschlaggebend sind hierbei folgende Gründe: [50]

- Einfachheit: MQTT ist ein leichtgewichtiges Protokoll, das speziell für den Einsatz in eingeschränkten Umgebungen entwickelt wurde. Es ist einfach zu implementieren und erfordert minimale Bandbreite
- Robustheit: MQTT bietet eingebaute Mechanismen zur Erkennung und Behandlung von Verbindungsabbrüchen. Dies stellt sicher, dass Nachrichten zuverlässig zugestellt werden und die Kommunikation stabil bleibt. Ist ein Empfänger zur Zeit der Sendung nicht verfügbar bleibt das Netz trotzdem bestehen und die Nachricht wird gesendet.

- QoS (Quality of Service): MQTT bietet drei QoS-Stufen, die garantieren, dass Nachrichten je nach Bedarf mindestens einmal, höchstens einmal oder genau einmal zugestellt werden.
- Paketgröße: Durch die geringe Größe der MQTT-Pakete ist das Protokoll ideal für Geräte mit begrenzten Ressourcen und Netzwerkbandbreite.

HTTP (HyperText Transfer Protocol) ist das Standardprotokoll für die Kommunikation im Web. REST (Representational State Transfer) ist eine Architektur, die auf HTTP aufbaut und weit verbreitet für Webservices genutzt wird.

Es ist im Vergleich zu MQTT schwergewichtiger, da es mehr Overhead erzeugt. Es ist nicht so effizient in Bezug auf Bandbreitennutzung und Latenz. HTTP-Verbindungen sind auch weniger robust gegenüber Verbindungsabbrüchen, da sie keinen eingebauten Mechanismus zur Wiederverbindung haben.

CoAP ist ein spezialisiertes Webprotokoll für eingeschränkte Geräte und Netzwerke. Es basiert auf dem UDP-Protokoll und ist darauf ausgelegt, eine effiziente und schnelle Kommunikation zu ermöglichen.

Es ist nicht so weit verbreitet wie MQTT und hat eine geringere Unterstützung in der Entwicklergemeinschaft. Ebenfalls ist es komplexer zu implementieren und zu warten, insbesondere in Bezug auf Sicherheitsaspekte.

Die Wahl fiel auf MQTT aufgrund seiner Einfachheit und Effizienz, die es ideal für IoT-Projekte macht. Es bietet robuste Mechanismen zur Handhabung von Verbindungsabbrüchen und sorgt für eine zuverlässige Kommunikation zwischen den Geräten. Im Vergleich zu HTTP/REST ist MQTT ressourcenschonender und besser für die Anforderungen eines IoT-Umfelds geeignet. CoAP wurde aufgrund seiner geringeren Verbreitung und höheren Komplexität verworfen, obwohl es in bestimmten Szenarien ebenfalls eine gute Wahl sein kann. MQTT bietet jedoch die beste Kombination aus Stabilität, Einfachheit und Effizienz für das vorliegende Projekt.

### 3.3.4 Programmiersprache

Arduino Projekte werden standardmäßig in C++ geschrieben. Um die nötigen Bibliotheken, wie MQTT oder spezifische Sensorenfunktionen zu benutzen, wurde von C++ nicht abgewichen.

Für die Implementierung der Controller wurde Python als Programmiersprache gewählt. Diese Entscheidung basierte auf den zahlreichen Vorteilen, die Python speziell für das Projekt bietet.[51]

Ein wesentlicher Grund für die Wahl von Python ist die umfangreiche Verfügbarkeit von Bibliotheken und Frameworks. Python bietet eine Vielzahl von Bibliotheken, die speziell für IoT-Anwendungen entwickelt wurden, wie beispielsweise ‘paho-mqtt’ für die MQTT-Kommunikation. Zudem existieren mehrere Web-Frameworks in Python, wie Flask

oder Django für Web-Interfaces, falls diese in einer späteren Iteration des Projektes integriert werden sollten.

Ein weiterer Vorteil ist die einfache Implementierung. Die Sprache ist bekannt für ihre klare und leicht verständliche Syntax, die es Entwicklern ermöglicht, schnell funktionalen Code zu schreiben und zu verstehen. Diese einfache Syntax und die hohe Abstraktionsebene von Python ermöglichen einen schnellen Workflow und Entwicklung, was besonders in der dynamischen und iterativen Welt der IoT-Projekte von Vorteil ist.

Python bietet zudem weitreichenden Support für KI-Funktionen. Die Sprache ist die bevorzugte Wahl für maschinelles Lernen und künstliche Intelligenz, mit umfangreichen Bibliotheken wie TensorFlow, Keras und scikit-learn, die problemlos in das Projekt integriert werden können. Auch für die Datenanalyse ist Python hervorragend geeignet, mit Bibliotheken wie Pandas und NumPy, die eine effiziente Analyse und Verarbeitung großer Datenmengen ermöglichen.

Die große und aktive Community stellt einen weiteren Pluspunkt dar. Diese Gemeinschaft bietet umfangreiche Unterstützung, von Tutorials und Foren bis hin zu Open-Source-Projekten und Beispielcodes. Zudem wird Python kontinuierlich weiterentwickelt und regelmäßig aktualisiert, wodurch die Sprache stets aktuell bleibt und sich kontinuierlich verbessert.

Java hingegen ist eine weit verbreitete Programmiersprache und bietet ebenfalls Plattformunabhängigkeit und eine große Anzahl an Bibliotheken. Allerdings ist der Entwicklungsprozess in Java oft komplexer und zeitaufwändiger. Java erfordert oft ausführlichere Boilerplate-Codes. Dies kann die Entwicklungsgeschwindigkeit und die Flexibilität bei schnellen Iterationen, die für IoT-Projekte notwendig sind, beeinträchtigen.

C ist eine sehr leistungsfähige Sprache, die besonders in der Embedded-Entwicklung weit verbreitet ist. Sie bietet eine feingranulare Kontrolle über die Hardware und ist extrem effizient. Allerdings ist die Entwicklung in C im Vergleich zu Python deutlich komplizierter und fehleranfälliger. Die manuelle Speicherverwaltung und die weniger abstrahierte Syntax erhöhen die Komplexität und das Risiko von Fehlern. Zudem fehlen in C viele der modernen Bibliotheken und Frameworks, die Python für IoT und KI bietet.

Zusammenfassend bietet Python durch seine umfangreichen Bibliotheken, einfache Implementierung, Unterstützung für KI-Funktionen und starke Community eine ideale Grundlage für die Entwicklung von IoT-Projekten. Es ermöglicht eine schnelle Entwicklung und einfache Integration von Hardware- und Softwarekomponenten.

### 3.3.5 Auswahl der Datenbank

Generell gibt es viele verschiedene Arten von Datenbanken. Hierzu gehören dokumentenbasierte wie etwa MongoDB, relationale Datenbanken wie PostgreSQL oder graphische wie Neo4j. Aufgrund der Popularität von IoT-Projekten und der hohen Nachfrage existieren inzwischen zahlreiche auf IoT spezialisierte Datenbanken, wie etwa InfluxDB,

TimescaleDB, CrateDB oder RethinkDB. Diese sind speziell für die großen Datenmengen angefertigt, welche jedoch einzeln nicht viel Speicher einnehmen. Im Normalfall werden nur einzelne Zahlen mit einem Tag gespeichert. Diese Time-Series Datenbanken verknüpfen neue Informationen automatisch mit einem Zeitstempel.

Für die Speicherung von Daten wurde InfluxDB als Datenbank ausgewählt. [52]

InfluxDB eignet sich aufgrund seiner Fähigkeit, einzelne Zahlenwerte mit genauen Zeitstempeln zu speichern, die leicht zu lesen und zu schreiben sind und schnelle Berechnungen ermöglichen. Dies ist besonders wichtig für die Anforderungen des Projekts, bei dem kontinuierlich Daten erfasst und analysiert werden müssen.

Ein wesentlicher Vorteil von InfluxDB liegt in seiner spezialisierten Auslegung für Daten. Im Gegensatz zu relationalen Datenbanken verwendet InfluxDB ein NoSQL-Modell, was das Schreiben von Daten erheblich beschleunigt. Komplexe Datenstrukturen sind nicht erforderlich, was die Effizienz erhöht.

Ein weiterer Grund für die Wahl von InfluxDB ist die Möglichkeit, einen eigenen Server zu hosten. Dies bietet Flexibilität und Kontrolle über die Datenbankumgebung, ohne zusätzliche Kosten. InfluxDB bietet eine kostenfreie Lösung, die es ermöglicht, die Datenbankumgebung an spezifische Projektanforderungen anzupassen und sie skalierbar zu halten.

Ein weiterer technischer Vorteil ist die Leistungsfähigkeit von InfluxDB bei der Skalierung und Verarbeitung großer Datenmengen. InfluxDB kann massive Mengen an IoT-Telemetrie-Daten ingestieren und indizieren, während es gleichzeitig Echtzeitanalysen und schnelle Abfragezeiten bietet. Dies ist entscheidend, da die Menge der IoT-Daten und die Anforderungen an deren Verarbeitung schnell wachsen.

Die Nutzung der Flux-Abfragesprache von InfluxDB ermöglicht es, Daten sowohl zu formen als auch abzufragen, was die Komplexität reduziert und die Effizienz erhöht. Flux bietet leistungsstarke Funktionen zur Datenanalyse direkt auf der Datenbankebene, ohne dass teure Abfragen zum Herunterladen und Modifizieren der Daten erforderlich sind.

Zusammenfassend bietet InfluxDB durch seine spezialisierten Funktionen für Zeitreihendaten, die schnelle und einfache Datenverarbeitung und die Fähigkeit zur Skalierung eine ideale Lösung für die Anforderungen des Projektes.

### 3.3.6 Sprachsteuerung

Für die Sprachsteuerung des IoT-Projektes wurde Alexa ausgewählt. Alexa ist der führende Anbieter von Smart-Speaker.[53] Außerdem ermöglicht es mit dem Alexa Skills Kit (ASK) die Erstellung eigener Programme, sogenannte Skills. Es ist hiermit um einiges offener als vergleichbare Sprachassistenten, wie etwa Google Home. Diese Skills sind Anwendungen, die eine interaktive Sprachschnittstelle verwenden, um Nutzern verschiedene Interaktionen zu ermöglichen. Nutzer können Alexa Skills verwenden, um alltägliche Aufgaben zu erledigen, wie zum Beispiel Nachrichten abzurufen, Musik zu hören oder Spiele zu spielen.

Zudem können Nutzer ihre Stimme nutzen, um cloud-verbundene Geräte zu steuern, wie beispielsweise das Einschalten von Lichtern oder das Anpassen des Thermostats.

Die Auswahl für Alexa beruht sich auf verschiedene Vorteile: [54]

- Alexa ist weit verbreitet und wird in zahlreichen Haushalten genutzt. Dies erhöht die Akzeptanz und die Nutzerfreundlichkeit, da viele Nutzer bereits mit der Funktionsweise von Alexa vertraut sind.
- Die einfache Verbindung von Alexa mit Amazon Web Services (AWS) ist ein entscheidender Vorteil. AWS bietet eine robuste Infrastruktur, die die Skalierbarkeit und Zuverlässigkeit des Projekts sicherstellt. Zudem ermöglicht die Nutzung von AWS Lambda eine effiziente und kostengünstige Serververwaltung, da keine eigenen Server erforderlich sind.
- ASK bietet eine umfassende Entwicklungsumgebung, die APIs, Tools, Code-Beispiele und technische Dokumentation umfasst. Dies erleichtert die Erstellung und Verwaltung von Skills erheblich. Insbesondere die Nutzung des Alexa-Developer-Consoles und der ASK SDKs in Programmiersprachen wie Node.js, Java und Python vereinfacht den Entwicklungsprozess.

Nutzer greifen auf Skill-Inhalte zu, indem sie Alexa auffordern, den Skill zu aktivieren. Nachdem das Aktivierungswort „Alexa“ gesagt wird, verarbeitet der Alexa-Dienst die Sprachdaten in der Cloud. Alexa erkennt die Sprache, bestimmt die Intention des Nutzers und sendet eine Anfrage zur Ausführung des entsprechenden Skills. Der Skill läuft als Dienst auf einer Cloud-Plattform und kommuniziert über eine HTTPS-Schnittstelle mit Alexa. Bei Aktivierung eines Alexa-Skills erhält der Skill eine POST-Anfrage mit einem JSON-Body, der die notwendigen Parameter zur Verarbeitung und Beantwortung der Anfrage enthält.

Alexa Skills nutzen Sprachinteraktionsmodelle, um festzulegen, welche Worte und Phrasen Nutzer sagen können, um den Skill zu steuern. Es gibt zwei Haupttypen von Sprachinteraktionsmodellen:

- Vorgefertigt: ASK definiert die zu verwendenden Worte und Phrasen. Zum Beispiel: „Alexa, schalte das Licht ein.“
- Benutzerdefiniert: Dieses Modell bietet Flexibilität und erfordert die Definition aller möglichen Arten, wie ein Nutzer eine Anfrage stellen könnte.

Der Entwurf eines Alexa Skills sollte natürlich, nutzerzentriert und visuell ansprechend sein. Die Entwicklung beginnt mit dem Design des Sprachinteraktionsmodells, gefolgt von der eigentlichen Implementierung des Skills. Die Verwendung des Alexa-Developer-Consoles oder Visual Studio Code ermöglicht das Testen des Skills während der Entwicklung.

ASK stellt Tools zum Testen des Skills zur Verfügung, darunter der Alexa-Simulator und die Alexa-App. Vor der Veröffentlichung muss der Skill den Zertifizierungsprozess durchlaufen, um sicherzustellen, dass er den Qualitäts-, Sicherheits- und Richtlinienanforderungen entspricht.

Nach der Veröffentlichung können Skills überwacht und analysiert werden. Die Entwicklerkonsole bietet Einblicke in die Nutzung, Analysen und Einnahmenverwaltung. Bei Problemen kann auf eine vorherige Version des Skills zurückgegriffen werden.

### 3.3.7 Testing

Zur Qualitätssicherung und Funktionsprüfung des IoT-Projektes wurden eine Mischung aus Cucumber-Tests und Unit-Tests eingesetzt. Diese Teststrategien ergänzen sich und bieten eine umfassende Abdeckung der Softwarequalität.

**Cucumber-Testing** Cucumber ist ein Werkzeug zur Verhaltensgesteuerten Entwicklung (Behavior-Driven Development, BDD). [55] Es ermöglicht die Erstellung von Tests in einer für Menschen lesbaren Sprache, meist in Gherkin-Syntax, die sowohl von Entwicklern als auch von Fachexperten verstanden wird. Ein typisches Cucumber-Szenario besteht aus drei Teilen:

- Given: Definiert den Anfangszustand
- When: Beschreibt die Aktion oder das Ereignis.
- Then: Beschreibt das erwartete Ergebnis.

Cucumber-Tests werden verwendet, um ganze Funktionen oder Anwendungsfälle zu testen. Sie helfen dabei, die Anforderungen des Geschäftsbereichs direkt mit den Entwicklungsergebnissen zu verknüpfen, indem sie sicherstellen, dass die entwickelten Funktionen den definierten Geschäftsregeln und -anforderungen entsprechen. Sie schaffen eine gemeinsame Sprache für die Beschreibung der Anforderungen und reduzieren so Missverständnisse

**Unit-Testing** Unit-Tests sind eine Form der Softwaretests, die einzelne Komponenten oder Einheiten des Quellcodes isoliert überprüfen. Diese Tests fokussieren sich auf kleine Codeabschnitte, wie Funktionen oder Methoden, und prüfen deren Korrektheit unabhängig vom Rest des Systems. Unit-Tests werden oft automatisiert und bieten schnelle Rückmeldung über den Zustand des Codes.

Unit-Tests werden verwendet, um kleine Codeeinheiten detailliert zu prüfen. Sie ermöglichen es, Fehler frühzeitig im Entwicklungsprozess zu erkennen und zu beheben, und tragen so zur Stabilität und Zuverlässigkeit des Codes bei.

Eine Kombination der beiden Testarten führt zu einer umfassenden Testabdeckung auf unterschiedlichen Abstraktionsebenen. Während Cucumber-Tests sicherstellen, dass die Software den Geschäftsanforderungen entspricht, garantieren Unit-Tests die technische Integrität der einzelnen Codeeinheiten. Zusätzlich verbessern sie die Wartbarkeit und Weiterentwicklung der Software. Cucumber-Tests dienen als lebende Dokumentation der Geschäftslogik, während Unit-Tests die technische Basis absichern. Dies erleichtert zukünftige Änderungen und Erweiterungen, da sowohl die Geschäftslogik als auch die Codeintegrität kontinuierlich geprüft werden.

### 3.3.8 Deployment Verfahren

Docker wurde für das Projekt ausgewählt, um eine effiziente Bereitstellung und Verwaltung der IoT-Anwendung zu gewährleisten. Die Hauptgründe für die Wahl von Docker sind wie folgt:

- Effizientes Herunterladen und Bereitstellen: Docker ermöglicht es, Anwendungen in Container zu verpacken, die alle erforderlichen Abhängigkeiten enthalten. Dadurch kann die Anwendung auf verschiedenen Rechnern identisch und effizient bereitgestellt werden. Das Herunterladen eines Docker-Images ist oft schneller und weniger fehleranfällig als das Einrichten einer Entwicklungsumgebung von Grund auf.
- Plattformunabhängigkeit: Mit Docker ist sichergestellt, dass die Anwendung in jeder Umgebung gleich läuft, sei es auf einem lokalen Rechner, einem Server oder in der Cloud. Dies reduziert die Probleme, die durch unterschiedliche Systemkonfigurationen entstehen können.
- Versionenkontrolle: Docker-Images können versioniert werden, was es ermöglicht, jederzeit zu einer vorherigen Version zurückzukehren. Dies ist besonders wichtig für die Nachverfolgbarkeit von Änderungen und die Sicherstellung der Stabilität der Anwendung.
- Einbindung von Tests: Docker unterstützt die Integration von automatisierten Tests, indem Testumgebungen schnell und konsistent bereitgestellt werden können. Dies fördert eine kontinuierliche Integration und kontinuierliches Testen (CI/CD), was die Qualität und Zuverlässigkeit der Anwendung erhöht.

Durch die Verwendung von Docker konnte die IoT-Anwendung effizient, konsistent und zuverlässig bereitgestellt und verwaltet werden, was zu einer verbesserten Entwicklungs- und Betriebsumgebung führte.

### 3.3.9 Serverinfrastruktur

Für das Projekt wurde ein Universitätsserver genutzt, der den Anforderungen entsprechend kostenfrei zur Verfügung gestellt wurde und alle notwendigen Richtlinien erfüllte. Der E-Mail-Verkehr mit dem Administrator Holger Szusz verdeutlicht die wesentlichen Aspekte der Servernutzung und deren Infrastruktur.

**Anforderungen** Kontinuierlicher Betrieb, um eine ununterbrochene Datenerfassung zu gewährleisten. Geringe Rechenleistung und Speicherplatz: Die Anwendung benötigt pro Stunde etwa 157 MB Speicher und verursacht einen minimalen I/O-Datenverkehr von ca. 3 KB die Sekunde. Docker-Infrastruktur: Es werden zwei unabhängig voneinander arbeitende Container benutzt, die Datenbank und der Controller. Geringer Netzwerkverkehr: Nach einer Stunde beträgt der Netzwerkverkehr etwa 1.9 MB eingehend und 1.8 MB ausgehend.

**Serverkapazität** Der bereitgestellte Server, ist mit leistungsstarken Komponenten ausgestattet:

- Prozessor: Intel i9, der für komplexe Berechnungen und Analysearbeiten geeignet ist.
- Grafikkarte: NVIDIA RTX 4090, die für grafikintensive Anwendungen und potenziell für maschinelles Lernen genutzt werden kann.

Mit diesen Ressourcen könnten auch spätere, aufwendige KI-Berechnungen durchgeführt werden.

Der Administrierende hat einen Benutzeraccount eingerichtet, der mit den gleichen Zugangsdaten wie das Lehr-Login genutzt werden kann. Der Zugriff auf den Server erfolgt über SSH, was auch ohne VPN möglich ist, falls Netzwerkprobleme auftreten.

Um den Zugriff zu erleichtern und die Sicherheit zu erhöhen, wurde empfohlen, SSH-Keys auf dem Login-Server zu hinterlegen, was die Authentifizierung vereinfacht und absichert.

## 3.4 Theoretisches Konzept

In diesem Teil wird die theoretische Vorgehensweise für die Bildauswertung und die Anpassung der Optimalwerte geschildert. Da eine Umsetzung den Rahmen der Arbeit übersteigt, werden lediglich Ansätze vorgestellt, um die dargelegten Problematiken zu lösen.

**Bildauswertung** Wie in vorangegangenen Projekten gezeigt wurde, ist Bildauswertung ein großes Feld, das je nach Anforderungen sehr komplex sein kann. Inzwischen gibt es jedoch verschiedene Bibliotheken, insbesondere in der Programmiersprache Python, die die Objekterfassung deutlich erleichtern. Da kleinere Messfehler in diesem Projekt der privaten Nutzung keine größeren Schäden verursachen und die Kamera stationär befestigt ist, ist es für einen ersten Prototypen ausreichend, Standardbibliotheken zu verwenden. Die folgende Methodik beruft sich größtenteils auf ein Projekt von Adrian Rosebrock.[56]

Generell muss bei einer Größeneinschätzung die Anzahl der Pixel in Relation zu einer Maßeinheit gebracht werden. Um das Modell zu trainieren, werden daher Bilder von verschiedenen Pflanzen benötigt, sowie ihre Maße (Höhe und Breite). Außerdem sollten die Objekte leicht zu identifizieren sein; ein einfarbiger Hintergrund vereinfacht daher das Training. Zudem sollte ein Referenzobjekt enthalten sein, beispielsweise ein Kreis auf dem Hintergrund, sodass das Objekt einfacher zu messen ist.

Anhand verschiedener Bilder können wir so die Relation des Objektes berechnen: Pixel geteilt durch wahre Länge. In diesem Fall werden die Berechnungen sowohl für die Pflanze als auch das Referenzobjekt durchgeführt. So können auch bei anderen Pflanzen zuverlässige Ergebnisse erzielt werden.

Um das Modell zu trainieren, sollten einzelne Bilder in Graustufen unterteilt werden. Mithilfe von OpenCV, einer Bibliothek für Bildverarbeitung und Computer-Vision, können Konturen erkannt und gekennzeichnet werden. Mithilfe dieser Konturen kann OpenCV einzelne Objekte identifizieren.[57]

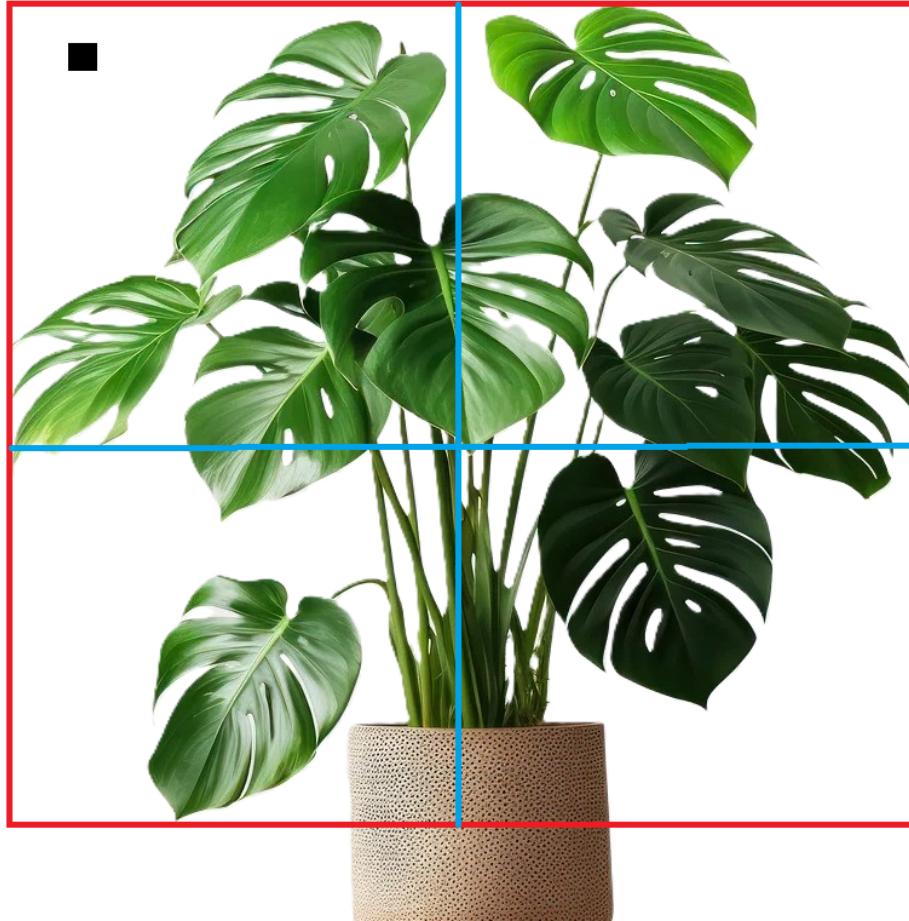


Abbildung 15: Messung einer Pflanze mittels Object Detection  
[11]

Sind die vier Seiten des Objektes eingegrenzt (rot), können Berechnungen durchgeführt werden, um die Maße zu bestimmen. Zunächst werden die Mittelpunkte der Seiten eingezeichnet, von diesen werden Linien zu den anderen Seitenmittelpunkten gezogen (blau), sodass ein Kreuz in dem Objekt-Rechteck entsteht. Diese zwei Linien werden in Relation zu den realen Daten gesetzt. Im Beispiel ist die blaue horizontale Linie 755 Pixel lang, die vertikale 682 Pixel. In echt wurde die Pflanze auf etwa 20,5cm breit und 19cm hoch gemessen. Das Quadrat ist bei 0,5cm 18 Pixel groß.

Wird das Modell, mit verschiedenen Pflanzen in verschiedenen Wachstumsphasen, mit ausreichend Daten trainiert, kann es, mithilfe des Referenzobjektes, sinnvolle Ergebnisse erzielen.

Sollten die initialen Ergebnisse nicht ausreichen, können mit mehr und genaueren Daten, verschiedenen Bildern aus unterschiedlichen Perspektiven und mehr Pre-Processing mithilfe der OpenCV-Bibliothek, die Ergebnisse weiter verbessert werden.

**Anpassung der Optimalwerte durch Machine Learning** Aufgrund der Aufgabe, einen nicht gelabelten Wert (Wachstum) zu verbessern, könnte ein Reinforcement-

Learning Ansatz sinnvoll sein.

Q-Learning ist eine Unterart des Reinforcement Learning. Es handelt sich, um eine stochastisches Modell, welches mithilfe der Anpassung von verschiedenen Faktoren einen Reward maximiert. Bei bedarf gibt es auch Q-Learning Modelle, welche Neuronale Netze benutzen.

Werte wie Temperatur, Bodenfeuchtigkeit und Sonneneinstrahlung sind hierbei die Input-Werte, welche verändert werden. Bei einem positiven Reinforcement-Ansatz versucht das System eine Belohnung zu maximieren, in diesem Fall das Wachstum der Pflanze.

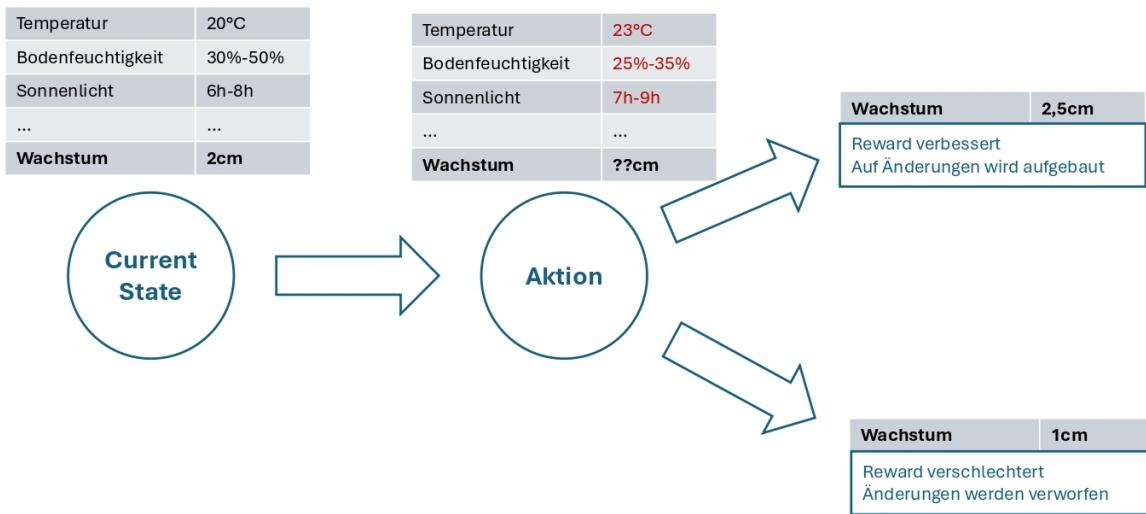


Abbildung 16: Genereller Workflow des Q-Learning Algorithmus

[11]

Die Python-Bibliothek Numpy erlaubt auf einfache Weise die Berechnungen für den Algorithmus. Hierbei werden die State-Werte der Umgebung zufällig verändert, der Reward (das Wachstum der Pflanze) bestätigt die getätigten Veränderungen oder widerlegt diese.[58] Eine Lern-Rate setzt fest wie stark es die Werte pro Iteration verändern darf. Dies ist von entscheidender Bedeutung für den Erfolg des Systems.

Nach genügend Iterationen mit der gleichen Pflanzengattung können so pro Pflanze verbesserte Umgebungsvariablen für die verschiedenen Wachstumsphasen bestimmt werden.

## 4 Implementierung

### 4.1 Projektarchitektur

Wie in Abschnitt 3 gezeigt, besitzt das Projekt verschiedene Teile, welche miteinander kommunizieren. Die Architektur aus dem Grobkonzept wird größtenteils übernommen.

Lediglich die Kommunikation mit dem Sprachassistenten wird von dem Main Controller übernommen.

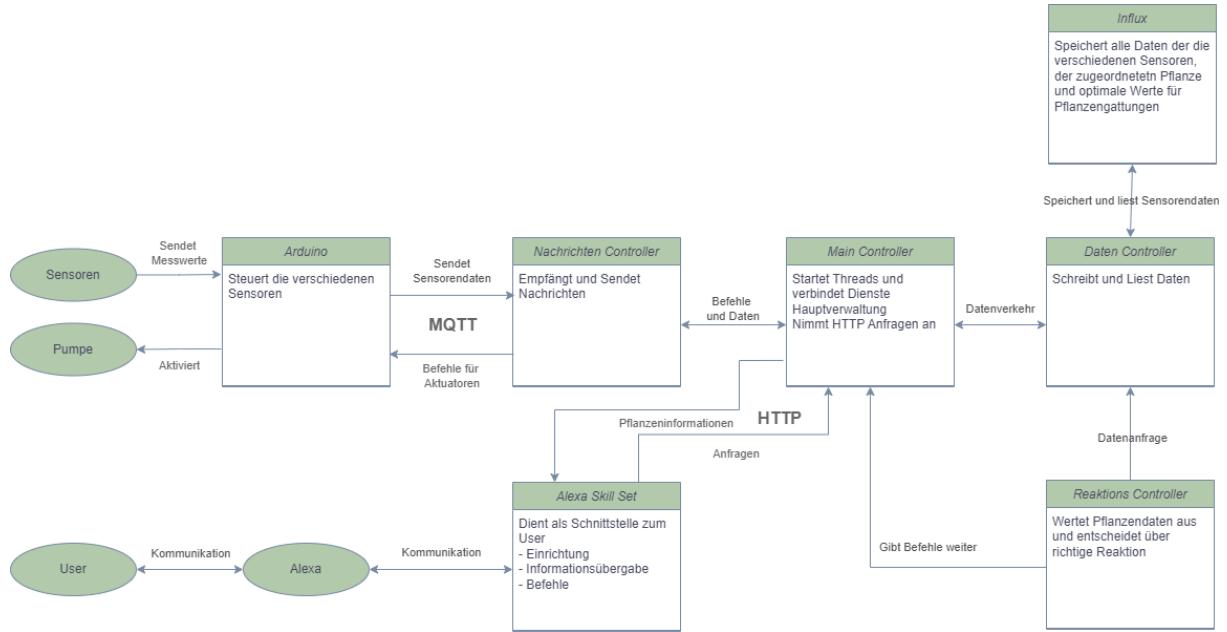


Abbildung 17: Flowchart der Software-Architektur

[11]

An den Arduino-Mikrocontroller sind verschiedene Sensoren angeschlossen. Dieser sendet hierbei Daten, und erhält Anweisungen, über einen MQTT-Kanal mit dem Main Controller, welche Aufgaben an andere Controller verteilt. Diese laufen auf einem Universitätserver. Hier werden Daten aufgenommen, gesendet, geschrieben, gelesen und ausgewertet.

## 4.2 I/O der Hardware

Der Arduino-Microcontroller besitzt zahlreiche Anschlüsse um verschiedenste IoT-Projekte zu unterstützen. Über ein Steckbrett wurden die verschiedenen Leitungen mit dem Microcontroller verbunden.

Die verschiedenen Sensoren benötigten hierfür jeweils Stromein- und ausgang. Bodenfeuchtigkeit und Lichtwerte wurden über analoge Eingänge aufgenommen. Der Temperatur- und Luftfeuchtigkeitssensor wurde über einen digitalen Eingang mithilfe einer I2C-Schnittstelle an den Microcontroller angebunden. Die Pumpe erhält das Signal zum Ein- und Ausschalten über einen digitalen Ausgang.

## 4.3 Mikrocontroller

Der Mikrocontroller muss folgende Aufgaben bewältigen können:

- Initialisierung der Sensoren

- Messung der Werte
- Umwandlung der Werte
- Senden von Daten
- Empfangen von Daten
- Einschalten der Pumpe

Diese Aufgaben werden in einer C++-Datei, mit einer zugehörigen yaml-Datei für verschiedene Parameter bewältigt.

Die verschiedenen Sensoren wurden an den Mikrocontroller angeschlossen und können so ausgelesen werden.

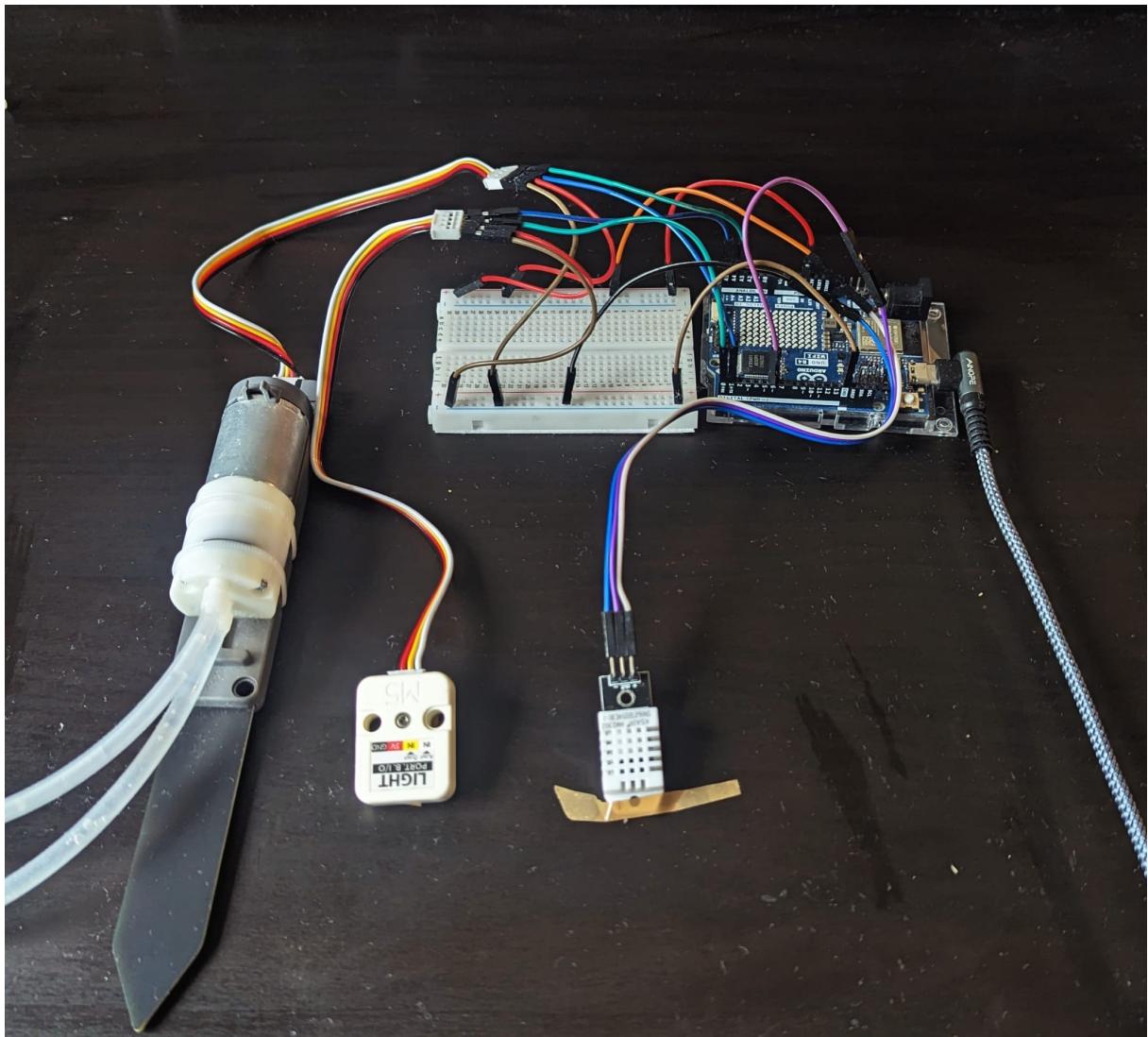


Abbildung 18: Mikrocontroller mit Sensoren, (v.l.n.r.) Pumpe Bodenfeuchtigkeit, Temperatur Luftfeuchtigkeit, Licht

[11]

#### **4.3.1 Sensormessung und Umwandlung der Daten**

Nach der Zuweisung der Anschlüsse des Boards, konnten die meisten Sensoren mittels eines analogen Inputs ausgelesen werden. Diese konnten so ohne weitere Installation oder spezifische Bibliotheken ausgelesen werden. Lediglich der Sensor für Temperatur und Luftfeuchtigkeit benötigt die spezifische Bibliothek "DHT". Dessen Werte wurden über den I2C Bus ausgelesen.

Die genaue Messung der Bodenfeuchtigkeit variiert geringfügig je nach Sensor, sie mussten daher zunächst kalibriert werden. Nach einigen Tests wurde herausgefunden dass für diesen speziellen Sensor 370 eine maximale und 795 eine minimale Bodenfeuchte bedeutet. So kann die Feuchtigkeit mittels einfacher Rechnung in Prozent ausgedrückt werden.

#### **4.3.2 Senden und Empfangen von Daten**

Die interne WLAN-Bibliothek "WiFiS3" wurde benutzt, um den Arduino mit dem Internet zu verbinden. Diese ist jedoch nicht völlig stabil, bei Netzwerkproblemen kann die Verbindung getrennt werden und wird nicht automatisch wieder aufgebaut. Um das Problem zu lösen wird vor dem Senden einer Nachricht der Status der WLAN-Verbindung nachgefragt. Besteht diese nicht, wird die Verbindung erneut hergestellt.

Mithilfe der Bibliothek "ArduinoMqttClient" kann zuverlässig eine MQTT Verbindung hergestellt werden. Hierbei wird auch ein Empfänger eingerichtet.

Über das zutreffende Topic wird der reine Datenwert (payload) des Sensors geschickt.

Erhält der Mikrocontroller eine Nachricht auf dem Topic zur Pumpenaktivierung wird diese Funktion gestartet.

#### **4.3.3 Einschalten der Pumpe**

Der Payload der MQTT-Nachricht zur Pumpenaktivierung erhält in bit-Form die zeitliche Dauer, wie lange die Pumpe pumpen soll. Nach dem Umwandeln in einen Integer, wird die Pumpe für so viele Sekunden aktiviert. Um Fehler und Manipulation vorzubeugen kann die Pumpe maximal zehn Sekunden lang am Stück pumpen.

#### **4.3.4 Zusammenfassung**

Der vollständige Versuchsaufbau sieht wie folgt aus:



Abbildung 19: Aufgebautes Gerät

[11]

#### 4.4 Main Controller

Der Main Controller dient als Steuerungseinheit. Er verwaltet die verschiedenen Tätigkeiten und verbindet die Dienste miteinander.

Der Standard Python-Logger wurde benutzt, um die Entwicklung, insbesondere in der Docker-Umgebung, zu vereinfachen und leichter nachzuvollziehen.

Parameter werden generell immer in externen yaml-Dateien gespeichert, und mit einer gesonderten Datei importiert. Hier wird überprüft, ob alle nötigen Parameter gespeichert sind und nur diese werden dann ausgewählt.

Der Main-Controller arbeitet hierbei in drei verschiedenen Threads. Einer für den Empfang von MQTT-Daten, einer für die regelmäßige Abfrage der gespeicherten Datenbank-Daten und einer zum Erhalten von HTTP-Anfragen des Sprachcontrollers. Diese werden über den Standard Python-Threader initialisiert.

Momentan werden alle Daten als Werte für eine Sukkulente gespeichert. Später soll die Datenbank selbst zuordnen welches Gerät, welche Pflanze enthält.

## 4.5 Nachrichten Controller

Der Nachrichten Controller verwaltet die Kommunikation über MQTT mit dem Mikrocontroller.

Mithilfe des MQTT-Clients kann eine stabile Verbindung zu einem MQTT-Kanal hergestellt werden, über den alle nötigen Nachrichten empfangen und gesendet werden können. Der Mikrocontroller schickt hierbei vier Nachrichten für Licht, Temperatur und Luft- und Bodenfeuchtigkeit. Diese werden getrennt voneinander verarbeitet.

## 4.6 Datenbankcontroller

### 4.6.1 Schreiben der Daten

Zunächst wird im die Influx-Instanz mit den richtigen Parametern initiiert. Erhält der Controller eine Nachricht wird das Topic (z.B. Temperatur) richtig zugeordnet. Stimmt das Topic nicht mit der internen Liste überein wird es nicht weiter verarbeitet. So können weitreichende Fehler vermieden werden.

Der Payload der Nachricht wird in einen Float umgewandelt und in die passende Tabelle geschrieben. Später wird für jedes Gerät eine eigene Tag-Nummer erstellt.

Ist dies der erste Eintrag in der Tabelle wird diese automatisch von Influx erstellt.

### 4.6.2 Lesen der Daten

Die No-SQL Anfrage für das Lesen der gespeicherten Daten sieht folgendermaßen aus:

```
<f"""
from(bucket: "{bucket}")
    |> range(start: -5m)
    |> filter(fn: (r) => r._ measurement == "{table_ name}")
    |> mean()""">
```

Der Durchschnitt der letzten fünf Minuten wird aus der entsprechenden Tabelle, beispielsweise Temperatur, entnommen. Nach einer Überprüfung, ob ein sinnvolles Ergebnis erzielt wurde, wird der Wert weitergegeben.

Plant Name	Succulent
Temperature min (C)	4
Temperature max (C)	27
Humidity min (rH)	40
Humidity max (rH)	50
Moisture min	33
Moisture max	45
Sun min (h)	6
Sun Category	direct
Notes	Grows the best if it has constant, strong light. Should lay a bit in the shade. Be careful of direct sun. Ideally they should get a lot of air circulation.

Tabelle 2: Tabelle für Sukkulanten

## 4.7 Reaktionscontroller

Der Reaktionscontroller benutzt simple Decision Trees, um die richtige Reaktion für derzeitige Umweltkonditionen zu finden.

Regelmäßig werden die Werte der Klasse PlantSensorData überschrieben. Die Klasse enthält die aktuellen Werte der Pflanze und die derzeitige Uhrzeit.

Die derzeitigen Werte werden aus der Datenbank gelesen und dem richtigen Attribut zugeordnet. Falls die Tabelle leer ist, wird explizit nichts zurückgegeben und eine Warnung ausgegeben. Dies ist unter anderem beim erstmaligen Starten der Fall, wenn noch keine Daten von den Sensoren erhalten wurden.

Die optimalen Werte für verschiedene Pflanzen sind in einer CSV-Tabelle gespeichert. Diese Werte wurden aus Erfahrungsberichten von Experten zusammengestellt und können jederzeit verändert werden. In dieser sind, der Pflanzengattung zugeordnet, die Werte gespeichert, in denen sich die reale Pflanze befinden sollte, um optimal zu wachsen. Bei einer späteren KI-Integrierung sollen diese Werte von dem System selbst und nicht von Experten angepasst werden.

Die Tabelle für die Sukkulanten sieht folgendermaßen aus:

”Moisture min“ und ”Moisture max“ sind in Prozent angegeben. ”Notes“ sind insbesondere für die Nutzenden gedacht, um bei der Pflanzenpflege zu helfen.

Um die Daten zu vergleichen sucht ein Algorithmus nach dem richtigen Eintrag in der Tabelle, und speichert diese in einer Klasse namens OptimalPlant.

Besonderer Fokus lag auf dem automatischen Bewässern der Pflanze. Der echte Wert der Bodenfeuchtigkeit wird mit den optimalen Werten verglichen. Liegt dieser darunter wird eine Funktion zur Aktivierung der Pumpe gestartet. Ein adäquates Ergebnis oder ein zu hoher Wert wird dem Nutzer mitgeteilt. Ähnlich gestalten sich die anderen Abfragen.

## 4.8 Sprachsteuerungs Controller

Die Implementierung eines Sprachsteuerungscontrollers mit Alexa Skills, erfordert einen neuen Skill in der Alexa Developer Console. Hier wird ein neues Interaktionsmodell kreiert. Intents sind hierbei die generellen Befehle, während Slots verschiedene Variablen repräsentieren.

Ein Interaktionsmodell sieht so wie folgt aus:

```
{
  "intents": [
    {
      "name": "PlantInfoIntent",
      "slots": [],
      "samples": [
        "give me information about my plant",
        "tell me about my plant",
        "what does my plant need",
        "how is my plant"
      ]
    },
    {
      "name": "AMAZON.HelpIntent",
      "samples": []
    },
    {
      "name": "AMAZON.CancelIntent",
      "samples": []
    },
    {
      "name": "AMAZON.StopIntent",
      "samples": []
    }
  ]
}
```

In diesem können verschiedene Sprachbefehle für einen Skill gelistet werden. Mit den Samples von PlantInfoIntent kann die normale Funktion des Skills aktiviert werden. die anderen Intents sind von Amazon vorgegeben, sie enthalten Phrasen wie "help", "how do I use this skill", "nevermindoder stop". Bei Bedarf können hier weitere Phrasen hinzugefügt werden.

Hiernach muss in der AWS Lambda Konsole der zugehörige Python-Code geschrieben werden, welcher das Backend durch eine HTTP Anfrage aufruft. Sagt der Nutzende nach dem Aktivieren des Skills eine der Phrasen auf, wird der zugehörige Python-Code aktiviert.

## 4.9 Testing

Zur Übersicht des Projektes wurde ein ”Featureßum Management der Bodenfeuchtigkeit erstellt. Dieses wurde in Cucumber geschrieben, um als Kommunikation zwischen Projektleiter und Entwickler zu stehen.

Das Feature sieht folgendermaßen aus:

Feature: Moisture Level Management

```
Scenario Outline: Moisture level <scenario_name>
    Given an intelligent vase which is ready to manage soil moisture
    And the vase holds a <plant_type>
    And we have a list of optimal environment for plants where we can search
    the matching plant <plant_type>
    And that plant requires an optimal moisture level between
    <min_optimal> and <max_optimal>
    When the plant has an average moisture level of <average_moisture>
    Then <expected_action> should happen
```

Examples:

scenario_name   min_optimal   max_optimal   average_moisture   expected_action
Below optimal   8   12   5   Activate Pump
Optimal   8   12   10   Do Nothing
Too high   8   12   15   Alert User

Nach der Implementation der richtigen SSteps” konnte dieser Test am Ende ohne Fehler durchlaufen werden.

Zusätzlich wurden mehrere Unit-Tests erstellt, um die Funktionsweise von Code-Einheiten zu testen, und sicherzustellen, dass verschiedenste Fälle abgedeckt sind. Hierzu gehören unter anderem die Reaktion auf verschiedene Arten von MQTT-Nachrichten, wie etwa das falsche Topic oder ein falscher Payload oder das Überprüfen einzelner Funktionen wie das Suchen einer bestimmten Pflanze in der CSV-Datei.

## 4.10 Deployment

Für das Deployment wurden GitHub und Docker benutzt.

GitHub diente als Versionskontrolle und als Testingtool. Bei einem push auf den main-branch wurden Workflows aktiviert, um Behave- und Unitests automatisch durchzuführen. Hierbei mussten verschiedene Konfigurationswerte in die Umweltvariablen von GitHub gespeichert werden.

Über DockerHubs werden neue Versionen des Projektes gepublished und auf den Universitätsserver gespielt.

Die Docker-cCmpose Datei installiert hierbei zwei verschiedene Services auf unterschiedlichen Ports, die Influx Datenbank und den Controller-Code. Die Lauffähigkeit von beiden wird mit einen Healthcheck garantiert. Die nötigen Konfigurationselemente erhält sie aus Sicherheitsgründen aus den Umgebungsvariablen des Systems. Zur Vereinfachung der Entwicklung werden hierbei alle bereits vorhandenen Einträge gelöscht, bei Bedarf können diese aber auch langfristig gespeichert werden.

## 5 Evaluation

In diesem Abschnitt wird die durchgeführte Arbeit überprüft. Hierbei wird bestimmt, ob die Anforderungen erfüllt wurden, wie sorgfältig der Code geschrieben wurde, wie geeignet die gewählten Technologien sind und inwiefern auf dem Projekt aufgebaut werden kann.

### 5.1 Anforderungen

Im Folgenden wird untersucht, ob die in Abschnitt 3 gestellten Anforderungen vom Projekt erfüllt werden.

**Automatischer Start und Sprachsteuerung** Das System verbindet sich nach dem Start automatisch mit den Diensten, es bedarf keinen weiteren Eingaben des Nutzenden. Um mit dem System zu interagieren benötigt der Nutzende lediglich den Sprachassistenten, über diesen kann er weitere Informationen erhalten und nach Hilfe fragen.

**Informationsabfrage in unter drei Sekunden** Die Reaktionszeiten des Systems sind sehr gut, von Anfrage bis Reaktion vergeht nur eine Sekunde:

```
2024-05-31 13:08:43 server-1    | root - INFO - Retrieving  
Data from InfluxDB  
2024-05-31 13:08:44 server-1    | root - INFO - Plant environment:  
Time: 2024-05-31  
11:08:44.570381,  
Temperature: 21.0,  
Light: 609.933333333333,  
Humidity: 57.66666666666664,
```

```
Moisture: 37.6
2024-05-31 13:08:44 server-1    | root - DEBUG - Searching for plant
                                'Succulent'.
2024-05-31 13:08:44 server-1    | root - INFO - Comparing sensor data
                                with optimal values
2024-05-31 13:08:44 server-1    | root - INFO - Soil Moisture
                                adequate.
```

**Senden der Daten jede Minute, Überprüfen der Daten alle fünf Minuten** Die Threads den Main Controllers und des Mikrocontrollers, sind so gestaltet, dass jede Minute die Sensorwerte gesendet werden, und alle fünf Minuten, die Werte der letzten fünf Minuten ausgewertet werden. Das System kann so schnell auf gefährliche Änderungen reagieren und wird nicht durch Ausreißer Messungen gestört.

**Ständige Verfügbarkeit** Durch verschiedene Sicherheitsvorkehrungen kann garantiert werden, dass das System, bei konstanter Stromzufuhr, nicht länger als eine Minute vom Netz verschwindet. Hiernach wird automatisch überprüft, ob alle Verbindungen bestehen.

**Zusammenfassung** Generell läuft das Programm ohne für den Nutzenden erkennbare Unterbrechungen, nimmt dabei neue Werte auf und wertet diese aus. Sowohl der Main-Controller als auch der Microcontroller, starten sich bei Fehlern von selber neu und stellen alle nötigen Verbindungen her.

Den Anschluss einer neuen Pflanze konnte noch nicht implementiert werden. Die Hinweise, um das Pflanzenwachstum zu verbessern, benötigen einen KI-Algorithmus, welcher die Umweltfaktoren gewichtet. Dies ist bisher nur in Planung.

Die Use-Cases wurden bis auf den Machine Learning Ansatz (3. Use Case) berücksichtigt. Das System kümmert sich ohne weitere Interferenz des Nutzers um die Pflanze und gießt sie automatisch. Bei Bedarf gibt es mehr Informationen preis, um den Nutzer bei der Pflege zu unterstützen. Dies ist jedoch optional, der Nutzer ist nicht gezwungen mit dem System zu interagieren.

Die gespeicherten Daten können in einer gehosteten Instanz angesehen werden. InfluxDB bietet ein intuitives Dashboard, in welchem die Daten visuell ansprechend dargestellt werden. Es können sowohl verschiedenste Queries gemacht werden, als auch Benutzer verwaltet und Daten verändert werden. Momentan werden aufgrund der geringen Menge an Daten diese nicht gelöscht.



Abbildung 20: Diagramm der Werte eines neuen Gerätes. (Licht wurde aus optischen Gründen entfernt) [24]

## 5.2 Umgesetzte Funktionen

Momentan verbindet sich ein neues Gerät automatisch mit dem System; weitere Angaben des Nutzenden sind nicht nötig. Jede Minute werden Sensorendaten gesendet, welche in einer Datenbank gespeichert werden. Alle 5 Minuten werden die Werte überprüft. Ein Logger stellt fest, wenn Werte nicht optimal sind und eine Pumpe wird eingeschaltet, falls der Boden der Pflanze zu trocken ist. Der Nutzende kann sich nach aktuellen Informationen über die Pflanze erkundigen.

Die Grundfunktionen sind hiermit umgesetzt, es bleiben jedoch noch viele weitere Funktionen offen, um ein für den Endkunden fertiges Produkt zu erstellen. Der Kunde sollte die Pflanzenart wählen können und mehrere Optionen haben, wie er mit dem Gerät interagieren möchte. Der Sprachassistent ist bisher lediglich rudimentär eingebunden, er könnte Warnungen geben, falls sich Werte außerhalb des optimalen Bereichs bewegen oder dem Nutzer verschiedene Möglichkeiten geben das System anzupassen: Beispielsweise die Pumpe auszustellen, Zeitabstände festzulegen, die Pflanzenart zu ändern oder neue optimale Werte eingeben.

Hierfür, und für die Einbindung der Machine Learning - Algorithmen müssen weitere Datenbanken erschaffen werden, welche die optimalen Werte einer Pflanzengattung für jede spezielle Pflanze anpassen kann. Momentan existiert nur eine statische CSV-Datei, Änderungen würden alle Pflanzen betreffen.

## 5.3 Qualität des Codes

Generell wäre Java für einen Großteil des Codes eine bessere Entscheidung gewesen.[59]

Durch den Fokus auf Klassen, wie etwa die optimale und echte Pflanze, wären klar gekennzeichnete Typen und Kompilierung von Vorteil gewesen. Java ist bei großen Projekten stabiler und Probleme bei der Kommunikation zwischen den Klassen sind schneller

ersichtlich. Auch ist Threading bei Python um einiges unstabiler, Multithreading wird überhaupt nicht unterstützt.

Zur Einschätzung des Codes wird das Lean-Modell, unterteilt in sieben Schritte, benutzt.[60]

- Verschwendungen minimieren: Regelmäßig wurde im Projekt nicht mehr genutzter Code gelöscht, anstatt ihn zu behalten oder zu kommentieren. Keine Funktion benutzt mehr Parameter als sie benötigt und gibt nur die für sie relevanten Werte zurück.
- Fokus auf Qualität: Funktionen wurden mithilfe von Cucumber erstellt, um Nutzen für den Kunden zu schaffen. Jede Funktion musste einen Mehrwert für den Kunden bringen, Ziel war es nicht ein theoretisches System zu erschaffen, sondern eines, welches in jedem Schritt den Kunden weiterbringt.
- Lerneffekte: Lerneffekte wurden durch die regelmäßige Absprache mit der Betreuerin verstärkt. Bei verschiedenen Technologien wurden die Vor- und Nachteile besprochen.
- Späte Festlegung: Vor der finalen Implementierung wurden die Technologien ausgiebig getestet, um ihre Grenzen und Möglichkeiten zu verstehen. Manche Sensoren hatten eingebaute Bibliotheken, um händische Rechnungen zu vermeiden. Ebenso bietet die NoSQL-Sprache von Influx viele Möglichkeiten in der Anfrage bereits die richtigen Ergebnisse zu erhalten, ohne aufwendige Rechnungen durchzuführen.
- Schnelle Lieferung: Es wurden kleine Iterationszeiten gewählt, um Fehler möglichst schnell zu erkennen und Gegenwert für den Kunden zu schaffen. Alle ein bis zwei Wochen wurde eine neue Version deployed. So war das System schon früh aktiv und lief ununterbrochen auf dem Server. Der Nutzen für den Kunden - die Pflanze am Leben zu halten - konnte so bereits früh erfüllt werden.
- Respekt vor Arbeitenden: Durch regelmäßige Rücksprachen mit dem Betreuen wurde sichergestellt, dass sich das Projekt jederzeit im Zeitrahmen befindet, und Ziele sinnvoll gesetzt werden. So konnte mit ausreichend Zeitpuffern gearbeitet werden.
- Optimierung: Durch den Fokus auf Cucumber und Unit Tests stand Optimierung von Anfang an im Vordergrund. Änderungen im Code und Fehler wurden so früh erkannt.

Generell wurden Klassen und Module möglichst klein gehalten. Ein Interface-Ansatz für die optimale und echte Pflanze erscheint im ersten Moment sinnvoll, bis auf ihre reale Repräsentation haben die beiden jedoch kaum etwas gemeinsam. Während die echte Pflanze auf eindeutigen Werten beruht, gibt die Optimalen lediglich Richtlinien vor. So haben die beiden Klassen keine übereinstimmenden Attribute und es wurde sich dafür entschieden zwei gänzlich voneinander getrennte Klassen zu benutzen.

## 5.4 Tauglichkeit des Mikrocontrollers und der Sensoren

Sowohl der Microcontroller als auch der Großteil der Sensoren sind für das Projekt geeignet.

Mithilfe der Arduino-Infrastruktur konnten externe Technologien leicht in das Projekt miteingebunden werden, der Arduino selbst wies nur Probleme bei der Internetverbindung auf. Für einen ersten Prototyp erwies sich der Microcontroller als ideal, wenn auch etwas zu leistungsfähig. Für spätere Builds reicht ein billigerer Microcontroller mit weniger Anschlüssen und einer geringeren Rechenfähigkeit aus. Der Microcontroller, ein Renesas RA4M1, arbeitet mit einer Taktung von 48MHz und 32kb Ram. Für einen nächsten Prototypen sollte der Controller zurückgestuft werden, beispielsweise für einen FA0E1, mit lediglich 32MHz. Dieser runtergeschraubte Controller sollte für die einfachen Berechnungen des Projektes ausreichend sein.

Die Sensoren blieben über den gesamten Einsatz stabil und gaben keine Fehler. Die Ergebnisse waren genau und konnten einfach übersetzt werden. Lediglich der Bodenfeuchtigkeitsmesser, welcher arbiträre Werte benutzt ist zu ungenau. Hier wäre ein anderer Sensor von Vorteil, dass alle Geräte die selben Werte produzieren. Ein geeichter Tensiometer ist hier sinnvoll, wie etwa der HYG EFS-10.[61]

## 5.5 Skalierung

Da bisher keine weiteren Geräte an das System angeschlossen wurden sind konnte das System nicht exakt getestet werden. Jedoch konnten mit einem Dummy-Sender Daten über MQTT geschickt werden, um zu testen wie sich die Technologien bei einem Vielfachen der Daten verhalten.

Durchschnittlich benötigt das Senden einer MQTT-Nachricht mit einem Dummy-Sender etwa 0.07 Sekunden. Dies ergibt etwa 850 Nachrichten pro Minute.

```
Published 83 to light_ard in 0.066460 seconds
Published 99 to moisture_ard in 0.081350 seconds
Published 74 to humidity_ard in 0.084083 seconds
Published 44 to temperature_ard in 0.078810 seconds
```

Der Sender konnte zwanzigmal gestartet werden, ohne dass Probleme beim Main-Controller entstehen. Ein echtes Gerät sendet 48 Nachrichten pro Minute. Es können mit diesen Technologien jetzt bereits etwa 350 Geräte verbunden werden. Beim Öffnen von weiteren Sendern kam es zu Problemen auf der Sendereinheit, nicht beim Controller.

Die Daten konnten von Influx ohne Komplikationen verarbeitet werden.

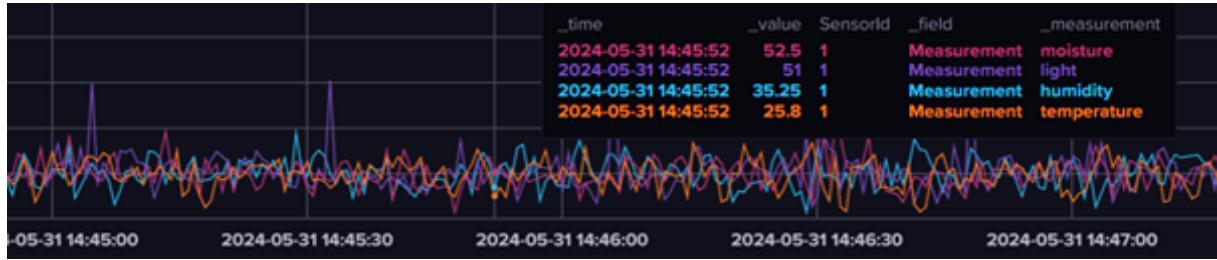


Abbildung 21: Diagramm der zufallsgenerierten Werte der Dummy-Sender. [24]

Anhand dieser Daten kann man davon ausgehen, dass sowohl MQTT, als auch InfluxDB imstande sind das System sinnvoll zu skalieren und für den Produktionsbetrieb eingesetzt werden können.

## 6 Fazit

### 6.1 Zusammenfassung

In dieser Arbeit wurden verschiedene, bereits implementierte Projekte, welche einen intelligenten Ansatz zur Pflanzenpflege verfolgten gesichtet. Relevante Informationen für eine an Konsumenten gerichtete intelligente Vase konnten zusammengefasst und auf das Projekt übertragen werden.

Aus verschiedenen Erfahrungsberichten konnten so die Anforderungen an das Projekt gestellt und verschiedene Use-Cases erstellt werden. Die intelligente Vase soll hiermit sowohl einen Mehrwert für "hands-off" Konsumenten herstellen, welche die Pflanzenpflege komplett dem System übergeben, als auch Enthusiasten, welche mehr Informationen und Tipps zur Pflege erhalten möchten.

Die Steuerung mithilfe eines Sprachassistenten macht hierbei die Nutzung des Systems intuitiver und zugänglicher, ohne eine weitere Quelle der Ablenkung für den Nutzer darzustellen.

Das Projekt konnte in großen Teilen den Anforderungen entsprechend umgesetzt werden. Die verschiedenen Technologien wurden sorgfältig ausgewählt, um den Ansprüchen gerecht zu werden und einen ersten, funktionierenden Prototypen zu erstellen. Dieser deckt bereits zwei wichtige Use-Cases ab, er sammelt über die Sensoren Informationen über die Pflanze und gießt diese bei Bedarf automatisch. Die Funktionsweise des Codes wird durch verschiedene Tests automatisch überprüft.

Mithilfe einer Auswertung der Ergebnisse wurde sowohl die Auswahl der Technologien, als auch die Codequalität und die verschiedenen bereits implementierten Funktionen überprüft. So kann der Prototyp in weiteren Iterationen verbessert und ausgebaut werden.

Insgesamt wurde ein fehlerfreier, konstant laufender Prototyp eines Systems geschaffen, welcher automatisch Umweltwerte einer Pflanze aufnimmt und die Daten speichert

und verarbeitet, um den Nutzenden die Pflanzenpflege zu erleichtern. Dieser kommuniziert mit dem Gerät ausschließlich über einen Sprachassistenten, die Interaktion wird auf ein Minimum beschränkt. Es werden Temperatur, Sonnenlicht und Luft-/ und Bodenfeuchtigkeit gemessen, der Nutzende über für die Pflanze ungünstige Werte hingewiesen und die Pflanze automatisch gegossen. Es ist jederzeit möglich, die Werte für die optimalen Umweltbedingungen der Pflanze anzupassen, das System lässt sich so dynamisch verändern.

Außerdem ist die Softwarearchitektur offen gehalten, um jederzeit weitere Sensoren, Aktuatoren oder Informationen über die Pflanze hinzuzufügen.

Es wurde die nötige Vorarbeit geleistet, um in den nächsten Schritten verschiedene KI-Technologien zu integrieren. Diese sollen die Umweltbedingungen automatisch anpassen, um das Wachstum der Pflanze zu optimieren.

## 6.2 Ausblick

Während die Sensoren auch auf lange Sicht gute Ergebnisse erzielen ist der Mikrocontroller für eine spätere Revision zu teuer. Hier sind einfache Mikrocontroller mit weniger Rechenleistung, Anschlägen und Speicherplatzverbrauch durch die Arduino Infrastruktur sinnvoller. Weitere Aktuatoren wie Lampen, ein Sprüher für Wasser, oder eine schließbare Haube, um die Luftfeuchtigkeit zu regulieren, könnten sinnvolle Wege sein, das System weiter zu automatisieren und das Pflanzenwachstum noch besser zu unterstützen.

MQTT und Influx erwiesen sich auch für größere Datenmengen als zuverlässig und können in der Produktionsphase weiterhin verwendet werden.

Der Code sollte für spätere Iterationen weiter aufgeteilt werden und unter anderem Java benutzen, um stabiler zu laufen und vor allem Multithreading zu ermöglichen. Auch können noch weitere Funktionen eingebaut werden, um dem Nutzenden mehr Möglichkeiten zu geben mit dem System zu interagieren.

Im Anschluss sollte der Anschluss einer Kamera erfolgen. Mithilfe eines Object-Detections-Algorithmus können die Bilder benutzt werden, um das Wachstum der Pflanze einzuschätzen. Auch können die Erkenntnisse des Machine-Learnings Abschnittes implementiert werden, um das Wissen auszubauen, welche Umweltparameter besonders wichtig für das Pflanzenwachstum sind. Durch kleine Veränderungen dieser Werte kann so das optimale Wachstum begünstigt werden.

Auf den Erfahrungen des Prototyps kann nun der Bau eines kohärenten Gerätes erfolgen, welcher die Sensoren und den Microcontroller sinnvoll in einem Bau unterbringt, in den eine Vase gestellt werden kann.

# Abbildungsverzeichnis

1	Ein Hydroponisches Anbausystem . . . . .	3
2	Eine Smart Vase mit Display . . . . .	4
3	Diagramm des Herstellers Rainpoint . . . . .	5
4	Diagramm der AI-Einbindung in der Champignon Aufzucht . . . . .	7
5	Bildverarbeitung anhand der Fourier Analyse . . . . .	8
6	Prozess der Entscheidungsfindung[18] . . . . .	9
7	Regelwerk für eine Spannungsausgabe. [24] . . . . .	12
8	Flowchart der allgemeinen Software-Architektur . . . . .	20
9	Flowchart des Mikrocontrollers . . . . .	21
10	Flowchart des Main Controllers . . . . .	22
11	Lichtsensor . . . . .	23
12	Temperatur- und Luftfeuchtigkeitssensor . . . . .	24
13	Pumpe und Bodenfeuchtigkeitssensor . . . . .	25
14	Kamera . . . . .	26
15	Messung einer Pflanze mittels Object Detection . . . . .	35
16	Genereller Workflow des Q-Learning Algorithmus . . . . .	36
17	Flowchart der Software-Architektur . . . . .	37
18	Mikrocontroller mit Sensoren, (v.l.n.r.) Pumpe Bodenfeuchtigkeit, Temperatur Luftfeuchtigkeit, Licht . . . . .	38
19	Aufgebautes Gerät . . . . .	40
20	Diagramm der Werte eines neuen Gerätes. (Licht wurde aus optischen Gründen entfernt) [24] . . . . .	47
21	Diagramm der zufallsgenerierten Werte der Dummy-Sender. [24] . . . . .	50

## **Tabellenverzeichnis**

1	Einteilung der Intelligenten Systeme . . . . .	3
2	Tabelle für Sukkulanten . . . . .	42

## Quellcodeverzeichnis

1	Arduino Controller, C . . . . .	60
2	Haupt Controller, Python . . . . .	63
3	Plant Sensor Data, Python . . . . .	65
4	Optimal Plants, Python . . . . .	66
5	Response Handler für Bodenfeuchtigkeit, Python . . . . .	67
6	Vergleich der optimalen und reelen Werte der Bodenfeuchtigkeit . . . . .	68
7	Docker Compose, Go . . . . .	70
8	Cucumber Test, Python . . . . .	72

# Anhang

Im Folgenden wird der Code des Projektes präsentiert. Zusätzlich ist ein Video angehängt, wie die Pflanze gegossen wird, und wie mit Alexa interagiert wird.

```
#include <ArduinoMqttClient.h>
#include <WiFiS3.h>
#include "DHT.h"

#include "arduino_secrets/arduino_secrets.h"

//WiFi Variables
const char ssid[] = SECRET_SSID;
const char pass[] = SECRET_PASS;

//MQTT Variables
const char* mqtt_broker = MQTT_BROKER;
const int mqtt_port = MQTT_PORT;

//MQTT Topics
const char* light_topic = LIGHT_DATA_TOPIC;
const char* moisture_topic = MOISTURE_DATA_TOPIC;
const char* temperature_topic = TEMPERATURE_DATA_TOPIC;
const char* humidity_topic = HUMIDITY_DATA_TOPIC;
const char* waterpump_activate_topic = WATERPUMP_ACTIVATE_TOPIC;
const char* waterpump_error_topic = WATERPUMP_ERROR_TOPIC;

//How often Sensor Data is sent
const long interval = INTERVAL_SEND;

//Arduino Board Configuration
const int LightDigital_Input = LIGHT_DIGITAL_INPUT;
const int LightAnalog_Input = LIGHT_ANALOG_INPUT;
const int WaterpumpDigital_Output = WATERPUMP_DIGITAL_OUTPUT;
const int MoistureAnalog_Input = MOISTURE_ANALOG_INPUT;
const int DHTPIN = TEMPERATURE_HUMIDITY_DIGITAL_INPUT;

//Temperature and Humidity Sensor
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
```

```

//Restart Timer
const unsigned long validation_interval = VALIDATION_INTERVAL;
unsigned long start_time;

//WiFi and MQTT Client
WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);

void setup() {
    //Initialize serial and wait for port to open:
    Serial.begin(9600);

    while (!Serial)
        delay(10);

    setupWifi();
    setupMQTT();
    //setup MQTT Message receiver
    setupMQTTReceiver();

    setupSensors();
    start_time = millis();
}

void loop() {
    mqttClient.poll();
    readAndSendMeasurements();
    Serial.println();

    Serial.print("Time left to validate: ");
    Serial.print(validation_interval - (millis() - start_time));
    Serial.println(" ms.");
    if(millis() - start_time >= validation_interval) {
        validateConnection();
    }
    delay(interval);
}

```

```

void setupSensors() {
    pinMode(LightDigital_Input, INPUT);
    pinMode(LightAnalog_Input, INPUT);
    pinMode(WaterpumpDigital_Output, OUTPUT);
    pinMode(MoistureAnalog_Input, INPUT);
    dht.begin();
}

void setupWifi() {
    while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
        // failed, retry
        Serial.print("Connection to WPA SSID ");
        Serial.print(ssid);
        Serial.println(" failed. Retrying... ");
        delay(2000);
    }
    Serial.println("You're connected to the network");
    Serial.println();
}

void setupMQTT() {
    Serial.print("Attempting to connect to the MQTT broker: ");
    Serial.println(mqtt_broker);

    while (!mqttClient.connect(mqtt_broker, mqtt_port)) {
        // MQTT connection failed, retry
        Serial.print("MQTT connection failed! Error code = ");
        Serial.print(mqttClient.connectError());
        Serial.println(". Retrying... ");
        delay(2000);
    }

    Serial.println("You're connected to the MQTT broker!");
    Serial.println();
}

void setupMQTTReceiver() {
    mqttClient.onMessage(onMqttMessage);
    Serial.print("Subscribing to topic: ");
}

```

```

Serial.println(waterpump_activate_topic);
mqttClient.subscribe(waterpump_activate_topic);
}

void readAndSendMeasurements() {
    int light_value = analogRead(LightAnalog_Input);
    float temperature_value = dht.readTemperature();
    float humidity_value = dht.readHumidity();
    float moisture_value = analogRead(MoistureAnalog_Input);

    moisture_value = calculateMoisture(moisture_value);

    Serial.print("Light Value: ");
    Serial.println(light_value);
    sendDataOverMQTT(light_value, light_topic);

    Serial.print("Moisture Value: (in %): ");
    Serial.println(moisture_value);
    sendDataOverMQTT(moisture_value, moisture_topic);

    Serial.print("Temperature Value(in °C): ");
    Serial.println(temperature_value);
    sendDataOverMQTT(temperature_value, temperature_topic);

    Serial.print("Humidity Value(in %rH): ");
    Serial.println(humidity_value);
    sendDataOverMQTT(humidity_value, humidity_topic);
}

//Max-Wert: 795
//Min-Wert: 370
float calculateMoisture(float moisture_value) {
    float moisture_percent = moisture_value - 370;
    moisture_percent = moisture_percent / 425 * 100;
    moisture_percent = 100 - moisture_percent;
    return moisture_percent;
}

void sendDataOverMQTT(int data, const char* topic) {

```

```

Serial.print("Sending data to topic: ");
Serial.println(topic);

//send message, the Print interface can be used to set the message
→ contents
mqttClient.beginMessage(topic);
mqttClient.print(data);
mqttClient.endMessage();
}

void onMqttMessage(int messageSize) {
    Serial.println();
    Serial.print("Received a message with topic ");
    Serial.println(mqttClient.messageTopic());
    const char* received_topic = mqttClient.messageTopic().c_str();

    // Allocate a buffer to hold the incoming message
    char messageBuffer[messageSize + 1]; // +1 for null terminator
    memset(messageBuffer, 0, sizeof(messageBuffer)); // Clear the buffer

    // Read the message into the buffer
    int bytesRead = 0;
    while (mqttClient.available() && bytesRead < messageSize) {
        char c = mqttClient.read();
        messageBuffer[bytesRead] = c;
        bytesRead++;
    }

    // Null-terminate the buffer
    messageBuffer[bytesRead] = '\0';

    // Convert the message buffer to an integer
    int received_value = atoi(messageBuffer);

    // Now 'receivedValue' contains the integer value from the MQTT
    → message
    Serial.print("Received value: ");
    Serial.println(received_value);
}

```

```

if (strcmp(received_topic, waterpump_activate_topic) == 0) {
    Serial.println("Activating Waterpump.");
    activate_pump(&received_value);
}

void activate_pump(int* waterpump_activation_time) {
    if (*waterpump_activation_time > 10) {
        *waterpump_activation_time = 10;
    }
    Serial.print("Waterpump pumping for ");
    Serial.print(*waterpump_activation_time);
    Serial.println(" seconds.");
    digitalWrite(WaterpumpDigital_Output, HIGH);
    delay(*waterpump_activation_time * 1000);
    digitalWrite(WaterpumpDigital_Output, LOW);
}

void validateConnection() {
    if (WiFi.status() != WL_CONNECTED) {
        Serial.println("Connection to WiFi lost. Reconnecting...");
        setupWifi();
    }
    if (!mqttClient.connected()) {
        Serial.println("MQTT Client disconnected. Reconnecting...");
        setupMQTT();
        setupMQTTReceiver();
    }
    start_time = millis();
}

```

Quellcode 1: Arduino Controller, C

```

import threading
import logging
from flask import Flask, jsonify
from influxdb_client.client.write_api import SYNCHRONOUS
from logic.mqtt_package.mqtt_client import MQTTClient
from logic.response_package.response_handler import ResponseHandler
from logic.database_package.database_handler import DatabaseHandler
from logic.plants_config.optimal_plants.optimal_plant_environment import
    ↳ OptimalPlant
from logic.plants_config.plant_sensor_data.plant_sensor_data import
    ↳ PlantSensorData

# Logging Configuration
logging_level=logging.DEBUG
main_logger = logging.getLogger()
main_logger.setLevel(logging_level)

# Setup Stream Handler
stream_handler = logging.StreamHandler()
stream_handler.setLevel(logging_level)
formatter = logging.Formatter("%(name)s - %(levelname)s - %(message)s")
stream_handler.setFormatter(formatter)
main_logger.addHandler(stream_handler)

# Flask app initialization
app = Flask(__name__)

# TODO database of plants with the id of the arduino
plant_name = "Succulent"

def calculation_thread(database_handler, response_handler):
    while True:
        try:
            optimal_plants = OptimalPlant.load_from_csv()
            real_plant =
                ↳ PlantSensorData.get_current_values(database_handler)
            response_handler.judge_environemnt(optimal_plants,
                ↳ real_plant, plant_name)

```

```

        threading.Event().wait(60)

    except ValueError as ve:
        logging.error(f"Error : {ve}")
        threading.Event().wait(60)
    except Exception as e:
        logging.error(f"An unexpected error occurred: {e}")
        threading.Event().wait(360)

@app.route('/plant_info', methods=['GET'])
def get_plant_info():
    database_handler = DatabaseHandler()
    real_plant = PlantSensorData.get_current_values(database_handler)

    temperature = str(real_plant.temperature)
    soil_moisture = str(real_plant.humidity)
    humidity = str(real_plant.light)
    sunlight = str(real_plant.moisture)

    plant_info = {
        "temperature": f"{temperature} degree Celsius",
        "soil_moisture": f"{soil_moisture} percent",
        "humidity": f"{humidity} percent",
        "sunlight": f"{sunlight} lumen"
    }

    return jsonify(plant_info)

if __name__ == "__main__":
    # Create an instance of the MQTTSsubscriber class
    database_handler = DatabaseHandler()
    mqtt_client = MQTTClient(database_handler)
    response_handler = ResponseHandler(mqtt_client)

    # MQTT Subscriber Thread
    subscriber_thread = threading.Thread(target=mqtt_client.subscribe)
    subscriber_thread.start()

    # Database Handler Thread

```

```

database_handler_thread =
    → threading.Thread(target=calculation_thread,
    → args=(database_handler, response_handler))
database_handler_thread.start()

# Flask App Thread
flask_thread = threading.Thread(target=lambda:
    → app.run(host='0.0.0.0', port=5000))
flask_thread.start()

try:
    while True:
        # We can perform other tasks here while the client is
        → subscribed
        pass
except KeyboardInterrupt:
    mqtt_client.stop()
    database_handler_thread.join()
    flask_thread.join()

```

Quellcode 2: Haupt Controller, Python

```

from datetime import datetime

class PlantSensorData:
    def __init__(self, temperature, light, humidity, moisture,
                 current_time):
        self.temperature = temperature
        self.humidity = humidity
        self.light = light
        self.moisture = moisture
        self.current_time = current_time

    @classmethod
    def get_current_values(cls, database_handler):
        try:
            tables = database_handler.retrieve_data()
            if tables is not None:
                temperature = None
                light = None
                humidity = None
                moisture = None
                # Parse the data from tables
                for table in tables:
                    if table is not None:
                        for record in table.records:
                            if record is not None:
                                measurement = record.get_measurement()
                                field_value = record.get_value()
                                logging.debug(f"Measurement:
                                  {measurement}")
                                logging.debug(f"Field Value:
                                  {field_value}")

                                if measurement == 'temperature':
                                    temperature = field_value
                                elif measurement == 'light':
                                    light = field_value
                                elif measurement == 'humidity':
                                    humidity = field_value

```

```

        elif measurement == 'moisture':
            moisture = field_value

    current_time = datetime.now()
    # Create an instance of PlantSensorData
    data = cls(temperature, light, humidity, moisture,
               current_time)
    logging.info(f"Plant environment: Time:
                  {data.current_time}, Temperature:
                  {data.temperature}, Light: {data.light}, Humidity:
                  {data.humidity}, Moisture: {data.moisture}")
    return data

else:
    logging.warning("Tables are empty")
    return None

except Exception as e:
    logging.error(f"An unexpected error occurred: {str(e)}")
    return None

```

Quellcode 3: Plant Sensor Data, Python

```

import os

from .optimal_plants_list_functions import get_optimal_plants_as_list

dir_path = os.path.dirname(__file__)
csv_file_path = os.path.join(dir_path, "optimal_plants.csv")

class OptimalPlant:
    def __init__(self, plant, temperature_min, temperature_max,
                 humidity_min, humidity_max, moisture_min, moisture_max,
                 pause_for_watering, sun_min, sun_category, notes=None):
        self.plant = plant
        self.temperature_min = int(temperature_min)
        self.temperature_max = int(temperature_max)
        self.humidity_min = int(humidity_min)
        self.humidity_max = int(humidity_max)
        self.moisture_min = int(moisture_min)
        self.moisture_max = int(moisture_max)
        self.pause_for_watering = int(pause_for_watering)
        self.sun_min = int(sun_min)
        self.sun_category = sun_category
        self.notes = notes

    @classmethod
    def from_csv_row(cls, csv_row):
        return cls(*csv_row)

    @classmethod
    def load_from_csv(cls, file_path=csv_file_path):
        plants_data = get_optimal_plants_as_list(file_path)
        plants = [cls.from_csv_row(row) for row in plants_data]
        return plants

```

Quellcode 4: Optimal Plants, Python

```
class ResponseHandler:  
    def __init__(self, mqtt_client):  
        self.mqtt_client = mqtt_client  
    def judge_environemnt(self, optimal_plants, real_plant, plant_name):  
        matching_plant = find_optimal_plant_by_name(optimal_plants,  
            → plant_name)  
  
        logging.info("Comparing sensor data with optimal values")  
        moisture_response(matching_plant, real_plant, self.mqtt_client)
```

Quellcode 5: Response Handler für Bodenfeuchtigkeit, Python

```

activation_time_pump = 3

def moisture_response(matching_plant, real_plant, mqtt_client):
    actual_moisture = real_plant.moisture
    moisture_min = matching_plant.moisture_min
    moisture_max = matching_plant.moisture_max

    if actual_moisture > moisture_max:
        logging.info(f"Soil Moisture {actual_moisture} % is higher than
                     ↳ recommended value={moisture_max}. Please put the plant
                     ↳ inside or take it into the sun for a short time to dry.")
    elif actual_moisture < moisture_min:
        logging.info(f"Soil Moisture {actual_moisture} % is lower than
                     ↳ recommended value={moisture_min}. Activating waterpump.")
        mqtt_client.activate_pump(activation_time_pump)
    else:
        logging.info("Soil Moisture adequate.")

```

Quellcode 6: Vergleich der optimalen und reelen Werte der Bodenfeuchtigkeit

```

version: "3"
services:
  influxdb:
    image: influxdb:latest
    ports:
      - "8086:8086"
    volumes:
      - influxdb:/var/lib/influxdb
    environment:
      - DOCKER_INFLUXDB_INIT_MODE=setup
      - DOCKER_INFLUXDB_INIT_USERNAME=root
      - DOCKER_INFLUXDB_INIT_PASSWORD=${DOCKER_INFLUXDB_INIT_PASSWORD}
      - DOCKER_INFLUXDB_INIT_ORG=my_org
      - DOCKER_INFLUXDB_INIT_BUCKET=my_bucket
      - DOCKER_INFLUXDB_INIT_RETENTION=1w
      - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=${INFLUX_TOKEN}
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8086/ping"]
      interval: 10s
      timeout: 5s
      retries: 5

  server:
    image: tyjga/intelligent_vase:v0.5.4
    ports:
      - 8000:8000
    depends_on:
      influxdb:
        condition: service_healthy
    environment:
      - INFLUX_TOKEN=${INFLUX_TOKEN}
      - INFLUX_URL=${INFLUX_URL_DOCKER}
      - MQTT_BROKER_ADDRESS=${MQTT_BROKER_ADDRESS}
      - MQTT_BROKER_PORT=${MQTT_BROKER_PORT}
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8000"]
      interval: 10s
      timeout: 5s

```

```
    retries: 5

volumes:
  influxdb: null
```

Quellcode 7: Docker Compose, Go

```

from behave import given, when, then

import io
from contextlib import redirect_stdout

from datetime import datetime

from logic.plants_config.plant_sensor_data.plant_sensor_data import
    PlantSensorData
from logic.database_package.database_handler import DatabaseHandler
from logic.mqtt_package.mqtt_client import MQTTClient
from logic.plants_config.optimal_plants.optimal_plant_environment import
    OptimalPlant
from logic.plants_config.optimal_plants.optimal_plants_list_functions
    import find_optimal_plant_by_name
from logic.response_package.response_functions import moisture_response

@given("an intelligent vase which is ready to manage soil moisture")
def step_given_intelligent_vase(context):
    temperature_value = 25.0
    light_value = 500
    humidity_value = 60.0
    moisture_value = 35
    current_time_value = datetime.now()

    context.plant = PlantSensorData(temperature_value, light_value,
                                    humidity_value, moisture_value, current_time_value)

@given("the vase holds a {plant_type}")
def step_given_vase_holds_plant(context, plant_type):
    context.plant_type = plant_type

@given("we have a list of optimal environment for plants where we can
    search the matching plant {plant_type}")
def step_given_plant_requires_optimal_level(context, plant_type):
    optimal_plants = OptimalPlant.load_from_csv()
    context.matching_plant = find_optimal_plant_by_name(optimal_plants,
                                                       plant_type)

```

```

@given("that plant requires an optimal moisture level between
      ↳ {min_optimal_level:d} and {max_optimal_level:d}")
def step_given_plant_requires_optimal_level(context, min_optimal_level,
                                             max_optimal_level):
    context.matching_plant.moisture_min = min_optimal_level
    context.matching_plant.moisture_max = max_optimal_level

@when("the plant has an average moisture level of
      ↳ {average_moisture_level}")
def step_at_certain_average_moisture(context, average_moisture_level):
    context.plant.moisture = float(average_moisture_level)

@then("{expected_action} should happen")
def step_then_expected_action(context, expected_action):
    database_handler = DatabaseHandler()
    context.mqtt_client = MQTTClient(context)

    captured_output = io.StringIO()
    with redirect_stdout(captured_output):
        moisture_response(context.matching_plant, context.plant,
                           ↳ context.mqtt_client)
    print_output = captured_output.getvalue()

    print(f>this is the printed output: {print_output}")

    if expected_action.lower() == "alert user":
        assert print_output.lower(), f"Soil Moisture
          ↳ {context.plant.moisture}% is too high. Please put the plant
          ↳ inside or take it into the sun for a short time to dry."
    elif expected_action.lower() == "activate pump":
        assert print_output.lower(), f"Soil Moisture
          ↳ {context.plant.moisture}% is too low. Activating waterpump."
    elif expected_action.lower() == "do nothing":
        assert print_output.lower(), f"Soil Moisture adequate."

```

Quellcode 8: Cucumber Test, Python

# Literatur

- [1] Automate Team. „From Soil to Tech: The Impact of Automation in Agriculture.“ Accessed on May 28, 2024, A3 Automate. (2023), Adresse: <https://www.automateshow.com/blog/from-soil-to-tech-the-impact-of-automation-in-agriculture>.
- [2] A. Narishkin und M. Tejapaibul. „Americans spend billions on houseplants, only to kill half of them.“ Accessed on May 28, 2024, Insider Inc. (2022), Adresse: <https://www.businessinsider.com/houseplant-industry-americans-billions-die-2022-3>.
- [3] „Every year, more than 16 million plants at home, die in the Netherlands.“ Accessed on May 28, 2024, Holland Times. (2024), Adresse: <https://www.hollandtimes.nl/articles/tipsandreviews/every-year-more-than-16-million-plants-at-home-die-in-the-netherlands/>.
- [4] ZPYXBH, *ZPYXBH Smart Indoor Garden Kräutergarten Indoor Hydroponic System Mit LED Licht Hydroponisches Anzuchtsystem Grow Home Gewächshaus Kit Für Zimmergewächshaus Automatischer Timer Bewässerungssystem*, 2023. Adresse: <https://shorturl.at/dJ4tx>.
- [5] Emsa, *Emsa M5261700 Click & Grow Smart Garden 3 Indoor-Garten, passend für 3 Kräuterkapseln*, weiß, 2023. Adresse: <https://shorturl.at/AFLV>.
- [6] Geberioz, *Geberioz Indoor Garten in Braun - Hydroponisches Anzuchtsystem mit Fernbedienung - 130W LED Pflanzenlampe mit 48 Pods - Gewächshaus Kräutergarten (Braun)*, 2023. Adresse: <https://shorturl.at/iYr1h>.
- [7] Pflanzenfabrik. „Das Tropfsystem der Hydroponik.“ (2023), Adresse: <https://pflanzenfabrik.de/hydroponik-tropfssystem/>.
- [8] V. Muller, *Lua, the smart planter with feelings*, Visit the IndieGoGo Page, 2019. Adresse: <https://www.indiegogo.com/projects/lua-the-smart-planter-with-feelings#/>.
- [9] ZPYXBH, *LYEVA Ivy Smart Planter, Smart Planter Robot, Smart Pet Planter, lustiger interaktiver Blumentopf, ordentliches Schreibtisch-Setup-Geschenk, für Innendekoration (Purple)*, Visit the Amazon Product Page, 2023. Adresse: <https://shorturl.at/FxqMA>.
- [10] L. Lawton, „Taken by the Tamagotchi: How a toy changed the perspective on mobile technology,“ *The iJournal: Student Journal of the University of Toronto's Faculty of Information*, Jg. 2, Nr. 2, 2017.

- [11] RainPoint Irrigation, *RainPoint Smart WiFi Sprinkler Timer*, RainPoint Irrigation, Visit the RainPoint Product Page, 2024. Adresse: <https://www.rainpointonline.com/collections/wi-fi-hose-timers/products/smart-watering-timer-with-wifi-hub>.
- [12] Zukunftsstiftung Landwirtschaft. „Bäuerliche und industrielle Landwirtschaft.“ Zukunftsstiftung Landwirtschaft Büro Berlin (Projektleitung), Marienstr. 19-20, D-10117 Berlin, Deutschland, Tel: +49-(0)30-284 82 320, Fax: +49-(0)30-284 82 329, Zukunftsstiftung Landwirtschaft. (Year of Access), Adresse: <https://t.ly/e40G9>.
- [13] European Commission. „Research and innovation.“ European Commission, Directorate General for Agriculture and Rural Development, European Commission. (Year of Access), Adresse: [https://agriculture.ec.europa.eu/sustainability/research-and-innovation\\_en](https://agriculture.ec.europa.eu/sustainability/research-and-innovation_en).
- [14] D. S. Kumar, V. Haritha, C. Pravalika, V. U. Kumar und P. Vijay, *Automatic Irrigation System Using IOT*, Preprint, 2018. Adresse: <https://www.ijarst.in/public/uploads/paper/476341704899923.pdf>.
- [15] Z. Mushtaq, S. S. Sani, K. Hamed u.a., „Automatic agricultural land irrigation system by fuzzy logic,“ in *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*, IEEE, 2016, S. 871–875. Adresse: <https://ieeexplore.ieee.org/abstract/document/7726287>.
- [16] S. Seesansui, T. Phurahong und J. Sarasook, „The Automatic Temperature Control for Agricultural Plant House,“ in *2020 International Conference on Power, Energy and Innovations (ICPEI)*, Chiangmai, Thailand, 2020, S. 37–40. doi: 10.1109/ICPEI49860.2020.9431413.
- [17] S. N. S. Al-Humairi, P. Manimaran, M. I. Abdullah und J. Daud, „A smart automated greenhouse: Soil moisture, temperature monitoring and automatic water supply system (peaty, loam and silty),“ in *2019 IEEE Conference on Sustainable Utilization and Development in Engineering and Technologies (CSUDET)*, IEEE, 2019, S. 111–115.
- [18] R. Barauskas, A. Kriščiūnas, D. Čalnerytė u.a., „Approach of AI-Based Automatic Climate Control in White Button Mushroom Growing Hall,“ *Agriculture*, Jg. 12, Nr. 11, S. 1921, 2022. Adresse: <https://www.mdpi.com/2077-0472/12/11/1921>.
- [19] Faster Capital, *Die Magie der Fourier Transformation: Eine FFCS-Perspektive*, Document, Copyright 2024. All Rights Reserved., 2024. Adresse: [https://t.ly/RL\\_ME](https://t.ly/RL_ME).

- [20] R. Gandhi, „R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms,“ *Towards Data Science*, Juli 2018, Published in Towards Data Science. Adresse: <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>.
- [21] K. Jha, A. Doshi, P. Patel und M. Shah, „A comprehensive review on automation in agriculture using artificial intelligence,“ *Artificial Intelligence in Agriculture*, Jg. 2, S. 1–12, 2019, ISSN: 2589-7217. DOI: 10.1016/j.aiia.2019.05.004. Adresse: <https://www.sciencedirect.com/science/article/pii/S2589721719300182>.
- [22] R. Singh und Prajneshu, „Artificial neural network methodology for modelling and forecasting maize crop yield,“ *Agricultural Economics Research Review*, Jg. 21, S. 5–10, 2008.
- [23] C. Prakash, A. S. Rathor und G. S. M. Thakur, „Fuzzy based Agriculture expert system for Soyabean,“ Jg. 113, 2013.
- [24] B. J. LaMeres und M. H. Nehrir, „Design and implementation of a fuzzy logic-based voltage controller for voltage regulation of a synchronous generator,“ *IEEE Computer Applications in Power*, Jg. 19, Nr. 4, S. 12, 1998. Adresse: [https://www.montana.edu/blameres/vitae/publications/a\\_thesis/thesis\\_001\\_bsee.pdf](https://www.montana.edu/blameres/vitae/publications/a_thesis/thesis_001_bsee.pdf).
- [25] GeeksforGeeks. „Fuzzy Logic — Einführung,“ GeeksforGeeks. (unknown), Adresse: <https://www.geeksforgeeks.org/fuzzy-logic-introduction/>.
- [26] N. S. Chauhan. „Decision Tree Algorithm, Explained.“ Published on KDnuggets on February 9, 2022, KDnuggets. (Feb. 2022), Adresse: <https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html>.
- [27] Stiftung für Qualität und Wirtschaftlichkeit im Gesundheitswesen. „Wie funktioniert das Nervensystem?“ Stiftung für Qualität und Wirtschaftlichkeit im Gesundheitswesen. (2023), Adresse: <https://www.gesundheitsinformation.de/wie-funktioniert-das-nervensystem.html> (besucht am 19.03.2024).
- [28] L. Wuttke. „Künstliche Neuronale Netzwerke: Definition, Einführung, Arten und Funktion,“ Datasolut. (2023), Adresse: <https://datasolut.com/neuronale-netzwerke-einfuehrung/> (besucht am 19.03.2024).
- [29] A. Arnx. „First neural network for beginners explained (with code),“ Towards Data Science. (2019), Adresse: <https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cf37e06eaf> (besucht am 19.03.2024).
- [30] A. D. Dongare, R. R. Kharde und A. D. Kachare, „Introduction to artificial neural network,“ *International Journal of Engineering and Innovative Technology (IJEIT)*, Jg. 2, Nr. 1, S. 189–194, 2012.

- [31] AltexSoft Inc. „Unsupervised Learning: Algorithms and Examples.“ (2021), Adresse: <https://www.altexsoft.com/blog/unsupervised-machine-learning/>.
- [32] Synopsys, Inc. „What is Reinforcement Learning?“ (2024), Adresse: <https://www.synopsys.com/ai/what-is-reinforcement-learning.html>.
- [33] Lark Editorial Team. „Generalization.“ (2023), Adresse: [https://www.larksuite.com/en\\_us/topics/ai-glossary/generalization](https://www.larksuite.com/en_us/topics/ai-glossary/generalization).
- [34] V. Kanade. „What Is Pattern Recognition? Working, Types, and Applications,“ Spiceworks Inc. (Mai 2023), Adresse: <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-pattern-recognition/>.
- [35] Trendskout. „Bilderkennung (KI): von den Anfängen der Technologie bis zu den unzähligen Geschäftsanwendungen von heute,“ Trendskout. (2024), Adresse: <https://trendskout.com/de/losungen/image-recognition-und-bilderkennung/>.
- [36] Stanford Vision Lab, Stanford University, Princeton University. „ImageNet.“ Home-Website, Stanford Vision Lab. (2020), Adresse: <https://www.image-net.org/>.
- [37] Manasa. „5 Best Machine Learning Algorithms for Image Recognition.“ 2 min read, Medium. (Aug. 2023), Adresse: <https://medium.com/@mansi89mahi/5-best-machine-learning-algorithms-4-image-recognition-ab0eee5e2931>.
- [38] International Business Machines Corporation. „What are convolutional neural networks?“ International Business Machines Corporation. (), Adresse: <https://www.ibm.com/topics/convolutional-neural-networks>.
- [39] S. Cho, T. Kim, D.-H. Jung u. a., „Plant growth information measurement based on object detection and image fusion using a smart farm robot,“ *Computers and Electronics in Agriculture*, Jg. 197, S. 106407, 2023. DOI: [10.1016/j.compag.2023.106407](https://doi.org/10.1016/j.compag.2023.106407). Adresse: <https://www.sciencedirect.com/science/article/pii/S0168169923000911>.
- [40] C. Writer, *Monitoring Plant Growth using Computer Vision*, <https://blog.roboflow.com/monitor-plant-growth/>, Dez. 2023.
- [41] D. Pandey, U. Suman und A. Ramani, „An Effective Requirement Engineering Process Model for Software Development and Requirements Management,“ in *2010 International Conference on Advances in Recent Technologies in Communication and Computing*, 2010, S. 287–291. DOI: [10.1109/ARTCom.2010.24](https://doi.org/10.1109/ARTCom.2010.24).
- [42] Figma, Inc. „Use case template.“ (Year), Adresse: <https://www.figma.com/templates/use-case-template/>.
- [43] „Growth Spurts.“ Accessed on May 28, 2024, The Sill Inc. (2024), Adresse: <https://www.thesill.com/blog/how-do-plants-grow>.

- [44] „Environmental factors affecting plant growth.“ Accessed on May 28, 2024, Oregon State University. (2024), Adresse: <https://extension.oregonstate.edu/gardening/techniques/environmental-factors-affecting-plant-growth>.
- [45] Arduino S.r.l. „Arduino® UNO R4 WiFi.“ (2024), Adresse: <https://store.arduino.cc/products/uno-r4-wifi>.
- [46] M5Stack. „Light Sensor Unit with Photo-resistance.“ (2024), Adresse: <https://shop.m5stack.com/products/light-sensor-unit>.
- [47] Adafruit Industries, LLC. „Adafruit Sensirion SHT40 Temperature & Humidity Sensor - STEMMA QT / Qwiic.“ (Year), Adresse: <https://www.adafruit.com/product/4885>.
- [48] M5Stack. „Watering Unit with Moisture Sensor and Pump.“ (Year), Adresse: <https://shop.m5stack.com/products/watering-unit-with-mositure-sensor-and-pump>.
- [49] „OV7670 300KP VGA camera.“ © 2024 AZ-Delivery, AZ-DELIVERY. (2024), Adresse: <https://t.ly/Rcqwr>.
- [50] EMQX Team. „What Is MQTT and Why Is It the Best Protocol for IoT?“ © 2013-2024 EMQ Technologies Inc. All rights reserved. (2023), Adresse: <https://www.emqx.com/en/blog/what-is-the-mqtt-protocol>.
- [51] „Internet of Things with Python.“ © 2011-2021 www.javatpoint.com. All rights reserved. Developed by Tpoint Tech. (2021), Adresse: <https://www.javatpoint.com/internet-of-things-with-python>.
- [52] A. Sargent. „How InfluxDB Works with IoT Data.“ 548 Market St, PMB 77953, San Francisco, California 94104, © 2024 InfluxData Inc. All Rights Reserved. Sitemap. (2021), Adresse: <https://www.influxdata.com/blog/how-influxdb-iot-data/>.
- [53] A. Fleck. „Alexa, What’s America’s Favorite Smart Speaker?“ Accessed on May 28, 2024, Statista GmbH. (2024), Adresse: <https://www.statista.com/chart/23943/share-of-us-adults-who-own-smart-speakers/>.
- [54] „What is the Alexa Skills Kit?“ Last updated: Apr 23, 2024, Amazon.com, Inc. or its affiliates. (2024), Adresse: <https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html>.
- [55] „What is Cucumber?“ SmartBear Software. (2019), Adresse: <https://cucumber.io/docs/guides/overview/>.
- [56] A. Rosebrock. „Measuring size of objects in an image with OpenCV.“ Last updated on July 8, 2021. (2024), Adresse: <https://pyimagesearch.com/2016/03/28/measuring-size-of-objects-in-an-image-with-opencv/>.

- [57] OpenCV. „Object Detection.“ Generated on Wed Dec 27 2023 21:56:15. (2023), Adresse: [https://docs.opencv.org/4.9.0/d5/d54/group\\_\\_objdetect.html](https://docs.opencv.org/4.9.0/d5/d54/group__objdetect.html).
- [58] „Q-Learning Agents.“ The MathWorks, Inc. (1994-2024), Adresse: <https://de.mathworks.com/help/reinforcement-learning/ug/q-learning-agents.html>.
- [59] I. US. „Java vs. Python: Main Differences and What to Choose.“ accessed on May 28, 2024. (2023), Adresse: <https://t.ly/pR2Jj>.
- [60] B. Lutkevich und V. Silverthorne. „Lean Software Development (LSD).“ TechTarget, accessed on May 28, 2024. (2024), Adresse: <https://www.computerweekly.com/de/definition/Lean-Software-Development-LSD>.
- [61] „HYG EFS-10 Feuchtesensor.“ Accessed on May 28, 2024, reichelt elektronik GmbH. (2024), Adresse: <https://t.ly/C-540>.