

Gymnasium Buckhorn

Besondere Lernleistung im Fachbereich Informatik

Aufbau und Betrieb eines IoT-Netzwerkes zur Überwachung von Anbauflächen
am Beispiel eines Gewächshauses

Maximilian Nobis

Kurs: Informatik 4 Std./Woche

Betreuerin: Frau Marinescu

Datum: 2022/2023

Vorwort

Durch mein Praktikum am Ende des zweiten Semesters im Juni 2022 bei einem Ausrüster für Informationstechnologie in der Schifffahrtsbranche war ich motiviert, mich mit dem dort Erlernten vertieft zu beschäftigen. Zugleich habe ich durch Gespräche mit Freunden, der Teilnahme an dem Wirtschaftswettbewerb der ZBW und durch die Vorlesungen an den Freitagen im Gymnasium Buckhorn erfahren, dass man eine „Besondere Lernleistung“ (BLL) in die Benotung des Abiturs einbringen kann. So kam ich auf die Idee „das Angenehme mit dem Nützlichen zu kombinieren“, und eine „Besondere Lernleistung“ im Fachbereich Informatik zu fertigen.

Danksagung

Ich möchte an dieser Stelle meiner Betreuerin und Informatiklehrerin Frau Marinescu für das Ermöglichen, die Hilfestellung und die Motivation zu dieser Besonderen Lernleistung im Fachbereich Informatik am Gymnasium Buckhorn ganz herzlich danken.

Meinen Eltern danke ich für die Bereitschaft mir das Gewächshaus für meine Besondere Lernleistung zur Verfügung zu stellen. Im Verlauf des Projektes kam auch der finanzielle Aufwand für einiges an technischer Ausstattung und ein Cloudvolumen hinzu, den sie übernahmen.

Für den Austausch mit meinen Eltern, Freunden und Bekannten, die mir regelmäßig Feedback gegeben und mich moralisch unterstützt haben, danke ich hier ebenfalls ausdrücklich.

Einleitung

Diese Dokumentation beschreibt den Prozess der Entwicklung und des Betriebes meines Schülerprojekts im Bereich des Internet of Things (IoT), welches ich im Rahmen dieser besonderen Lernleistung am Gymnasium Buckhorn aufgebaut, programmiert und stetig erweitert habe.

Dabei gehe ich nicht nur auf den Fortschritt ein, sondern erläutere auch den geschriebenen Code und die verwendete Hardware. Das Ziel dieser Dokumentation soll es sein einen guten Überblick den Leser zu geben was im Laufe dieses Projektes alles geschafft worden ist.

Inhaltsverzeichnis

VORWORT	2
DANKSAGUNG	3
EINLEITUNG	4
INHALTSVERZEICHNIS	5
ABKÜRZUNGS- UND FREMDWORTVERZEICHNIS	6
DER WEG ZUR IDEE	7
VORÜBERLEGUNGEN	8
1. VERSION	9
AUFBAU UND UMSETZUNG	9
AUSWAHL DER PFLANZUNG (A)	9
AUSWAHL DER SENSOREN (B)	11
ZUSAMMENBAU ESP32 UND SENSOREN (C)	13
ESP32-PROGRAMMIERUNG (C)	13
2. VERSION	15
EINBINDUNG IN EIN CLOUDSYSTEM (E)	15
DATEN AN DEN SERVER SCHICKEN (E)	17
EINRICHTEN DER DATENBANK (D)	19
DATEN IN DER DATENBANK SPEICHERN (D)	20
GRUNDSTRUKTUR DER WEBSITE (F)	21
VERBINDUNG ZWISCHEN SERVER UND INTERNETSEITE (CLIENT) ERSTELLEN	22
WEITERE ENTWICKLUNG DER WEBSITE	23
Navbar	23
Main	23
Notifications	24
not found page	24
Passwort Reset	25
History	25
Documentation	26
EINBAU DER SENSOREN IN DAS GEWÄCHSHAUS (C)	27
3. VERSION	28
Diskussion	30
ZUSAMMENFASSUNG	30
PERSPEKTIVEN	30
QUELLENVERZEICHNIS	32
ABBILDUNGSVERZEICHNIS	36
EIGENSTÄNDIGKEITSERKLÄRUNG	37

Abkürzungs- und Fremdwortverzeichnis

AIoT Artificial Intelligence of Things - beschreibt die Kombination aus Artificial Intelligence (AI) und dem Internet of Things (IoT)

Arduino IDE integrierte Entwicklungsumgebung (IDE)- openSource-Software, integrierte Entwicklungsumgebung (IDE)- openSource-Software

Config File Ein Config-File (Konfigurationsdatei) ist eine Datei, die verwendet wird, um die Parameter und Anfangseinstellungen für einige Computerprogramme zu konfigurieren

ESP32 Der ESP32 ist eine kostengünstige und mit geringem Leistungsbedarf ausgeführte 32-Bit-Mikrocontrollerfamilie der chinesischen Firma Espressif, die im Jahr 2016 vorgestellt wurde. Die Mikrocontroller ermöglichen durch ihre offene Bauweise den Aufbau und die Vernetzung von netzwerkbasierenden Aktuatoren und Sensoren., muss noch erklärt werden

Github GitHub ist ein netzbasierter Dienst zur Versionsverwaltung für Software-Entwicklungsprojekte.

KI Künstliche Intelligenz

MERN Stack Bündelung von vier Schlüsseltechnologien zur Erstellung einer dreischichtige Architektur von (Frontend, Backend, Datenbank) vollständig mit JavaScript.

MQTT ist ein offenes Netzwerkprotokoll für Machine-to-Machine-Kommunikation (M2M), das die Übertragung von Telemetriedaten in Form von Nachrichten zwischen Geräten ermöglicht, trotz

hoher Verzögerungen oder beschränkter Netzwerke.

Open Source: Offene Quelle - Software, deren Quelltext öffentlich und von Dritten eingesehen, geändert und genutzt werden kann 7

OS steht für "operatingsystem"

Overengineeering Overengineering beschreibt eine Situation, in der ein Produkt robuster konstruiert wird oder mehr Merkmale aufweist, als es für seinen Verwendungszweck notwendig ist oder vom Anwender erwartet wird., Overengineering beschreibt eine Situation, in der ein Produkt robuster konstruiert wird oder mehr Merkmale aufweist, als es für seinen Verwendungszweck notwendig ist oder vom Anwender erwartet wird.

Overhead In diesem Kontext bezieht sich Overhead auf die zusätzlichen Daten oder Informationen

Raspberry Pi Der Raspberry Pi ist ein Einplatinencomputer, der von der britischen Raspberry Pi Foundation entwickelt wurde. Der Rechner enthält ein Ein-Chip-System von Broadcom mit einer Arm-CPU. Die Platine hat das Format einer Kreditkarte.

TCP Das Transmission Control Protocol (TCP, engl. für „Übertragungssteuerungsprotokoll“) ist ein Netzwerkprotokoll, das definiert, auf welche Art und Weise Daten zwischen Netzwerkkomponenten ausgetauscht werden sollen. Nahezu sämtliche aktuelle Betriebssysteme moderner Computer beherrschen TCP und nutzen es für den Datenaustausch mit anderen Rechnern.

Der Weg zur Idee

In meinem Praktikum habe ich gelernt, dass das Implementieren von Netzwerken zum Austausch von Daten nicht ausschließlich wirtschaftlichen Zwecken dienen muss. Durch die Auswertung der erfassten Daten, der eingesetzten Sensortechnik kann z.B. ermittelt werden, wieviel Kraftstoff tatsächlich benötigt wird. Fast immer kann die Menge optimiert werden. Das bedeutet das Kraftstoff gespart werden kann. Ein reduzierter Verbrauch ist nicht nur gut für den Eigner, sondern auch für die Umwelt und das Klima. Mir war deshalb wichtig, dass neben den wirtschaftlichen auch ein ökologischer Effekt, durch meine Arbeit erreicht werden kann.

In meinen Überlegungen bin ich schließlich auf die Landwirtschaft gekommen. Auch hier kann ein optimierter Verbrauch zu gleichen oder besseren Ergebnissen führen und bedeuten, dass z.B. Wasser, Energie oder Düngemittel eingespart werden können. Meine Recherchen hatten ergeben, dass sich bereits vieles in der Landwirtschaft verändert hat und es einige Beispiele gibt, wie mit Hilfe von IT der Landwirt unterstützt wird und die Automation von Vorgängen vorantreibt.

Bei meinen Recherchen bin ich auf Projekte gestoßen, welche mich zutiefst inspiriert haben. Zum einen gibt es das Unternehmen „Square Roots Urban Growers, Inc.“, welches in städtischen Gebieten vertikale Farmen betreibt und Technologie einsetzt, um das Wachstum von frischem, nachhaltigem und lokalem Gemüse zu fördern. Zum anderen gibt es aber auch Unternehmen, wie FarmBot, welche intelligente Roboter für die Landwirtschaft entwickeln. Diese Roboter sind in der Lage, den Anbau von Gemüse und anderen Pflanzen zu automatisieren, indem sie die Bodenvorbereitung, die Aussaat, die Bewässerung und die Unkrautbekämpfung übernehmen. Die Technologie von FarmBot soll dabei helfen, den landwirtschaftlichen Ertrag zu steigern und gleichzeitig die Arbeit der Landwirte zu erleichtern.

Des Weiteren gibt es auch Open Source- Projekte, wie „AIoT Food Storages System“, welches eine Lösung für das Management von Lebensmittellagern unter Verwendung von IoT-Geräten und KI- anbietet. Die Lösung soll dabei helfen, die Lebensmittelqualität zu verbessern, Abfälle zu reduzieren und eine effizientere Lagerverwaltung zu ermöglichen. Dabei haben mich besonders Design von Hardware und Software erstaunt. All diese Projekte sind größtenteils leider erst in anderen Ländern präsent. In Deutschland habe ich nur bisher eines der Projekte auf dem Gut Karlshöhe gesehen,

welche neue Technologien zu erprobt. (Bild und weiter erklären) ZukunftsBauer*innen – Urban Farming und Robotik | (gut-karlshoehe.de)

Meine weiteren Nachforschungen haben mich darin bestärkt ein kostengünstiges System zu entwickeln, welches den Landwirt/ Nutzer anzeigt, in welchem Zustand das bewirtschaftete Feld bzw. Gewächshaus ist, wie die Wachstumsbedingungen verbessert werden können und dabei trotzdem die Ressourcen schonend eingesetzt werden. In einem weiteren Schritt könnten dann Überlegungen erfolgen für eine weitgehende Automatisierung der Anlage. Praktischerweise habe ich das Gewächshaus im eigenen Garten für mein Projekt nutzen können.

Mein Anspruch an mein Projekt war auf vorgefertigte Lösungen möglichst zu verzichten, um möglichst viel eigene Erkenntnisse zu gewinnen.

Vorüberlegungen

Um aus der Idee Wirklichkeit werden zu lassen, musste ich mir darüber bewusstwerden, welche einzelnen Schritte notwendig sind, um das Gewächshaus im Garten in ein Netzwerk einzubinden und brauchbare Daten zu erhalten, die für eine Verbesserung des Ertrages oder gar eine mögliche Automatisierung des Gewächshauses genutzt werden könnten. Dafür müssen die Daten gespeichert und analysiert werden.

Dafür mussten folgende Aufgabe ausgeführt werden:

- A. Auswahl der Pflanzen
- B. Auswahl der Sensoren und Hardwarekomponenten
- C. Aufbau eines lokalen Netzwerkes
- D. Datenbank erstellen
- E. Einbindung in ein Cloudsystem
- F. Programmieren der Website

Auf die einzelnen Punkte gehe ich im Verlauf der Dokumentation entsprechend des Projektverlaufes ein. Sie bedingen sich gegenseitig. Veränderungen in einem Arbeitsschritt führen zwangsläufig zu notwendigen Nachbesserungen an anderer Stelle, so dass eine klare Abgrenzung in der Dokumentation der einzelnen Schritte ebenso schwierig ist, wie das Befolgen einer klaren Reihenfolge.

1. Version

Aufbau und Umsetzung

Mir war wichtig, mein aus dem Praktikum erlerntes Wissen anzuwenden und zu erweitern.

Die fertige Anwendung (F) plante ich daher mithilfe des sogenannten MERN-Stacks zu erarbeiten. Es handelt sich um einen Open-Source-Technologiesatz, der die Front-End-Technologien und Back-End-Systeme von **MongoDB** (Datenbank im Back-End), **Express.js** (Web-Framework), **React** (JavaScript-Bibliothek) und **Node.js** (Web-Server) kombiniert.

Insgesamt bieten diese Tools Entwicklern die Möglichkeit, leistungsstarke und effiziente Webanwendungen zu erstellen und zu verwalten, die ich aus meinem Praktikum bereits kannte.

Die Hardware (B) sollte aus einem Raspberry Pi und einem ESP32 bestehen.

Auswahl der Pflanzung (A)

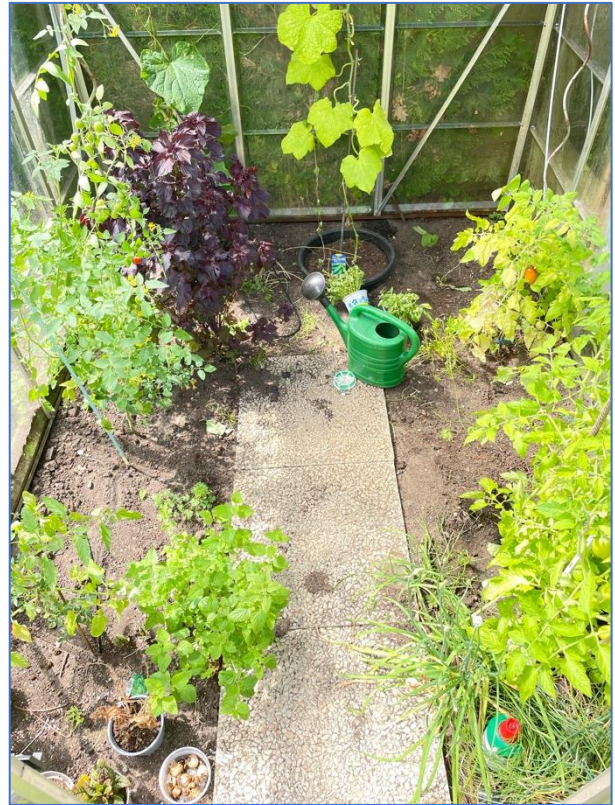
Die Auswahl der richtigen Pflanzen war für das Projekt von besonderer Wichtigkeit. Mir war bewusst, dass ich für eine gute Datenbasis mehrere Wochen bzw. Monate Daten erheben müsste. Die Wachstumsperiode vieler Pflanzen, selbst im Gewächshaus endet jedoch im Oktober. Ich brauchte Pflanzen, die praktisch das ganze Jahr wachsen können, um das Projekt nicht dadurch enden lassen zu müssen, dass die Bepflanzung stirbt. Ich benötigte also eine robuste Pflanzenart, bei der ich auch auf Vergleichsdaten, also Erfahrung anderer, zurückgreifen konnte. Nach Rücksprache mit meiner Mutter und ausgiebiger Internetrecherche habe ich mich klar für Feldsalat entschieden, der alle Kriterien erfüllt.

Feldsalat ist nicht nur leicht aufzuziehen und pflegearm - er bietet auch den Vorteil, dass er vorzugsweise in der dunklen Jahreszeit bzw. Winter kultiviert wird, also die Zeit, auf die ich mit meinem Projekt zusteuerte.

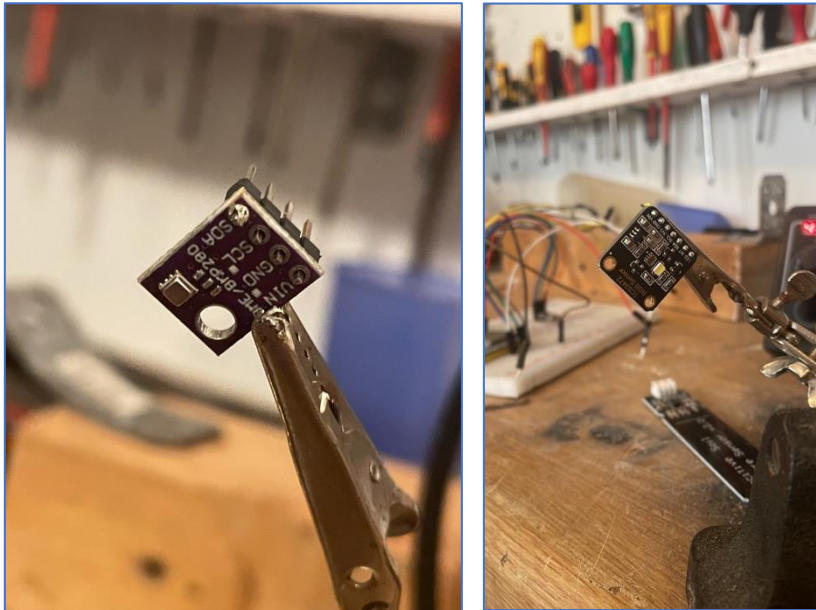
Den Feldsalat habe ich auf der gesamten Anbaufläche des Gewächshauses ausgesät:



So sah das Gewächshaus aus:



Auswahl der Sensoren (B)



Basierend auf der Entscheidung für Feldsalat recherchierte ich, welche Sensoren den Zustand des Feldsalates aufzeigen können. Wegen der Vielzahl an Wachstumsfaktoren und entsprechend hohen Anzahl an verschiedenen Sensoren habe ich versucht sogenannte Multisensoren zu verwenden, die mehrere Faktoren gleichzeitig erfassen können.

Bei meiner Recherche habe ich auch die Erfahrungen anderer zu den einzelnen Komponenten einbezogen und miteinander verglichen, um wirklich zuverlässige Sensoren zu finden. In den Videobeschreibungen anderer Nutzer wurden beispielsweise Sensoren mithilfe von Oszilloskopen verglichen, was mir auch einen guten Einblick in die Funktionsweise dieser Sensoren gegeben hat.

Ich habe mich dann für folgende Sensoren entschieden:

Zum einen habe ich mich für den Lichtsensor „TCS3472“ entschieden. Er kann präzise die Helligkeit in Lux angeben. Außerdem kann er die Farbzusammensetzung des Lichtes bestimmen.

Dann habe ich noch den „BME 280“ gekauft. Dies ist ein Sensor, der die Lufttemperatur, den Luftdruck und die Luftfeuchtigkeit messen kann. Dies sind weitere wichtige Faktoren für das Wachstum der Pflanze.

Außerdem habe ich einen Sensor zur Bestimmung der Bodenfeuchtigkeit gekauft. Er ist essenziell, um zu schauen, ob die Pflanze genügend Wasser für ihr Wachstum zur Verfügung steht.

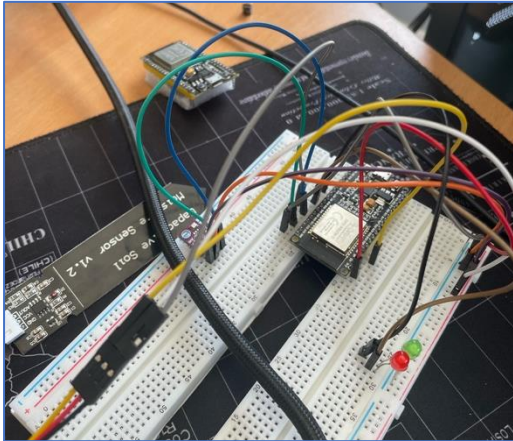
Überdies wäre auch noch möglich gewesen z.B. einen pH-Wert-Sensor oder weitere Sensoren für die Erfassung der Bodenqualität in das System zu integrieren. Aus finanziellen Gründen und Gründen der Komplexität (Stichwort „Overengineering“) habe ich dies Überlegungen jedoch in die Zukunft gelegt.

Nach Ankunft, der von mir bestellten Sensoren, begann ich umgehend mit dem Aufbau des geplanten Netzwerkes. (C) Doch schon bald traten erste Probleme auf. Zum einen stellte sich heraus, dass mein Raspberry Pi viel zu alt war, um die nötigen Programme reibungslos zu installieren. Dies dauerte eine gefühlte Ewigkeit und selbst dann konnte die CPU keine neue Software wie beispielsweise MongoDB mehr ausführen. Da ich außerdem Daten im Sekundentakt erhielt, erwies sich der Raspberry Pi als völlig überfordert. Es war schnell klar, dass ich hierfür eine Alternative benötigte, um das Projekt weiterzuführen.

Dennoch begann ich damit, den ESP32 zu programmieren. Der Microcontroller, hergestellt von Espressif, erwies sich als hervorragend geeignet, um analoge und digitale Sensordaten zu empfangen, zu verarbeiten und mittels MQTT-Protokolls an den Server, in diesem Fall den Raspberry Pi, zu übermitteln. Mein erster Schritt bestand jedoch darin, dass der ESP32 zunächst selbst die Sensordaten aufnahm.

Zusammenbau ESP32 und Sensoren (C)

Die Verkabelung der Sensoren war für mich persönlich eine Herausforderung. Ich hatte zwar bereits Grundkenntnisse damit und auch Erfahrungen beim Verlöten elektronischer Bauteile. Jedoch gingen die Baupläne für die Verknüpfung des ESP32 mit den Sensoren über mein bisheriges Verständnis hinaus. Glücklicherweise hatte ich ein



wenig Hilfe von meinem Großvater, der gelernter Elektriker ist. Auch Tutorials auf YouTube haben mir in dieser Phase meines Projektes geholfen.

ESP32-Programmierung (C)

Ich habe den ESP32 vorübergehend an meinen PC angeschlossen und ihn mithilfe des Programms Arduino IDE programmiert. Die dadurch erfassten Werte konnte ich über den Serial Monitor auslesen.

Zunächst einmal war es wichtig, dass die Pins (Anschlüsse) eingebunden werden. Dies geschah so:

```
1. #include <Wire.h>
```

Dann wurden die Pins aktiviert:

```
1. #define AOUT_PIN 34  
2. #define AOUT_PIN2 32
```

Um dann das Setup der Pins zu erstellen:

```
1. void setup() {  
2.     pinMode(34, OUTPUT);
```

```
3.           pinMode(32, OUTPUT);
4. }
```

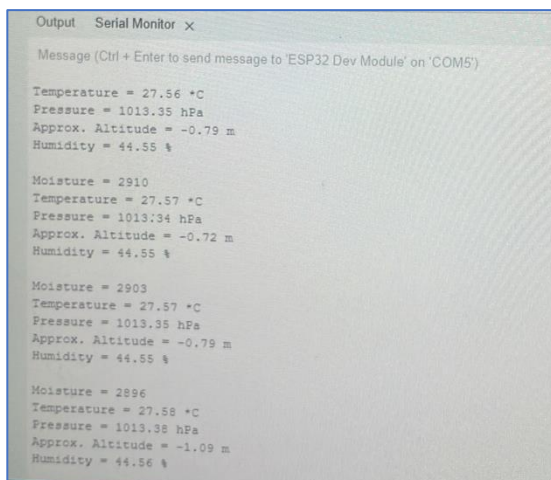
Nun sollten die Werte dauerhaft ausgegeben:

```
1. void loop() {
2.     moistureg = analogRead(AOUT_PIN);
3. }
```

Um brauchbare Werte zu erfassen, musste der String umgeformt werden:

```
1. void loop() {
2.     char moigString[8];
3.     dtostrf(moistureg, 1, 2, moigString);
4.     Serial.print("Moisture: ");
5.     Serial.println(moigString);
6. }
```

Dann konnte ich die ersten Werte auslesen:



The screenshot shows the Serial Monitor window in the Arduino IDE. The title bar reads 'Output Serial Monitor x'. Below the title bar, there is a message: 'Message (Ctrl + Enter to send message to 'ESP32 Dev Module' on 'COM5')'. The main area of the window displays the following sensor data, repeated four times:

```
Temperature = 27.56 °C
Pressure = 1013.35 hPa
Approx. Altitude = -0.79 m
Humidity = 44.55 %

Moisture = 2910
Temperature = 27.57 °C
Pressure = 1013.34 hPa
Approx. Altitude = -0.72 m
Humidity = 44.55 %

Moisture = 2903
Temperature = 27.57 °C
Pressure = 1013.35 hPa
Approx. Altitude = -0.79 m
Humidity = 44.55 %

Moisture = 2896
Temperature = 27.58 °C
Pressure = 1013.32 hPa
Approx. Altitude = -1.09 m
Humidity = 44.56 %
```

2.Version

Einbindung in ein Cloudsystem (E)

Da ich, wie schon erwähnt, massive Probleme mit meinem Raspberry Pi hatte, musste ich ihn aus meinem Konzept streichen. Dies hatte allerdings zur Folge, dass ich einen Ersatz für ihn brauchte. Die einfachste Möglichkeit war ihn durch ein aktuelleres und damit leistungstärkeres Modell zu ersetzen. Wegen der anhaltenden Chip-Krise waren die Raspberry Pi ausverkauft und in naher Zukunft auch nicht verfügbar. Auf Verkaufsplattformen, wie eBay etc. wurden sie deutlich überteuert angeboten. So habe ich mich dann gegen eine Lösung mit eigener Hardware entschieden und damit für eine cloudbasierte Lösung.

Dies tat ich in dem Bewusstsein, dass eine Lösung mit der richtigen Hardware wahrscheinlich langfristig günstiger und zudem auch zuverlässiger erreichbar ist, allerdings reizte mich zu diesem Zeitpunkt die Idee mein Netzwerk in eine Cloudlösung zu überführen und auch hier erste Erfahrungen zu sammeln.

Ich entschied mich für die Azure- Cloudlösung von Microsoft. Schon nach wenigen Klicks in der Anmeldemaske wurde mir eine fertige Lösung für eine zuverlässige Datenbank und Online-Monitoring angeboten. Meinem Anspruch folgend, möglichst viel eigenes Wissen zu sammeln, entschied ich mich für jedoch eine selbstentwickelte Lösung. Somit blieb mir auch die Möglichkeit perspektivisch schnell auf eine eigene Hardware zurückbauen zu können, sollte ich eine Raspberry Pi zu einem vernünftigen Preis erhalten.

Als Betriebssystem kamen viele Möglichkeiten in Frage. Tatsächlich habe ich auch mehrere Systeme ausprobiert. Zunächst habe ich Linux als Betriebssystem erprobt, da dies ein paar Vorteile gegenüber Windows bietet. (Vorteile benennen?)

Wegen der fehlenden Visuellen Oberfläche und der Tatsache, dass es nur die Möglichkeit der Verbindung über TCP in ein Terminal gab, entschied ich mich schließlich für eine virtuelle Maschine mit 4 Kerne (cores) und 16 GB Arbeitsspeicher (RAM), die mithilfe eines Windowsserver OS (operating system) lief. Dies war für meine Bedürfnisse völlig ausreichend.

Zunächst habe ich die MQTT-Verbindung mittels MQTT-Broker eingerichtet. Der Mosquitto MQTT-Broker war perfekt für die Kommunikation zwischen dem Arduino und dem Server. MQTT ist ein offenes Netzwerkprotokoll für die Kommunikation zwischen Geräten, welches vor allem im Bereich des Internets der Dinge (IoT) verwendet wird. Die Vorteile von MQTT, die mich zur Anwendung überzeugten waren, dass es auf TCP/IP läuft, weniger Overhead als HTTP hat und Nachrichten in einer Warteschlange gespeichert werden, bis sie an ihr Zielgerät gesendet werden können. Außerdem verfügt das Protokoll über eine integrierte Unterstützung für die Wiederverbindung und die Geräte und Applikationen sind komplett entkoppelt. Für mich war jedoch wichtig, dass man Ports in der Firewall öffnet. Nur so war sichergestellt, dass zwischen dem Arduino und dem Server bei Azure kommuniziert werden kann. In dem Fall war es der Port 1883 aus und eingehend.

Die entstandene Lösung habe ich ausgiebig getestet. Hierzu gibt es zum Beispiel die Möglichkeit über den MQTT Explorer. Das ist eine Open-Source-Desktopanwendung. Sie bietet eine grafische Benutzeroberfläche (GUI), mit der Benutzer MQTT-Broker, -Clients und -Themen durchsuchen, abonnieren, veröffentlichen und testen können.

Daten an den Server schicken (E)

Im dritten Schritt habe ich dann den Code vom Arduino so verändert, dass die aufgenommenen Werte von den Sensoren zur Cloud geschickt werden:

Dazu war es erst einmal notwendig, dass der ESP mit dem W-LAN verbunden wird.

Dies macht man wie folgt:

```
1. const char* ssid = "xxxxxx-xxxxxx";  
2. const char* password = "xxxxxxxxxxxxxxxxxxxxxxxxxx";
```

Danach bindet man noch die IP-Adresse des MQTT-Brokers ein:

```
1. const char* mqtt_server = "192.168.178.121";
```

Um nun den ESP mit dem Internet zu verbinden, brauchen wir neben der Setup-Funktion eine weitere Funktion für die Einbindung ins W-LAN.

```
1. void setup() {  
2.   setup_wifi();  
3. }  
4. void setup_wifi() {  
5.     delay(10);  
6.     Serial.println();  
7.     Serial.print("Connecting to ");  
8.     Serial.println(ssid);  
9.     WiFi.begin(ssid, password);  
10.    while (WiFi.status() != WL_CONNECTED) {  
11.        delay(500);  
12.        Serial.print(".");  
13.    }  
14.    Serial.println("");  
15.    Serial.println("WiFi connected");  
16.    Serial.println("IP address: ");  
17.    Serial.println(WiFi.localIP());  
18. }
```

Für einen eventuellen Verbindungsabbruch brauchen wir zum Sicherstellen der loop-Funktion eine weitere Funktion, die überprüft, ob die Verbindung immer noch besteht:

```
1. if (!client.connected()) {  
2.     setup_wifi();  
3.     delay(5000);  
4.     reconnect();  
5. }
```

```
5.         delay(5000);
6.     }
```

Die reconnect funktion sieht dann so aus:

```
1. delay(5000);
2. while (!client.connected()) {
3.     Serial.print("Attempting MQTT connection...");
4.     // Attempt to connect
5.     if (client.connect("ESP8266Client")) {
6.         Serial.println("connected");
7.         // Subscribe
8.         client.subscribe("esp32/output");
9.     } else {
10.        Serial.print("failed, rc=");
11.        Serial.print(client.state());
12.        Serial.println(" try again in 1 second");
13.        delay(1000);
14.    }
15. }
16. }
```

Nun müssen wir nur noch auf dem Router überprüfen, ob die Verbindung zwischen dem ESP mit dem W-LAN besteht.

Sollte dies nicht der Fall sein, müssen eventuell noch Ports über den Router freigegeben werden.

Nun müssen die Werte an den Mosquitto-Broker geschickt werden:

```
1. void setup() {
2.   client.setServer(mqtt_server, 1883);
3.   // falls auch Nachrichten zurückgeschickt werden sollen:
4.   // client.setCallback(callback);
5.
6.   }
7.   Die Funktion würde dann so aus sehen:
8. void callback(char* topic, byte* message, unsigned int length) {
9.   Serial.print("Message arrived on topic: ");
10.  Serial.print(topic);
11.  Serial.print(". Message: ");
12.  String messageTemp;
13.
14.  for (int i = 0; i < length; i++) {
15.    Serial.print((char)message[i]);
16.    messageTemp += (char)message[i];
17.  }
```

Im letzten Schritt muss dann noch die Nachricht veröffentlicht werden:

```
1. void loop() {  
2.           client.publish("esp/air/temperature", tempString);  
3.       }
```

Einrichten der Datenbank (D)

Nachdem nun die Verbindung steht, wird es Zeit die Datenbank einzurichten:

Wie bereits erläutert, hatte ich mich für MongoDB entschieden, um den MERN Stack zu nutzen. Entsprechend lud ich mir die Software auf der Seite des MongoDB Community Server herunter und entpackte sie.

Dank eines Installationsmanagers wird man relativ gut durch die Installation gesteuert. Ich habe zudem ein Config File angefertigt, welches u.a. den Installationspfad festlegt.

```
1. # Where and how to store data.  
2. storage:  
3.   dbPath: C:\Program Files\MongoDB\Server\6.0\data  
4.   journal:  
5.     enabled: true  
6. # engine:  
7. # wiredTiger:  
8.  
9. # where to write logging data.  
10. systemLog:  
11.   destination: file  
12.   logAppend: true  
13.   path: C:\Program Files\MongoDB\Server\6.0\log\mongod.log  
14.  
15. # network interfaces  
16. net:  
17.   port: 27017  
18.   bindIp: 10.0.0.4,127.0.0.1,0.0.0.0  
19. #security:  
20. security:  
21.   authorization: 'enabled'  
22.
```

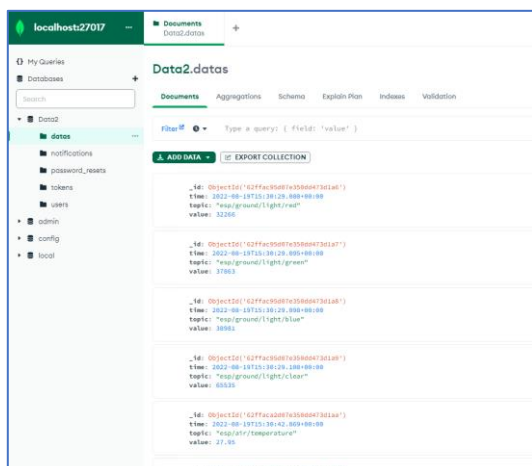
Es bietet sich zudem an, den MongoDB Kompass zu nutzen. Da dieser eine graphische Aufbereitung seiner Datenbank anbietet. Als letzten Schritt ist es nun noch wichtig die Ports zu öffnen, um die Daten erfassen zu können.

Daten in der Datenbank speichern (D)

Dann müssen wir die MQTT-Nachrichten in die MongoDB Datenbank schreiben. Hierfür habe ich mich eines fertigen Skriptes bedient und es an meine Bedürfnisse angepasst:

```
1. #!/usr/bin/env python3
2. import paho.mqtt.client as mqtt
3. import datetime
4. from pymongo import MongoClient
5. import logging
6. logging.basicConfig(filename='mqtt2mongodb.log', level=logging.DEBUG)
7. def on_connect(client, userdata, flags, rc):
8.     print("Connected with result code "+str(rc))
9.     client.subscribe("esp/#")
10.    logging.info('Connected with result code '+str(rc))
11. def on_message(client, userdata, msg):
12.    receiveTime = datetime.datetime.now()
13.    message = msg.payload.decode("utf-8")
14.    isfloatValue = False
15.    try:
16.        val = float(message)
17.        isfloatValue = True
18.    except:
19.        isfloatValue = False
20.    post = {"time": receiveTime, "topic": msg.topic, "value": val}
21.    if isfloatValue:
22.        print(str(receiveTime) + ": " + msg.topic + " " + str(val))
23.        logging.info(str(receiveTime) + ": " + msg.topic + " " + str(val))
24.        post = {"time": receiveTime, "topic": msg.topic, "value": val}
25.    else:
26.        print(str(receiveTime) + ": " + msg.topic + " " + message + msg.name)
27.        logging.info(str(receiveTime) + ": " + msg.topic + " " + message + msg.name)
28.        post = {"time": receiveTime, "topic": msg.topic, "value": message, "name": msg.name}
29.    collection.insert_one(post)
30. # Set up client for MongoDB
31. #mongoClient = MongoClient(username='admin', password='admin', port='27017')
32. mongoClient = MongoClient(
33.     "mongodb://localhost:27017")
34. db = mongoClient.Data2
35. collection = db.datas
36. # Initialize the client that should connect to the Mosquitto broker
37. client = mqtt.Client()
38. # Blocking loop to the Mosquitto broker
39. client.on_message = on_message
40. client.loop_forever(), 1883)
41. # Blocking loop to the Mosquitto broker
42. client.loop_forever()
```

Die ersten Daten in der Datenbank konnte ich dann mithilfe des MongoDB-Compass auslesen:



Grundstruktur der Website (F)

Jetzt geht es an die Darstellung der Messwerte auf der Website. Alles ist so weit vorbereitet, dass der Benutzer schließlich auch sehen kann, wie es um die Messwerte der Anbaufläche, hier im Gewächshaus, steht. Der Anspruch ist, dass dies möglichst benutzerfreundlich geschieht.

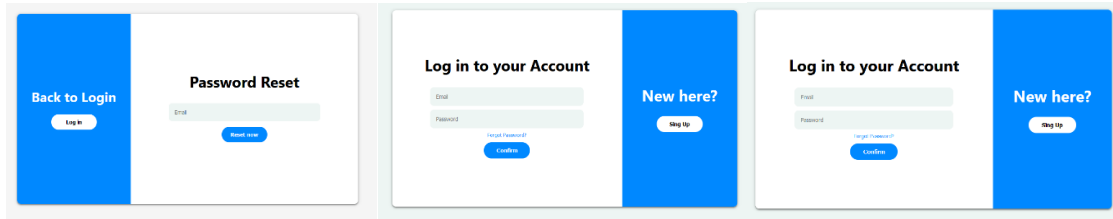
Die Grundstruktur sieht also wie folgt aus:



Zunächst wird eine Landingpage benötigt. Sie soll zunächst die wichtigsten Informationen zum Zweck der Homepage enthalten und die Registrierung von Benutzern ermöglichen.

Zudem werden Unterseiten zur Landingpage notwendig. Hierüber sollen Log-in und Log-out der registrierten Kunden erfolgen. Des Weiteren sollen dem eingeloggten Nutzer übersichtlich die erfassten Daten seiner vernetzten Anbaufläche zur Verfügung stehen. Hierfür bedarf es eine Kommunikation zwischen der Internetseite und dem Server, der die Datenbank beherbergt. Hierfür ist ein weiterer Schritt notwendig. Dies geschieht mittels einer API-Schnittstelle. Die Daten sollen passwortgeschützt sein. Dieses Passwort

wird verschlüsselt gespeichert, um den Datenschutz sicherzustellen. Auf diese Weise kann der Schaden vermindert werden beim Hacken der Datenbank.



Verbindung zwischen Server und Internetseite (Client) erstellen

Zunächst einmal habe ich dafür einen neuen Ordner angelegen müssen und diesen mit GitHub verbunden. Ich habe GitHub als Plattform benutzt, um den Stand des Projektes dauerhaft in der benannten Cloud zu sichern. Außerdem war es nun auch möglich das Projekte von überall herunterzuladen und daran zu arbeiten.

Ich habe mich dann für ein Client-Server-Modell, dem Standardkonzept für die Verteilung von Aufgaben innerhalb eines Netzwerkes, entschieden, da so die Komponenten von Client und Server auf unterschiedlichen Rechnern laufen können. Dafür mussten zwei weitere Ordner erstellt werden.

Zunächst einmal musste der Server aufgesetzt werden, damit der Client seine API-Anfragen an den Server senden kann.

Dazu wird erst einmal React über die Kommandozeile (Command-line interface) initialisiert.

1. `npx create-react-app my-app`
2. `cd my-app`
3. `npm start`

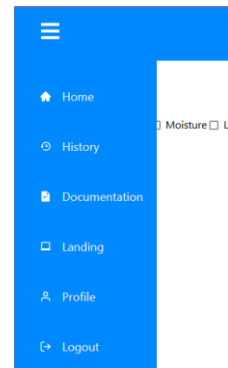
Die standardmäßig von React vorgeschlagenen und erstellten Dateien mussten nun meinen Bedarfen entsprechend angepasst bzw. gelöscht werden.

Für ein tieferes Verständnis des Tools habe ich einen Online-Kurs zur Erstellung von Serverstrukturen gemacht. Dies hat mir zum einen geholfen die Anwendung von React an sich zu verstehen und zum anderen hatte ich dann eine Grundstruktur entwickelt, auf die ich aufbauen konnte.

Weitere Entwicklung der Website

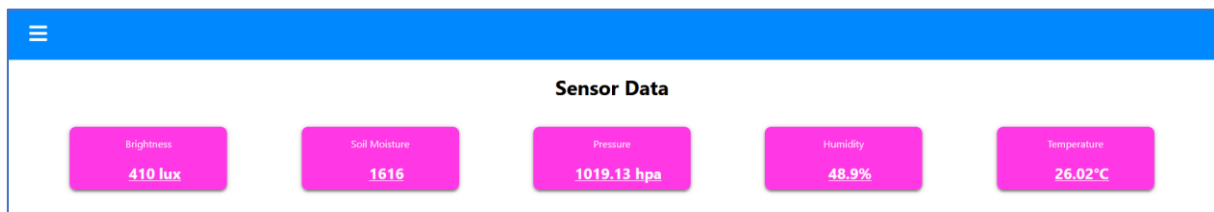
Navbar

Für das bessere und schnellere Navigieren auf der Website habe ich eine Navigationsleiste/Navbar eingefügt. Von ihr aus kann der Benutzer simpel zu allen Unterseiten navigieren.

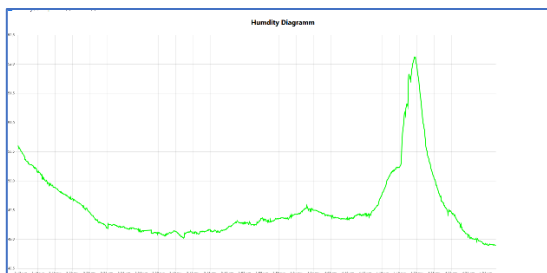


Main

Bei den weiteren Überlegungen zur Verbesserung meiner Hauptseite (Main) entschied ich, dass es wichtig ist, die aktuellen Sensordaten schnell und live anzuzeigen. Sie sollten für den Benutzer direkt einsehbar sein. Dafür habe ich mehrere kleine Boxen mit den Infos ganz oben platziert.



Außerdem denke ich, dass der Benutzer aber auch sehen möchte, wie sich die Daten im Laufe des Tages verändert haben. Dazu habe ich entsprechende Diagramme eingefügt.



Die Umsetzung davon gestaltete sich allerdings schwierig. Zum einen kam das Problem auf, dass sich bereits nach wenigen Wochen der Datenerfassung bereits mehrere Millionen Datensätze aufgebaut hatten. Zudem kamen im Sekundentakt neue hinzu. Dies war sowohl für den Server als auch für den Browser zu viel. Ich musste die Darstellung daher schlanker gestalten und weniger Daten abfragen. Allerdings war dieser Kompromiss nicht die optimale Lösung. So stellte ich vorübergehend weniger Daten im Diagramm dar, bis ich eine bessere Lösung dafür gefunden hatte.

Dabei wurde mir ein weiteres Problem bewusst. Die bis dahin erfassten Daten waren nicht datiert, so war eine korrekte Darstellung der Daten im Diagramm nicht garantiert.

Leider musste ich dann alle Daten, die ich bis dahin gesammelt habe, aufgeben und die neuen Daten mit einem Datumsstempel versehen. Dies geschieht an der Stelle im Script, wenn die MQTT-Datensätze in die Datenbank von MongoDB geschrieben werden.

Notifications

In Anlehnung an meine Erfahrungen aus dem Praktikum wollte ich auf der Website Nachrichten anzeigen lassen. Diese sollten wichtige Meldungen oder Warnungen zu den Messdaten enthalten, die in Abwesenheit entstanden sind. Wenn es beispielsweise Frost gab, sollte dies dann mit der genauen Zeitspanne angezeigt werden oder aber wenn übermäßige Trockenheit herrschte, sollte gemeldet werden, falls diese Pflanzen gegossen werden mussten. Es könnte sogar ein entsprechender Hinweis per E-Mail oder Textnachricht (SMS) verschickt werden. Dies war sehr aufwendig, da dauerhaft ein Script laufen muss, welches die einkommenden Nachrichten bewertet und sie gegebenenfalls als Nachricht verschickt.

Der Vorteil lag jedoch auf der Hand, dass dadurch, wenn auch zunächst nur manuell, in die Versorgung der Pflanzung eingegriffen werden kann. Schließlich war das auch das langfristige Ziel des Projektes. Ich wollte zukünftig in die Steuerung der Bewässerungs-, Heiz- oder Lüftungssysteme eingreifen können. In der aktuellen Projektphase ist diese Funktion jedoch noch nicht eingebunden.

not found page

Bei dem weiteren Ausbau meiner Website, bin ich häufig auf Unterseiten gestoßen, die es noch nicht gab. Dazu habe ich dann eine “not found page” erstellt. Damit erfolgte der entsprechende Hinweis, dass man einer Seite gelandet ist, die nicht existiert. Beim Design habe mich dazu bei großen Webseiten inspirieren lassen. Wichtig bei der Programmierung ist die Möglichkeit über einen Button zurück zu Startseite gelangen zu können.

page not found

Sorry, we can not find the page you are looking for!

Error: 404

[Zurück zur Startseite](#)

Password Reset

Wem ist es nicht schon passiert, dass er sein Passwort vergessen hat? Daher war es aus meiner Sicht notwendig auch eine „Passwort vergessen“-Option einzufügen.

Dafür habe ich einen Button Hauptseite hinzugefügt, der direkt auf die Seite führt, wo man sein man dann per Eingabe der E-Mail-Adresse sein Passwort zurücksetzen kann.

Die Anfrage wird dann an den Server geschickt.

Hier ist es dann wichtig sicherzustellen, dass die Identität der Benutzer geprüft werden kann. Dafür habe ich programmiert, dass eine vorübergehende Seite erstellt wird, die nur über einen Link geöffnet werden kann.

Hierfür wird von dem Server ein Token generiert. Dieser wird dann an die Seite angehängen.

Wenn ich also mein Passwort zurücksetzen möchte, schickt der Server dann eine Mail an mich mit dem Link: <http://beispiel.com/pwd-reset/token> über den ich dann das Passwort zurücksetzen kann, indem ich ihn öffne. Der Server überprüft, ob der Token mit dem versandten übereinstimmt und lässt dann den Benutzer ein neues Passwort vergeben.

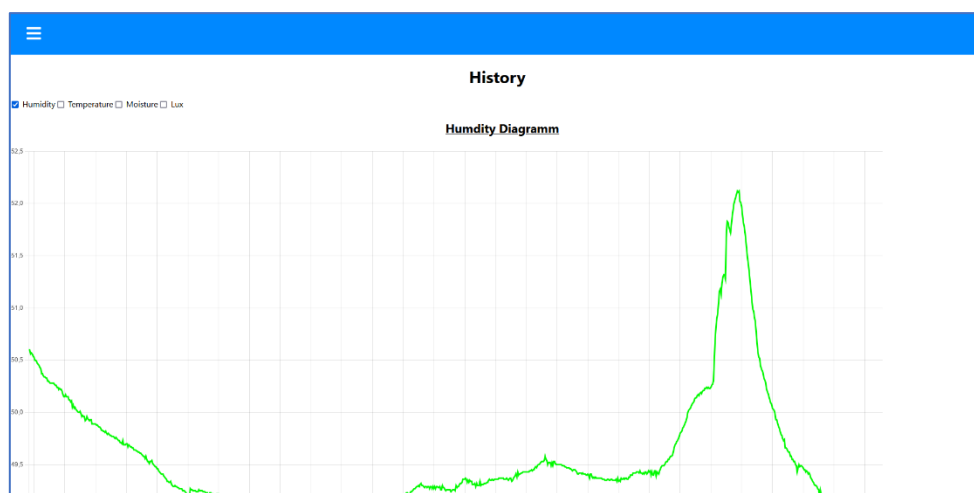
History

Ein weiterer wichtiger Punkt bei der Programmierung der Website war für mich die Schaffung der Möglichkeit, Elemente der Main Seite auf eine History-Seite zu verschieben. So kann die Main Seite übersichtlich bleiben und dennoch kann ich einen detaillierten Blick auf die Daten zu ermöglichen.

Wichtig war es hier für mich, dass die Diagramme nicht die Gesamtheit der Werte anzeigen, sondern nur diejenigen die der Benutzer auswählt. Dafür habe ich mehrere Checkboxes eingefügt.

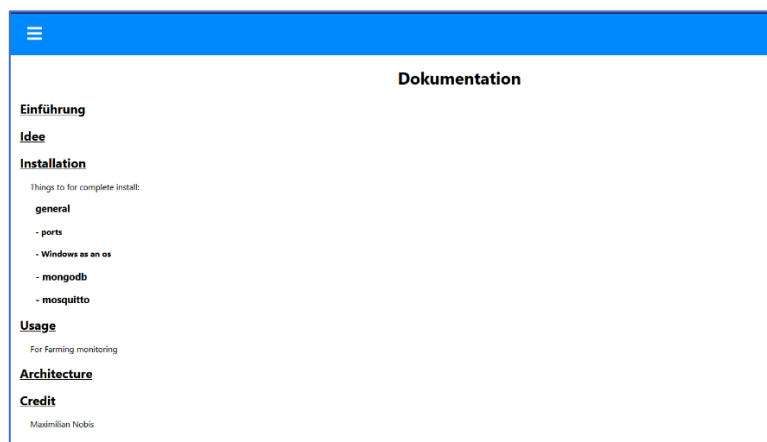
Hier können nun die Luftfeuchtigkeit, die Temperatur, die Bodenfeuchtigkeit und die Helligkeit in Abhängigkeit von der Zeit angezeigt werden.

In der Zukunft sollen hier auch noch weitere Auswahlmöglichkeiten entstehen. So ist geplant, dass der Benutzer die Zeitspanne einstellen kann. Des Weiteren sollen die Diagramme so verschiebbar sein, dass mehrere Werte vergleichend angezeigt werden können.



Documentation

Das neueste Modul der Website sieht eine Dokumentation für den Benutzer vor, die einzelne Funktionalitäten erklärt.



Einbau der Sensoren in das Gewächshaus (C)

Nach erfolgreichem Abschluss der Testphase, -alles lief-, musste ich eine langfristige Lösung für die Stromversorgung der Sensoren finden. Dies geschah bislang noch über eine Powerbank in Bodennähe neben den Pflanzen, die regelmäßig gegossen werden mussten. Das war auf Dauer gesehen nicht perfekt.

Daher habe ich mich dazu entschieden oberhalb der Eingangstür zum Gewächshaus eine Box zu installieren, die alle technischen Geräte enthielt und zumindest spritzwassergeschützt ist. Perspektivisch könnte man bei weiteren Kunden mit solchen Boxen die Installation simpel gestalten, da bereits vorgefertigt werden könnte.

3. Version

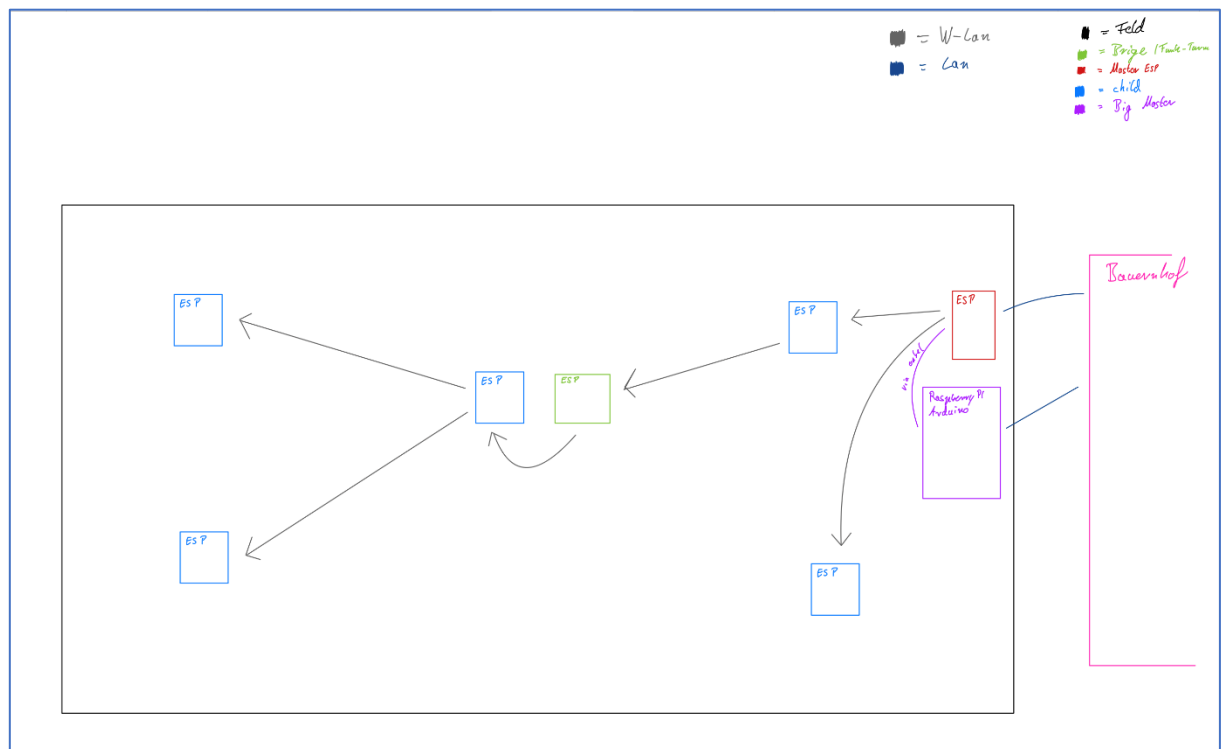
Bei der Erweiterung meines Projektes auf die 3.Version wollte ich ein grundlegendes Problem angehen. Bei dem Einsatz auf großen Feldern muss die Verbindung zum Internet gewährleistet werden, um Übertragung der Daten von den Sensoren zu ermöglichen.

Nach einiger Recherche und Nachdenken kam ich auf unterschiedliche Lösungen. Zum einen bestand die Möglichkeit den ESP32 mit Hilfe des Helium-Netzwerkes zu verbinden. Wegen der hohen Kosten und der nicht garantierten Netzwerkabdeckung fiel diese Lösung schnell aus.

Des Weiteren besteht die Möglichkeit die Verbindung über SIM-Karten herzustellen. Wenn man davon ausgeht, dass ein durchschnittliches Anbaufeld ca. $1 \times 0,5$ km groß ist, braucht man etwa fünf Sensor-Stationen, um den Zustand des Feldes bestmöglich zu erfassen. Dies würde bei Ausstattung mit je einer SIM-Karte pro Station also auch monatlich eine erhebliche Summe kosten.

Meine Absicht war jedoch eine kostengünstige, wenn nicht sogar kostenlose Möglichkeit zu finden. Daher war es meine Idee das Internet „weiterzuleiten“. Dies konnte man dann direkt über die bereits installierten fünf Stationen erreichen. Es brauchte somit nur einen Internetzugang und nicht fünf, wie bei der vorherigen Lösung.

So sieht dann das Konzept aus:



Es ist ein eigenes Netzwerk mit eigenen Rollen. Die Kommunikation hier soll über ESP-NOW ermöglicht werden. Mithilfe des ohnehin schon verbauten W-LAN-Chips konnte ein W-LAN aufgebaut werden, welches im optimalen Fall bis zu 500 Meter weit reichen konnte. Diese Dimensionen waren mir vor der Recherche nicht bewusst.

Das Netzwerk besteht aus fünf Rollen:

Die erste Rolle ist der „Head Master“. Dies ist der Server der Cloudseite (hier MS Azure). Von ihm werden an alle Systeme Updates und andere Befehle geschickt.

Die zweite Rolle ist der „Big Master“. Diese Rolle übernimmt der Arduino oder ggf. ein Raspberry Pi. Er ist verantwortlich dafür, die Updates herunterzuladen und ggf. die Website lokal auf einem Display anzuzeigen. Des Weiteren ist er dafür verantwortlich, die Internetverbindung aufzubauen. Je nach Situation muss er von einer LAN oder einer W-LAN- Verbindung ein eigenes W-LAN aufbauen, über die sich der Master verbinden kann. Dabei sorgt er auch noch dafür alle Daten gebündelt an den Server der Cloud zu senden. Darüber hinaus obliegt es ihm die Daten nach Duplikaten zu filtern.

Der „Master“ (3. Rolle) ist im ESP-NOW-Netzwerk wiederum dafür da, die Daten als eine Brücke zu empfangen und an den Head Master weiterzugeben. Er befehligt auch alle ESP32.

Die vierte Rolle „Bridge“ bzw. Funkturm ist dazu da, um auf weite Entfernungen die Verbindung zu ermöglichen. Er ist also eine Art Brücke zwischen dem Headmaster und den anderen ESP32, die als Sensoren da sind.

Die letzte Rolle bildet das sog. „Child“, welches nur Daten von seinen Sensoren sendet und Updates empfängt.

Diskussion

Zusammenfassung

Zusammenfassend beschreibt das Dokument einen Prozess zur Entwicklung meines Schülerprojekts im Bereich des Internet of Things (IoT). Es geht auch auf die Herausforderungen ein, die während des Prozesses aufgetreten sind und wie ich sie gelöst habe.

Das Dokument beschreibt die technischen Aspekte des Projekts, einschließlich der Hardware- und Softwarekomponenten und wie sie miteinander interagieren. Es werden auch mögliche Erweiterungen und Verbesserungen für das Projekt vorgeschlagen.

Insgesamt bietet das Dokument einen detaillierten Einblick in den Prozess der Entwicklung meines Schüler-IoT-Projekts, einschließlich der Ideenfindung und der technischen Umsetzung.

Perspektiven

Konkret habe ich die Erweiterung um ein neues Modul geplant, welches eine Kalenderfunktion enthält. Mit diesem Kalender kann der Benutzer seine Termine, wie

zum Beispiel geplante Erntetermine, einstellen. Mithilfe von Künstlicher Intelligenz sollen dann Vorhersagen über den tatsächlichen Erntetermin getroffen werden können.

Ein weiteres Feature des Systems wird die Automatisierung sein. Denkbar ist beispielsweise, wenn eine Benachrichtigung angezeigt wird, dass die Pflanze trocken ist, dass automatisch gegossen wird. Außerdem könnten tägliche Werte wie die Lufttemperatur basierend auf der Werte der Vergangenheit verglichen mit dem angekündigten Wetter vorhergesagt werden.

Darüber hinaus wird mit Regression eine nachvollziehbare Linie generiert, um abweichende Werte zu analysieren.

Die Umsetzung dieser Version ist bisher in der Planungsphase, soll aber baldmöglichst umgesetzt werden.

Das Projekt bietet zahlreiche Möglichkeiten zur Weiterentwicklung und unterschiedliche Anwendungsszenarien. Ein erster Schritt könnte sein, das System um weitere Sensoren und Aktoren zu erweitern, um so weitere Anwendungsfälle abzudecken und das System flexibler zu gestalten.

Um das Projekt “massentauglich” zu machen, müssten weitere Anbauflächen, also Benutzer, hinzukommen. Auch die Integration von Künstlicher Intelligenz könnte weitere Anwendungsmöglichkeiten eröffnen. Eine Möglichkeit hierfür wäre beispielsweise die Implementierung von Machine Learning-Algorithmen zur automatischen Erkennung bestimmter Umweltbedingungen. Des Weiteren könnte die Benutzeroberfläche erweitert und noch weiter optimiert werden, um sie noch intuitiver und benutzerfreundlicher zu gestalten. Zudem bietet sich die Möglichkeit, das System in bereits bestehende IoT- Plattformen zu integrieren und somit noch weitreichender, flexibler und kompatibler zu gestalten.

Insgesamt bin ich mit der Entwicklung meines Projektes zufrieden. Ich habe in der Zeit mein Wissen in den behandelten Bereichen, wie die Netzwerk- und Datenbankerstellung oder die Programmierung von Websites deutlich erweitern können.

Quellenverzeichnis

Erklärungen für Fremdwörter

<https://t2informatik.de/wissen-kompakt/overengineering/>

<https://de.wikipedia.org/wiki/Internetprotokollfamilie>

https://de.wikipedia.org/wiki/Transmission_Control_Protocol

https://en.wikipedia.org/wiki/Configuration_file

<https://de.wikipedia.org/wiki/GitHub>

https://de.wikipedia.org/wiki/Transmission_Control_Protocol

Bau Prozess:

<https://www.youtube.com/watch?v=oz0a7Ur7nko>

<https://www.instructables.com/ESP-NOW-WiFi-With-3x-More-Range/>

<https://www.youtube.com/watch?v=B6NZWLKGoCA>

https://github.com/espressif/esp-now/blob/master/User_Guide.md

<https://stackoverflow.com/questions/4308610/how-to-ignore-certain-files-in-git>

<https://github.com/HQarroum/awesome-iot>

https://github.com/gwilken/ariadne-io/tree/master/cloud_service

https://github.com/shivamsingha/aiot-food-storage/blob/main/sys_design.jpg

https://create.arduino.cc/projecthub/Arduino_Genuino/securely-connecting-an-arduino-mkr-wifi-1010-to-aws-iot-core-a9f365

https://seeeddoc.github.io/Grove-Gas_Sensor-O2/

<https://www.youtube.com/playlist?list=PLqTrJcRXDhn7XZvojDIINuW2GRUuskCI>

https://www.youtube.com/watch?v=-P8lj_LNwrM&list=PL4ZeBG19K1KSVioOQpXqktopZp_F1eIZZ

<https://www.youtube.com/watch?v=nyqykZK2Ev4>

<https://www.google.com/search?client=firefox-b-d&q=mqtt+ports>

<http://mqtt-explorer.com/>

<https://www.google.com/search?client=firefox-b-d&q=raspberryPi>

<https://www.google.com/search?client=firefox-b-d&q=esp32>

ph wert

<https://hevodata.com/blog/install-mongodb-on-ubuntu/#repo>

<https://docs.arduino.cc/tutorials/uno-wifi-rev2/uno-wifi-r2-mqtt-device-to-device><https://stackoverflow.com/questions/68266937/arduino-does-not-connect-to-mosquitto-on-both-windows-and-ubuntu>

<https://www.youtube.com/watch?v=a3H9rIj07sk>

<https://www.youtube.com/watch?v=HGgyd1bYWwE>

<https://jsfiddle.net/BlackLabel/yg9haq85/>

<https://www.youtube.com/watch?v=IVPDfnH39sg>

<https://levelup.gitconnected.com/fetch-api-data-with-axios-and-display-it-in-a-react-app-with-hooks-3f9c8fa89e7b>

<https://bobbyhadz.com/blog/javascript-cannot-read-property-find-of-undefined>

<http://net-informations.com/js/iq/unerror.htm>

<https://dev.to/iamatifriaz/how-to-fix-error-in-plugin-react-was-conflicted-between-packagejson-and-eslint-config-react-app-365d>

<https://www.youtube.com/watch?v=W36qvPICuSs>

<https://www.npmjs.com/package/react-spinners>

<https://bobbyhadz.com/blog/react-map-array-reverse>

<https://react-icons.github.io/react-icons/icons?name=bi>

<https://www.appsloveworld.com/reactjs/200/436/react-how-to-get-loading-spinner-to-show-while-api-data-loads-into-my-chart-js>

<https://stackoverflow.com/questions/21389341/disable-animation-with-charts-js>

https://js.devexpress.com/Documentation/Guide/UI_Components/Chart/Rotate_and_Invert_the_Chart/

<https://www.chartjs.org/docs/latest/samples/advanced/data-decimation.html>

<https://stackoverflow.com/questions/65348218/react-how-to-get-loading-spinner-to-show-while-api-data-loads-into-my-chart-js>

<https://react-bootstrap.github.io/components/spinners/>

<https://reactjs.org/docs/faq-ajax.html>

wifi problem

<https://www.youtube.com/watch?v=6zbEVAXVBjI>

<https://interline.pl/Information-and-Tips/LONG-RANGE-IOT>

https://www.oreilly.com/library/view/beginning-lora-radio/9781484243572/html/469311_1_En_7_Chapter.xhtml

heap out of mem problem

<https://sebastian.com/javascript-heap-out-of-memory/>

password reset

<https://stackoverflow.com/questions/45898789/react-router-pass-param-to-component>

<https://www.youtube.com/watch?v=kfw61IxHgW8>

<https://www.youtube.com/watch?v=npsi7ZkqvQo&t=317s> //5h

<https://myaccount.google.com/security?rapt=AEjHL4O36kXjcS7Rzqb3mllt5I9FRXWdK9KU-oPu1uygOZ9o9G6yoFLWYz4Hw4wExXNBHDoQLg-C6bZwUY4Qy-J0rZW-vqJw>

<https://blog.logrocket.com/client-side-routing-react-react-location/>

<https://medium.com/geekculture/how-to-use-nodemailer-with-gmail-to-send-e-mails-7198e707025d>

<https://kinsta.com/de/blog/gmail-smtp-server/>

<https://github.com/sk-Jahangeer/node-email-password-reset/blob/master/routes/passwordReset.js>

<https://www.geeksforgeeks.org/mongoose-updateone-function/>

<https://github.com/facebook/react/issues/24928>

https://www.youtube.com/watch?v=AClnCg_WCJk

<https://www.positronx.io/react-axios-tutorial-make-http-get-post-requests/>
<https://stackabuse.com/sending-post-json-requests-with-axios/>
<https://www.positronx.io/react-axios-send-asynchronous-http-post-request-tutorial/>
<https://www.bezkoder.com/react-axios-example/>
https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_storage_getitem
<https://blog.logrocket.com/localstorage-javascript-complete-guide/>
<https://www.youtube.com/watch?v=HGgyd1bYWwE>
<https://dev.to/pawel/how-to-secure-your-api-s-routes-with-jwt-token-4bd1>
<https://github.com/leftlogic/leftlogic/tree/master/data> //good example

format time

<https://stackoverflow.com/questions/68467263/convert-array-of-full-date-to-array-of-month-in-react>
<https://www.tutorialspoint.com/formatting-javascript-object-to-new-array>
<https://stackoverflow.com/questions/68761102/how-to-format-json-body-sent-from-axios>
<https://stackoverflow.com/questions/14638018/current-time-formatting-with-javascript>
<https://javascript.info/date>
<https://www.chartjs.org/docs/latest/axes/cartesian/time.html>
<https://www.chartjs.org/docs/latest/axes/cartesian/time.html>
<https://github.com/chartjs/chartjs-adapter-date-fns>
<https://moment.github.io/luxon/#/formatting>
<https://momentjs.com/docs/#/get-set/>
<https://github.com/chartjs/chartjs-adapter-moment>
<https://github.com/chartjs/awesome#adapters>
<https://www.chartjs.org/docs/master/getting-started/integration.html#bundlers-webpack-rollup-etc>
<https://github.com/chartjs/Chart.js/issues/9085>
<https://javascript.info/date>

eslint problem

<https://www.digitalocean.com/community/tutorials/workflow-nodemon>

build

https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/deployment
<https://www.youtube.com/watch?v=npsi7ZkjuQo&t=317s>
<https://www.freecodecamp.org/news/how-to-build-a-backend-application/>
<https://create-react-app.dev/docs/deployment/>
<https://www.youtube.com/watch?v=1RHDhtbqo94>

select diagramm

<https://www.digitalocean.com/community/tutorials/7-ways-to-implement-conditional-rendering-in-react-applications>

<https://ej2.syncfusion.com/react/documentation/check-box/how-to/name-and-value-in-form-submit/>

<https://www.mediaevent.de/html/select.html>

<https://github.com/kamronbatman/joi-password-complexity>

<https://www.geeksforgeeks.org/how-to-use-map-to-create-lists-in-reactjs/>

<https://www.freecodecamp.org/news/debouncing-explained/> //timedelay

404

<https://www.hosteurope.de/blog/wie-sie-mit-einer-kreativen-404-fehlerseite-und-lustigen-inhalten-punkten/>

<https://naveenda.medium.com/creating-a-custom-404-notfound-page-with-react-routers-56af9ad67807>

<https://codepen.io/uiswarup/pen/XWdXGGV> //very nice

calc internet actions

<https://www.google.com/search?client=firefox-b-d&q=seconds+in+day>

<https://www.google.com/search?client=firefox-b-d&q=bytes+into+mb>

<https://www.google.com/search?client=firefox-b-d&q=ms+network+monitor>

<http://www.steves-internet-guide.com/mqtt-sensors-traffic-observations/>

<https://stackoverflow.com/questions/752934/monitoring-network-resource-utlization-and-performance-of-a-windows-applicatio>

Staatliche unterstützung für mein projekt

<https://www.getcouped.com/blog/foerderungen-zuschuesse-startups-2022>

<https://www.gruenderkueche.de/fachartikel/8-foerderprogramme-fuer-startups-teilfinanzierung-durch-oeffentliche-foerderung/>

Abbildungsverzeichnis

Alle Bilder die in diesem Dokument sind, wurden von mir gemacht und teils bearbeitet.

Eigenständigkeitserklärung

Ich versichere, dass die Arbeit von mir selbstständig erarbeitet wurde und ich keine anderen als die angegebenen Hilfsmittel benutzt habe. Diejenigen Teile der Arbeit, die anderen Werken im Wortlaut oder dem Sinn nach entnommen wurden, sind als solche kenntlich gemacht.

Maximilian Nobis