# HYDROKULTUR-GARTEN

Ein effizientes Garten Projekt

**ABBILDUNG 1: PVC-RÖHREN FÜR ANBAU VON SALATEN**

***Projektarbeit 2024***
***Mikroprozessortechnik***

Samuel Haab, Paul Vodak
Techniker Informatik, Fachrichtung Applikationsentwicklung
Klasse: Z-TIN-21-T-a, Dozent: Christian Meier

# 1 Inhaltsverzeichnis

# 2 Organisation

## 2.1 Einleitung

Im Rahmen des Diplomprojektes für das Modul 'Mikroprozessortechnik' wird in dieser Arbeit ein hydroponisches Garten-System konzipiert und realisiert. Das Projekt wurde massgeblich durch die Inspiration von Samuels Schwester angeregt, die eingeladen wurde, an einem umfangreichen internationalen Hydrokulturprojekt mitzuarbeiten, sich aber letztendlich gegen das Vorhaben entschied.

Obwohl sich Samuels Schwester gegen das Projekt entschieden hat, hat es ihn dazu motiviert, sich im Rahmen eines Mikrocomputerprojekts intensiv mit der Thematik der Hydrokultur auseinanderzusetzen.

Das Ziel ist es, die Prinzipien der Hydroponik - also die Kultivierung von Pflanzen in einer nährstoffreichen Lösung anstelle von traditioneller Erde - mit den technologischen Möglichkeiten der Mikroprozessortechnik zu verbinden. Das Projekt hat zum Ziel, einen hydroponischen Garten als Prototyp zu entwickeln. Dabei sollen Methoden für den effizienten Pflanzenanbau ohne Erde erforscht werden.

Der Schwerpunkt liegt auf der Integration von Mikroprozessortechnik zur präzisen Kontrolle und Überwachung der Wachstumsparameter. Hierbei wird auch die Automatisierung der Nährstoff- und Sauerstoffversorgung der Pflanzen berücksichtigt.

Es dient als Schnittstelle zwischen theoretischem Wissen und praktischer Anwendung. Das Diplomprojekt vermittelt fundierte Kenntnisse in der Hydroponik und in der Anwendung von Mikroprozessortechnik zur Optimierung pflanzlicher Wachstumsprozesse.

## 2.2 Pflichtenheft

### 2.2.1 Kurzübersicht

Entwicklung eines automatisierten Hydroponik-Systems unter Verwendung eines Arduino-Mikrocontrollers zur Kontrolle und Überwachung des Wachstums von Pflanzen.

### 2.2.2 Zielsetzung

Im Folgenden haben wir eine Liste von Zielen erstellt, die im Rahmen dieses Projekts erreicht werden sollen. Auf der Grundlage dieser Ziele wird dann das Material bestellt.

| **Ziele** | **Dringlichkeit** |
|---|---|
| Rohre zur Anpflanzung von Kräutern/Salate | Muss |
| Automatische Anpassung der Wasser- und Nährstoffzufuhr basierend auf den Sensorwerten | Muss |
| Automatisierte Wasseraufbereitung mit Sauerstoff | Soll |
| Überwachung pH-Wert | Muss |
| Überwachung Nährstoffkonzentration (EC-Wert) | Muss |
| Überwachung Wassertemperatur | Soll |
| Benutzerfreundliche Überwachung und Steuerung der Anlage | Soll |
| Regelung der Wachstumslampen nach einem vorgegebenen Zeitplan. | Soll |

### 2.2.3 System Komponenten

Im Folgenden haben wir eine Liste der Komponenten zusammengestellt, die für die erfolgreiche Durchführung des Projekts erforderlich sind:

| **Element** | **Bestellstatus** |
|---|---|
| Arduino Mikrocontroller | Im Kit vorhanden |
| PH-Sensor zur Messung des Säuregehalts | Bestellt |
| EC-Sensor zur Überwachung der Nährstoffkonzentration | Bestellt |
| Temperatursensor für das Wasser | Bestellt |
| Wasser- und Luft-Pumpen | Bestellt |
| LED-Wachstumslampen | Bestellt – Demo |
| Display | Im Kit vorhanden |
| Relais zur Steuerung von Pumpen und Lampen | Im Kit vorhanden |
| Verschiedene Grössen von Schläuchen und Verbindungsstücken | Bestellt |
| Stromversorgung und Verkabelung | Bereits vorhanden |

### 2.2.4 Anzeige

Für die Anzeige wird das L2C-Display verwendet, das im Arduino-Bausatz von Teko enthalten ist. Auf ihm sollen die live-Messwerte angezeigt werden.

### 2.2.5 Sicherheitsanforderungen

Da wir in diesem Projekt mit Wasser und Strom arbeiten, haben wir dafür gesorgt, dass alle Komponenten entweder mit 12V DC oder mit 5V DC arbeiten. So minimieren wir die Gefahr eines Stromschlages.

### 2.2.6 Terminplan und Meilensteine

#### 2.2.6.1 Termine

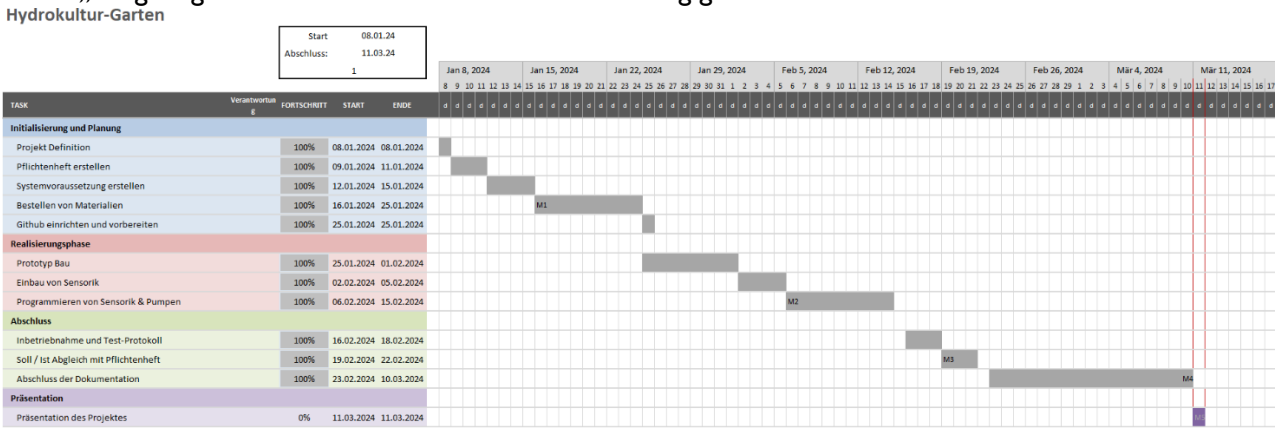
| | |
|---|---|
| Projektstart | 08.01.2024 |
| Abschluss | 11.03.2024 |
| Präsentation | 11.03.2024 |

#### 2.2.6.2 Meilensteine

| |
|---|
| Bestellen von Materialien |
| Abschluss Programmieren von Sensorik und Pumpen |
| Soll / Ist Abgleich und Tests |
| Abschluss der Dokumentation |
| Präsentation |

#### 2.2.6.3 Zeitplan

Nach erfolgreicher Definition des Projekts ist nun die Erstellung eines detaillierten Zeitplans möglich. Dieser Zeitplan dient als Grundlage für die systematische Durchführung der Arbeitsphasen, um die termingerechte Fertigstellung des Projekts zu gewährleisten. Im Zeitplan sind Meilensteine mit dem Präfix „M“ gefolgt von einer fortlaufenden Nummerierung gekennzeichnet.




(Excel Terminplan Vorlage, 2024)

**ABBILDUNG 2: TERMINPLAN**

### 2.2.7 Budget

Wir haben uns persönlich ein Budget von max. 200 Franken gesetzt. Die Komponenten wurden auf Aliexpress bestellt und kosteten jedoch lediglich 170 Franken.

## 2.3 Github

Um die Effizienz der Codebearbeitung zu steigern und eine Versionskontrolle zu implementieren, haben wir ein GIT-Repository eingerichtet und den Code darin hinterlegt.

Das GIT-Repo und den Code dazu kann unter folgendem Link geöffnet werden:
https://github.com/prycannatik/Growbox

# 3 SW-Engineering

## 3.1 Elektronik-Schema

Nachfolgend der Aufbau in Form eines detaillierten Elektronik-Schemas mit Steckerbelegung.

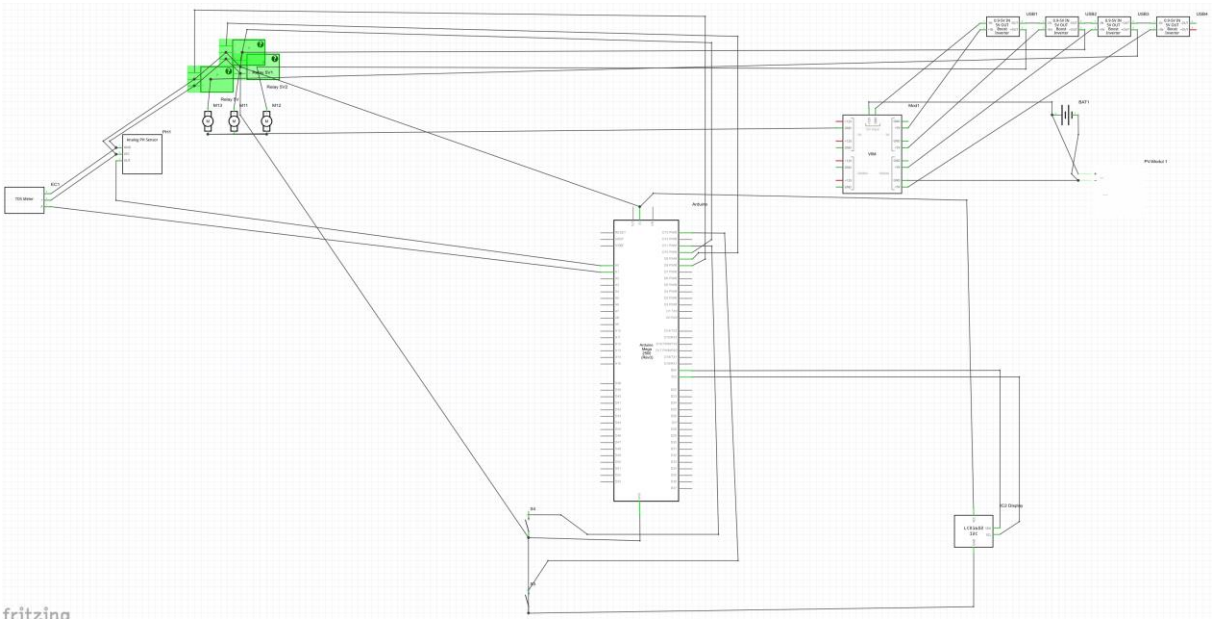



**ABBILDUNG 3: ELEKTRONIK-SCHEMA**

## 3.2 Demo Aufbau

Hier zu sehen ist der dazugehörige Komponentenaufbau mit Verdrahtung.

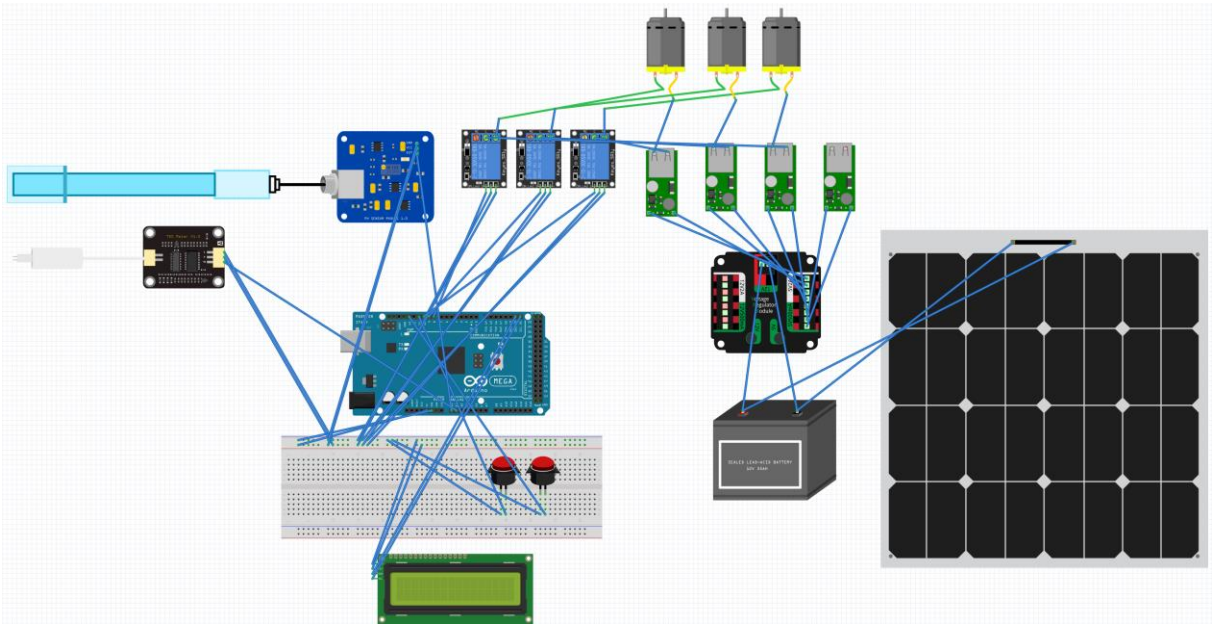


**ABBILDUNG 4: DEMO-AUFBAU**

## 3.3 Überlegungen zur SW-Modularisierung

Wir haben einen strukturierten Ansatz zur Modularisierung der Software verfolgt, um die Codequalität, Lesbarkeit und Skalierbarkeit zu verbessern. Dies beinhaltete die Verwendung einer externen Bibliothek für die Ansteuerung des IC2-Displays, die Aufteilung des Codes in eine Konfigurations-Datei und eine Hauptdatei und die Erstellung von Flussdiagrammen um die Übersichtlichkeit zu gewährleisten.

## 3.4 Fluss-Diagramme sämtlicher SW-Module

### 3.4.1 Übersicht

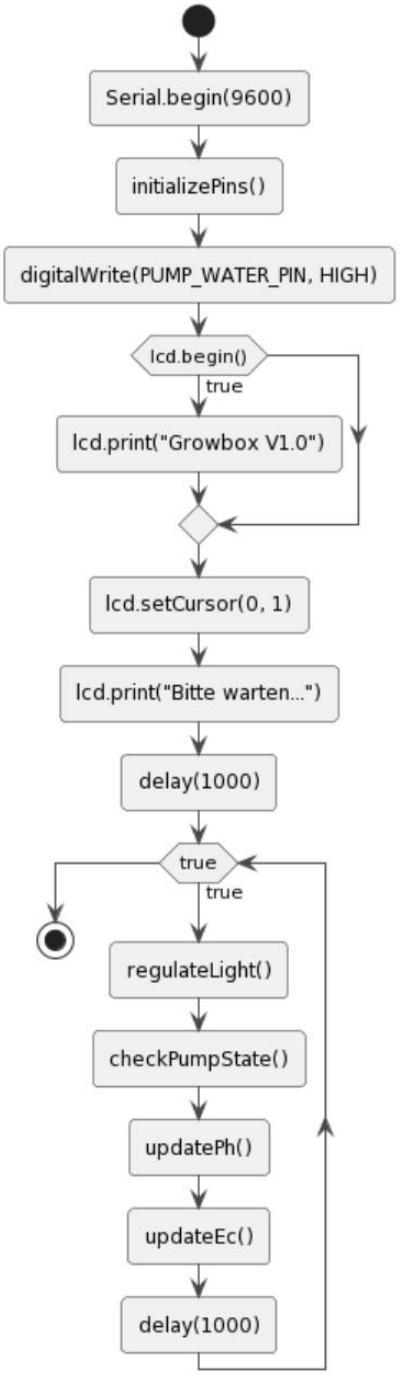



**ABBILDUNG 5: ÜBERSICHT FLUSS DIAGRAMM**

### 3.4.2 Methodendiagramme

#### 3.4.2.1 initializePins

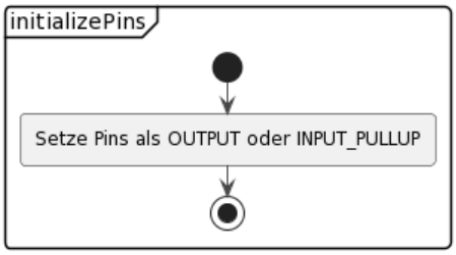



**ABBILDUNG 6: METHODENDIAGRAMM INITIALIZEPINS**

#### 3.4.2.2 regulateLight

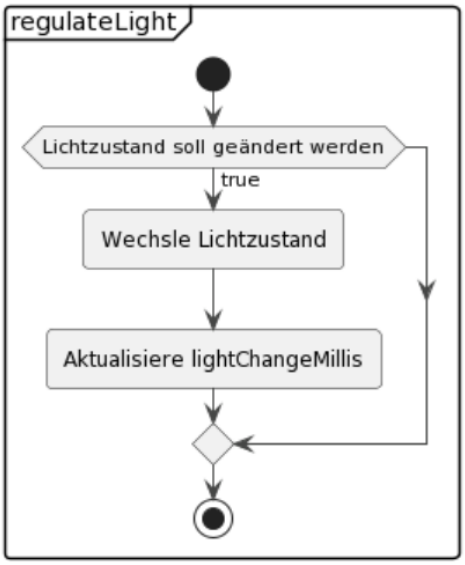



**ABBILDUNG 7: METHODENDIAGRAMM REGULATELIGHT**

#### 3.4.2.3 checkPumpState

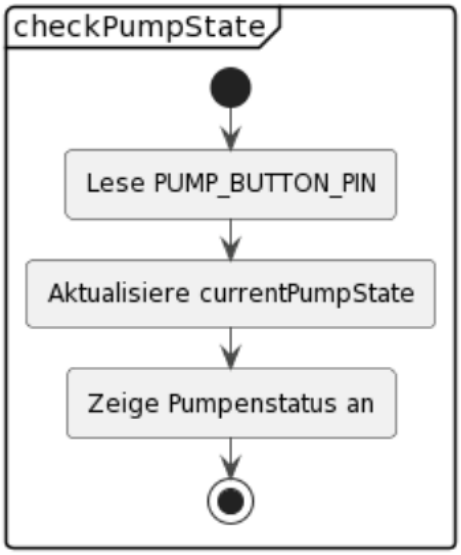



**ABBILDUNG 8: METHODENDIAGRAMM CHECKPUMPSTATE**

#### 3.4.2.4 updatePh

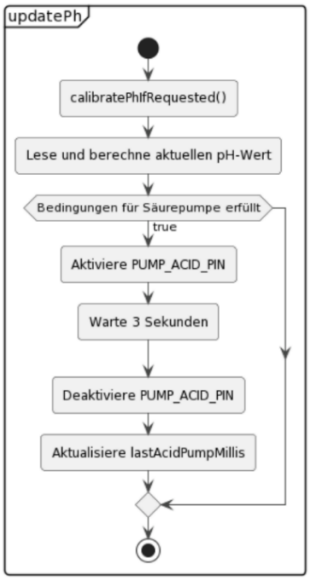



**ABBILDUNG 9: METHODENDIAGRAMM UPDATEPH**

#### 3.4.2.5 calibratePhifRequested

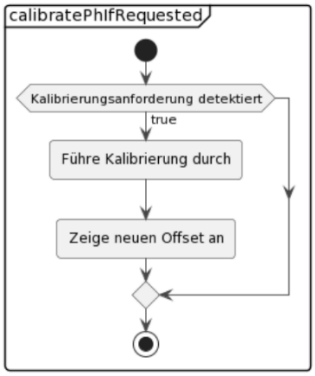



**ABBILDUNG 10: METHODENDIAGRAMM CALIBRATEPHIFREQUESTED**

#### 3.4.2.6 updateEc

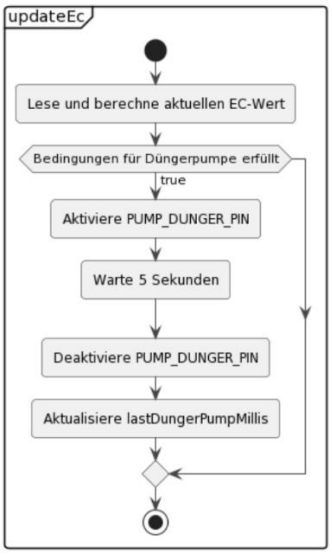



**ABBILDUNG 11: METHODENDIAGRAMM UPDATEEC**

## 3.5 Software-Regeln / Clean-Code

Um die Übersichtlichkeit des Codes zu gewährleisten, insbesondere bei der gemeinsamen Arbeit, haben wir spezifische Software-Regeln definiert, die bei der Programmierung berücksichtigt werden.

| **Software-Regel** | **Beschreibung** |
|---|---|
| Konfiguration | Für die Konfiguration sollte der Übersichtlichkeit halber eine eigene Konfigurations-Datei angelegt werden. |
| Konstanten | Werden in einem config-file deklariert und GROSS geschrieben. |
| Funktionen | Aufgaben die wiederverwendet werden, sollen in Funktionen ausgelagert werden. |
| Variablen | Werden wenn möglich auch im config-file deklariert und klein geschrieben. |
| Kommentare | Nur die wichtigsten Sachen werden kommentiert. Der Code muss so geschrieben werden, dass ein Entwickler diesen auch ohne Kommentare versteht. |

## 3.6 Verwendete 3rd-Party Fragmente

Folgende externe Bibliotheken wurden bei diesem Projekt eingesetzt:

| **Bibliothek-Name / Funktion** | **Zweck** | **Lizenz** | **Quelle** | **Datum** |
|---|---|---|---|---|
| LCDIC2 – Helder Rodrigues | Display | GPL-3.0 | https://www.arduino.cc/reference/en/libraries/lcdic2/ | 11.02.2024 |
| Mapfloat funktion | Berechnung EC-Wert | | https://forum.arduino.cc/t/help-with-ph-sensor-pin-abbreviations/323936/20 | 05.03.2024 |

## 3.7 Besonders ausgefeilte Features

### 3.7.1 IP65 Wasserschutz

Da die Anlage grösstenteils auf dem Balkon stehen wird, wurde ein wasserdichtes Gehäuse gekauft und ein Raster für die Elektronischen Sensoren 3D-gezeichnet und gedruckt.

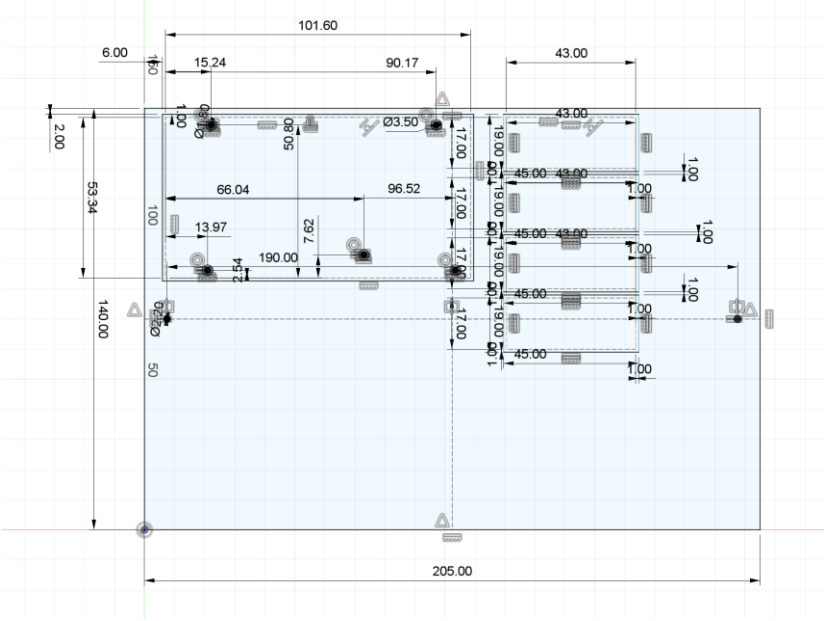



**ABBILDUNG 12: ERSTE SKIZZE DES MASSGESCHNEIDERTEN 3D-DRUCKS**

### 3.7.2 Batteriebetrieben

Die Elektronik der Sensoren wurde so ausgelegt, dass diese unabhängig vom 230V AC-Stromnetz mit einer alten Motorrad Batterie über einige Tage hinweg betrieben werden kann. Zukünftig werden wir eine Solarzelle organisieren, damit der Hydroponik Garten komplett eigenständig funktionieren kann.

### 3.7.3 Kalibrierung PH-Sensor

Der ursprüngliche Plan, 2 Kalibrierungspunkte zu verwenden wurde verworfen, da durch den erstmaligen Testaufbau die Notwendigkeit dessen als unnötig befunden wurde. Im finalen Aufbau wird der PH-Sensor einmalig mittels Referenz Flüssigkeit auf den PH-Wert 7 kalibriert.

**ABBILDUNG 13: KALIBRIERUNGS-VERSUCHE DES PH-SENSORS**

## 3.8 Zusammenbau der Elektronik

Dank des 3D-gezeichneten Elektronikrasters war der Zusammenbau der gesamten Elektronik verhältnismässig einfach. Es mussten jedoch noch Löcher in die wasserfeste Box gebohrt werden. Ausserdem gab es noch einige Anpassungen, bei denen ich nicht genau gemessen hatte oder wo ein Kabel im Weg war und improvisiert werden musste, um einen zweiten 3D-Druck zu sparen.

Die Kabellänge der Kabel im Arduino Kit ist zu lang. Mit einigen Anpassungen und anderer Hardware könnten in der Zukunft einige Kabel eingespart werden.

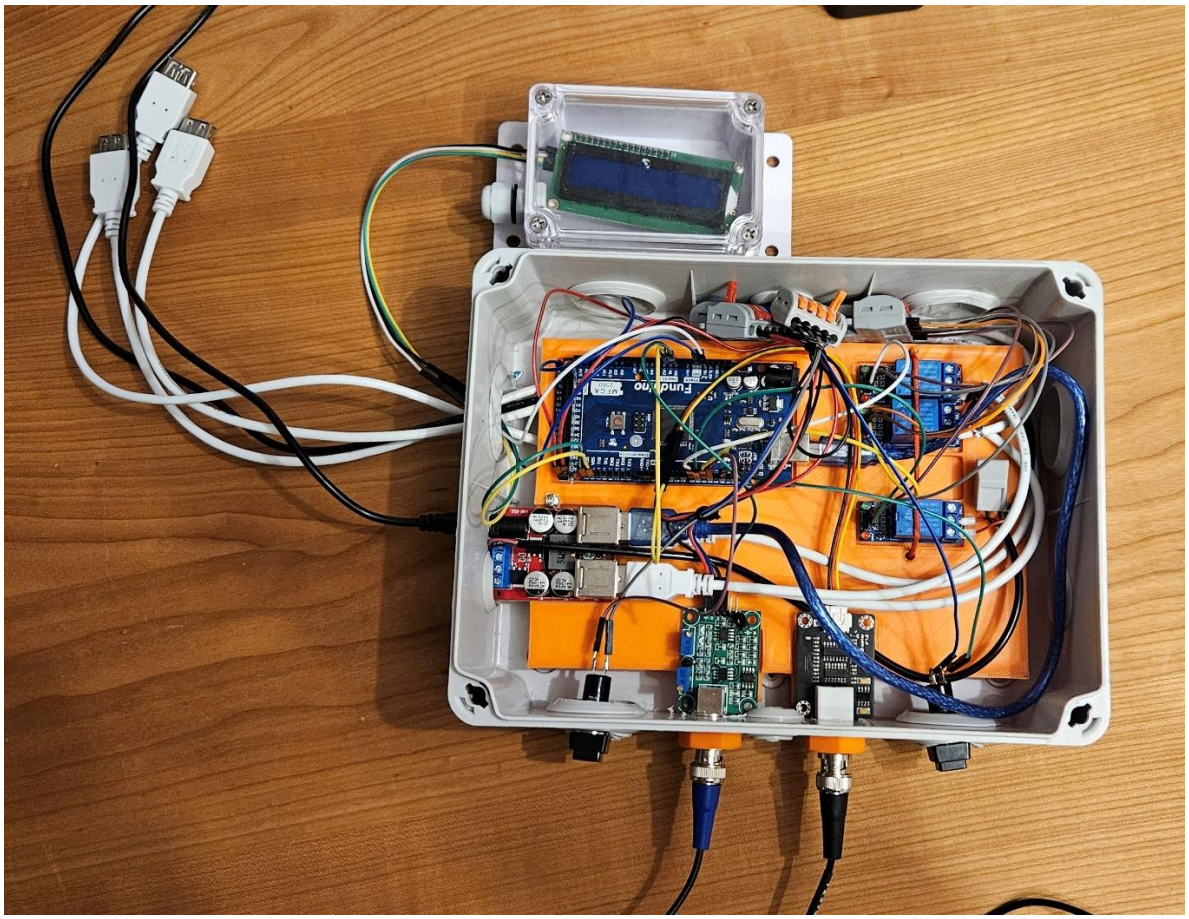


**ABBILDUNG 14: VERKABELUNG / EXTERNE STECKVERBINDUNGEN**

# 4 Aufbau und Konfiguration

Als einen der letzten Schritte haben wir die Zielwerte des Salats für pH und EC überprüft und in der Konfiguration gespeichert. Der Zielwert für den pH-Wert liegt zwischen 5 und 6,2 und der EC-Zielwert sollte etwa 2,6 betragen. Diese Werte wurden in der Konfigurationsdatei gespeichert.

Abschliessend wurde das System gemäss des im letzten Abschnitt definierten Testprotokolls aufgebaut.

**ABBILDUNG 15: FINALER AUFBAU FÜR TESTPROTOKOLL**

# 5 Abschluss

## 5.1.1 Inbetriebnahme- und Testprotokoll für Hydroponik-System

| **Datum der Inbetriebnahme** | **Tester** | **Bemerkungen** | |
|---|---|---|---|
| 05.03.2024 | Paul Vodak / Samuel Haab | Testprotokoll vor der ersten Bepflanzung | |
| | | | |
| *Inbetriebnahme-Checkliste:* | | | |
| | | | |
| **Komponente** | **Überprüft** | **Funktionstest** | **Bemerkungen** |
| Arduino Mikrocontroller | OK | OK | |
| pH-Sensor | OK | Teilweise | EC und PH Wert können nicht gleichzeitig gemessen werden, da EC Wert eine kleine Spannung abgibt. |
| EC-Sensor | OK | Teilweise | |
| Temperatursensor | N/A | N/A | Der Temperatursensor wurde im ersten Schritt komplexitätshalber weggelassen. |
| Wasser-Pumpe | OK | OK | |
| Luft-Pumpe | OK | OK | |
| LED-Wachstumslampen | FAIL | FAIL | Die LED-Wachstumslampen mussten im ersten Schritt weggelassen werden, die die Hydroponics Anlage draussen und nicht wie zuerst angedacht drinnen aufgestellt wurde |
| Relais | OK | OK | |
| Verkabelung und Anschlüsse | OK | OK | Betrieben durch Motorrad Batterie |
| | | | |
| **Funktion** | **Überprüft** | **Ergebnis** | **Bemerkungen** |
| Sensor-Kalibrierung | OK | | 1-Punkt Kalibrierung anstatt wie zuerst angedacht 2-Punkt |
| Nährstoffzufuhr-Steuerung | OK | | |
| Lichtsteuerung | N/A | | |
| Benutzeroberfläche | OK | | |
| Warnsystem | N/A | | Noch kein WIFI Controller |
| | | | |
| Abschliessende Überprüfung | OK | | Bestanden |
| | | | |
| **Kriterium** | **Überprüft** | **Erfüllt** | **Bemerkungen** |
| Dokumentation und Handbuch | OK | OK | Erstellt |

### 5.1.2 Soll-/IST Abgleich mit Pflichtenheft

#### 5.1.2.1 Zielsetzung

Nachfolgend die Auswertung der Zielsetzung, welche im Pflichtenheft definiert wurden.

| Soll-Ziele | Dringlichkeit | Ist-Status |
|---|---|---|
| Rohre zur Anpflanzung von Kräutern/Salate | Muss | OK |
| Automatische Anpassung der Wasser- und Nährstoffzufuhr basierend auf den Sensorwerten | Muss | OK |
| Automatisierte Wasseraufbereitung mit Sauerstoff | Soll | OK |
| Überwachung pH-Wert | Muss | PH-Sensor kann nicht permanent im Wasser gelassen werden |
| Überwachung Nährstoffkonzentration (EC-Wert) | Muss | OK |
| Überwachung Wassertemperatur | Soll | Pendent – Wird in der V2.0 umgesetzt |
| Benutzerfreundliche Überwachung und Steuerung der Anlage | Soll | OK – Umsetzung von MQTT in V2.0 |
| Regelung der Wachstumslampen nach einem vorgegebenen Zeitplan. | Soll | Nicht erreicht – Aufgrund von aussen anstatt innen-Aufbau |

#### 5.1.2.2 System Komponenten

Im Folgenden noch eine Übersicht über die System-Komponenten. Diese sind alle rechtzeitig eingetroffen.

| Element | Bestellstatus | Status |
|---|---|---|
| Arduino Mikrocontroller | Im Kit vorhanden | OK |
| pH-Sensor zur Messung des Säuregehalts | Bestellt | OK |
| EC-Sensor zur Überwachung der Nährstoffkonzentration | Bestellt | OK |
| Temperatursensor für das Wasser | Bestellt | OK |
| Wasser- und Luft-Pumpen | Bestellt | OK |
| LED-Wachstumslampen | Bestellt – Demo | OK |
| Display | Im Kit vorhanden | OK |
| Relais zur Steuerung von Pumpen und Lampen | Im Kit vorhanden | OK |
| Verschiedene Grössen von Schläuchen und Verbindungsstücken | Bestellt | OK |
| Stromversorgung und Verkabelung | Bereits vorhanden | OK |

#### 5.1.2.3 Meilensteine

Alle im Pflichtenheft definierten Meilensteine wurden termingerecht erreicht.

## 5.2 Fazit/Ausblick

Die Projektarbeit war eine spannende Herausforderung, die nicht nur unsere technischen Fähigkeiten, sondern auch unser allgemeines Wissen über die Pflanzenzucht im Wasser verbessert hat.

Mit Hilfe eines Arduino-Mikrocontrollers konnten wir ein automatisiertes Hydrokultursystem entwickeln, mit dem wir das Wachstum der Pflanzen effizient steuern und überwachen konnten.

Die Zusammenarbeit zwischen Samuel Haab und Paul Vodak wurde durch die Verwendung von GitHub für die Codeverwaltung erleichtert.

Durch die sorgfältige Auswahl der Hardwarekomponenten konnten wir das Projekt verhältnismässig reibungslos durchführen und innerhalb des vorgegebenen Zeit- und Budgetrahmens abschliessen.

Da die Projektdauer für einen ausgewachsenen Salat nicht ausreichte, war für uns das Inbetriebnahme- und Testprotokoll, mit dem der Zustand und die Funktionalität des Systems überprüft werden konnte, einer der wichtigsten Punkte.

Das Protokoll stellte sicher, dass alle Anforderungen erfüllt wurden, und trug zum erfolgreichen Abschluss des Projekts bei.

Ein Problem, das noch gelöst werden muss, ist, dass der pH-Wert und der EC-Wert nicht gleichzeitig gemessen werden können. Der EC-Sensor gibt eine Spannung an das Wasser ab, wodurch eine gleichzeitige Messung unmöglich ist. An einer möglichen Lösung haben wir aber bereits angefangen zu arbeite. Und zwar verwenden wir nun auch da geschaltete Pins für PH und EC VCC, um abwechselnd eine Spannung an den EC- und dann an den pH-Sensor anzulegen. Damit es jedoch einwandfrei funktioniert, muss hier in Zukunft noch ein Pull-Down Widerstand eingebaut werden.

Trotz dieser Herausforderungen konnten wir die meisten unserer Ziele erreichen und wertvolle Erfahrung in der Mikroprozessortechnik gewinnen.

Für die Zukunft planen wir, das System weiterzuentwickeln, indem wir die Überwachung der Wassertemperatur implementieren und die Benutzerfreundlichkeit durch die Integration von MQTT verbessern.

# 6 Abbildungsverzeichnis

# 7 Anhang

## 7.1 Arduino_config.h

```
#ifndef ARDUINO_CONFIG_H
#define ARDUINO_CONFIG_H

// Pin definitions
const int PH_PIN = A0;
const int PH_POWER = 5;
const int EC_PIN = A1;
const int EC_POWER = 6;

const int LIGHT_RELAY_PIN = 7;
const int PUMP_DUNGER_PIN = 8;
const int PUMP_WATER_PIN = 9;
const int PUMP_ACID_PIN = 10;
const int PH_CALIBRATION_BUTTON_PIN = 11;
const int PUMP_BUTTON_PIN = 13;

// Target values for lettuce
const float TARGET_EC = 2.60;  // in mS/cm
const float TARGET_PH_LOW = 5.0;
const float TARGET_PH_HIGH = 6.2;

// PH Calibration Offset
float phCalibrationOffset = 0;
float currentPhValue = 7;
float ph7ReadValue = 665;
float ph4ReadValue = 765;


// Light cycle in hours
const long LIGHT_ON_HOURS = 16;
const long LIGHT_OFF_HOURS = 8;
unsigned long lightChangeMillis = 0;
bool lightState = false;

//Button States
bool lastPhCalibrationButtonState = digitalRead(PH_CALIBRATION_BUTTON_PIN);
bool currentPhCalibrationButtonState = digitalRead(PH_CALIBRATION_BUTTON_PIN);

bool pumpsInactiveState = digitalRead(PUMP_BUTTON_PIN);
bool currentPumpState = false;

#endif
```

## 7.2 Main.ino

```
#include "arduino_config.h"
#include "LCDIC2.h"

LCDIC2 lcd(0x27, 16, 2);

void initializePins() {
  pinMode(LIGHT_RELAY_PIN, OUTPUT);
  pinMode(PUMP_DUNGER_PIN, OUTPUT);
  pinMode(PUMP_WATER_PIN, OUTPUT);
  pinMode(PUMP_ACID_PIN, OUTPUT);
  pinMode(PH_POWER, OUTPUT);
  pinMode(EC_POWER, OUTPUT);
  pinMode(PH_CALIBRATION_BUTTON_PIN, INPUT_PULLUP);
  pinMode(PUMP_BUTTON_PIN, INPUT_PULLUP);
}

void setup() {
  Serial.begin(9600);
  initializePins();
  digitalWrite(PUMP_WATER_PIN, HIGH);  // Wasser soll immer eingeschaltet sein
  if (lcd.begin()) lcd.print("Growbox V1.0");
  lcd.setCursor(0, 1);
  lcd.print("Bitte warten...");
  delay(1000);
}

void loop() {
  regulateLight();
  checkPumpState();
  updatePh();
  updateEc();
  delay(1000);
}

void checkPumpState() {
  currentPumpState = (digitalRead(PUMP_BUTTON_PIN) == pumpsInactiveState) ? false : true;
  String supplyPumpStatusText = currentPumpState ? "Pumpen aktiv     " : "Pumpen
inaktiv   ";
  Serial.println(supplyPumpStatusText);
  lcd.setCursor(0, 1);
  lcd.print(supplyPumpStatusText);
}

// pH calibration function
void calibratePhIfRequested() {
  int calibrationLiquidValue = 7;
  currentPhCalibrationButtonState = digitalRead(PH_CALIBRATION_BUTTON_PIN);
  if ((lastPhCalibrationButtonState == HIGH && currentPhCalibrationButtonState == LOW) ||
(lastPhCalibrationButtonState == LOW && currentPhCalibrationButtonState == HIGH)) {
    lcd.setCursor(0, 0);
    lcd.print("Kalibrierung...    ");
    delay(1000);

    phCalibrationOffset = 7 - (mapfloat(analogRead(PH_PIN), ph4ReadValue, ph7ReadValue,
4.01, 7.01));
    ph7ReadValue = analogRead(PH_PIN);

    lcd.setCursor(0, 0);
    lcd.print("Offset: ");
    lcd.print(String(phCalibrationOffset));
    delay(1000);
  }

  lastPhCalibrationButtonState = currentPhCalibrationButtonState;
}
```

```
// pH regulation function
void updatePh() {
  digitalWrite(EC_POWER, LOW); //PH und EC dürfen nie gleichzeitig strom haben
  delay(1000);
  digitalWrite(PH_POWER, HIGH);
  delay(1000);
  calibratePhIfRequested();
  currentPhValue = mapfloat(analogRead(PH_PIN), ph4ReadValue, ph7ReadValue, 4.01, 7.01);

  lcd.setCursor(0, 0);
  lcd.print("PH ");
  lcd.print(String(currentPhValue, 2));
  lcd.print(" ");
  Serial.print("PH Wert: ");
  Serial.println(currentPhValue, 2);

  static unsigned long lastAcidPumpMillis = 0;
  if (currentPumpState && currentPhValue > TARGET_PH_HIGH && millis() - lastAcidPumpMillis
> 60000) {  // 60 seconds delay
    digitalWrite(PUMP_ACID_PIN, HIGH);
    delay(3000);
    digitalWrite(PUMP_ACID_PIN, LOW);
    lastAcidPumpMillis = millis();
  }
}

//Manuell Gemessene Werte umwandeln
float mapfloat(long x, long in_min, long in_max, float out_min, float out_max) {
  return (float)(x - in_min) * (out_max - out_min) / (float)(in_max - in_min) + out_min;
}


void updateEc() {
  digitalWrite(PH_POWER, LOW);
  delay(1000);
  digitalWrite(EC_POWER, HIGH);
  delay(1000);
  int sensorValueEC = analogRead(EC_PIN);
  float voltageEC = sensorValueEC * (5.0 / 1023.0);
  float EC = voltageEC;

  lcd.print("EC: ");
  lcd.print(String(EC, 2));
  lcd.print("    ");
  Serial.print("EC Wert: ");
  Serial.println(EC, 2);

static unsigned long lastDungerPumpMillis = 0;
 Serial.println((millis() - lastDungerPumpMillis));
  if (currentPumpState && EC < TARGET_EC && (millis() - lastDungerPumpMillis) > 60000) {
    digitalWrite(PUMP_DUNGER_PIN, HIGH);
    delay(5000);
    digitalWrite(PUMP_DUNGER_PIN, LOW);
    lastDungerPumpMillis = millis();
  }
}

void regulateLight() {
  if ((lightState && millis() - lightChangeMillis >= LIGHT_ON_HOURS * 3600000UL) ||
(!lightState && millis() - lightChangeMillis >= LIGHT_OFF_HOURS * 3600000UL)) {
    lightState = !lightState;
    digitalWrite(LIGHT_RELAY_PIN, lightState ? HIGH : LOW);
    lightChangeMillis = millis();
  }
}
```