MORRIGAN:
PORTABLE MESSSTATION ZUR LESUNG VON WASSERDATEN AUF BASIS DER RASPBERRY PI PLATTFORM

Leonie Riedel, Michelle Wallmann
331 PK inf/mat 24/25
Februar 28, 2025

Inhaltsverzeichnis

Abstract....4
1. Einleitung....1
1.1. Motivation....1
1.2. Inspiration....1
1.3. Umfang....2
2. Konzeptentwicklung....2
2.1 Wassermessstationen....2
2.2 Citizen Science....3
2.3 ARWAQUTE....3
2.4 Konzeptionierung und Projektmanagement....3
3. Theoretische Grundlagen und praktische Umsetzung....4
3.1 Orientierung....4
3.2 Kommunikation mit dem Pi....5
3.2.1 Netzwerkkommunikation....5
3.2.2 Sensor Kommunikation....7
3.2.3 Umsetzung in Code....9
3.2.3.1 Das Frontend....9
3.2.3.1.1 Daten abrufe....9
3.2.3.1.2 Diagrammerstellung....10
3.2.3.1.3 Datenaktualisierung....10
3.2.3.1.4 Interaktive Steuerung....11
3.2.3.1.5 Bereitstellung des Frontends....11
3.2.3.2 Das Backend....11
3.2.3.2.1 main.py....12
3.2.3.2.2 server.py....13
3.2.3.4 Ausführung beim Hochfahren....14
3.3.1 Theorie....14
3.3.2 Verbindung mit dem Raspberry Pi....15
3.3.3 Code-Implementierung....15
3.4 pH-Wert....17
3.4.1 Theorie....17
3.4.2 Verbindung mit dem Raspberry Pi....18
3.4.3 Kalibrierung....19
3.4.4 Code-Implementierung....20
3.5. Total Dissolved Solids....21
3.5.1 Theorie....21
3.5.2 Verbindung mit dem Raspberry Pi....21
3.5.3 Kalibrierung....22
3.5.4 Code-Implementierung....23
3.6 Dissolved Oxygen....23
3.6.1 Theorie....23
3.6.2 Verbindung mit dem Raspberry Pi....24
3.6.3 Kalibrierung....25
3.6.4 Code-Implementierung....26
3.7 Turbidity....27
3.7.1 Theorie....27

3.7.2 Verbindung mit dem Raspberry Pi.....27
3.7.3 Kalibrierung.....28
3.7.4 Code-Implementierung.....29
5. Ergebnis Evaluation: Vergleich von ARWAQUTE und MORRIGAN.....30
6. Fazit.....31
7. Quellenverzeichnis.....33
8. Anhang.....35

Abstract

Die wachsende Bedeutung empirischer Forschung erfordert kostengünstige und mobile Messsysteme zur Umweltüberwachung. In dieser Arbeit wird eine tragbare Wassermessstation auf Basis des Raspberry Pi entwickelt, die Temperatur, pH-Wert, Trübung, gelösten Sauerstoff und Total Dissolved Solids (TDS) misst. Ziel ist es, bestehende Lösungen in Bezug auf Portabilität, Benutzerfreundlichkeit und Echtzeit-Datenverarbeitung zu optimieren.

Durch den Einsatz eines modularen Sensorkonzepts und eines webbasierten Interfaces können Messdaten in Echtzeit visualisiert und analysiert werden. Erste Tests zeigen, dass die Messstation trotz kostengünstiger Sensoren präzise Daten liefert, wobei Kalibrierung und Fehlerkorrektur eine zentrale Rolle spielen. Die Plattform ermöglicht es, Freiwilligen und Forschern unabhängig von festen Laboren zuverlässige Wasseranalysen durchzuführen.

Zukünftige Erweiterungen könnten eine breitere Sensorpalette, automatische Datenübertragung und eine verbesserte Langzeitstabilität der Messungen umfassen. Die Arbeit leistet damit einen Beitrag zur Open-Source Infrastruktur für Umwelt- und Citizen-Science-Projekte.

1. Einleitung

Diese Profilarbeit sowie die dazugehörige Messstation wurde im Rahmen des Profilkurses Informatik/Mathematik 24/25 der gymnasialen Oberstufe der Carl von Ossietzky Bremerhaven im Zeitraum vom 21.11.2024 bis zum 28.02.2025, von Leonie Riedel und Michelle Wallmann, erarbeitet.

1.1. Motivation

Bereits vor Projektbeginn bestand das Interesse an der Arbeit mit Mikroprozessoren und Single-Board-Computern (SBCs). Besonders fasziniert waren die Projektteilnehmer, davon wie Computer von ihren ursprünglichen, großen Modellen auf die kompakten, kreditkartengroßen Varianten geschrumpft sind, ohne dabei an Leistung zu verlieren. Michelle Wallmann interessierte sich zudem für die Arbeit mit Sensoren, da sie zuvor noch keine Erfahrungen auf diesem Gebiet gesammelt hatte.

1.2. Inspiration

Bevor sich für ein konkretes Thema entschieden wurde, stellte Herr Ehlert, ein Lehrer der gymnasialen Oberstufe des Carl von Ossietzky Schulzentrums Bremerhaven, das Erasmus+ Projekt in Thessaloniki vor. Dort hatte eine griechische Schule, das Second Lyceum of Kalamaria, eine Messstation zur Bestimmung der Trinkbarkeit von Wasser auf Basis der Arduino-Plattform entwickelt, die sie „ARWAQUTE“ nannten. Das Thema weckte unser Interesse, da Michelle Wallmann vor allem an der wasseranalytischen Seite interessiert war und Leonie Riedel der biologische Aspekt einer möglichen Profilarbeit interessierte. Wir stellten uns also die Frage: welche technischen und gestalterischen Verbesserungen sind notwendig, um eine

Wasssermessation portabler und anwenderfreundlich zu gestalten?

1.3. Umfang

Diese Arbeit umfasst die Entwicklung einer tragbaren und kostengünstigen Wassermessstation auf Raspberry Pi-Basis. Der Rahmen wurde wie folgt definiert:

- Portabilität: Die Messstation muss in einen 20L-Rucksack passen, um einen einfachen Transport zu ermöglichen.
- Unabhängiger Betrieb: Ein Netzteil darf nicht erforderlich sein, um in vivo-Messungen zu ermöglichen.
- Kostenlimit: Die Gesamtkosten (exkl. Versand/Zoll) dürfen 350 € nicht überschreiten, da viele einzelne Sensoren teurer sind.

Zusätzlich soll das System benutzerfreundlich sein:

- Automatisierte Sensorsteuerung durch Skripte.
- Einfacher Sensortausch für flexible Nutzung.
- Messwerte abrufbar ohne Geräteöffnung, um Nutzerfehler zu minimieren.

Die Netzwerkgröße ist auf 3 Peers limitiert, um die Systemkomplexität zu reduzieren.

2. Konzeptentwicklung

Die Entwicklung des Messstationskonzepts basierte auf drei Schwerpunkten: bestehende Wassermessstationen, Citizen Science und ARWAQUTE als Referenzprojekt.

2.1 Wassermessstationen

Messstationen sind entscheidend für die Überwachung von Wasserqualität und Umweltveränderungen (Pellerin et al. 2016). Konventionelle Systeme sind jedoch teuer, stationär und oft unflexibel (Fraisl et al. 2022), was sie für Privatpersonen oder kleinere Organisationen unpraktisch macht.

Unser Ziel war es, eine mobile,

kostengünstige Messstation zu entwickeln, die diese Einschränkungen überwindet und dennoch zuverlässige Daten liefert.

2.2 Citizen Science

Citizen Science ermöglicht es der Öffentlichkeit, wissenschaftliche Daten zu sammeln, was die Forschungskapazitäten erheblich erweitert (Ullrich 2024). Während der COVID-19-Pandemie wurde der Nutzen solcher Initiativen besonders deutlich (Rainear 2025).
Allerdings erfordert die Beteiligung von Freiwilligen eine einfache, kosteneffiziente Hardware (Weigelhofer & Pölz 2016). Unsere Arbeit knüpft hier an, indem sie eine niederschwellige Messlösung für Privatpersonen bereitstellt, um Citizen Science in der Wasseranalyse zu unterstützen.

2.3 ARWAQUTE

Das Second Lyceum of Kalamaria entwickelte ARWAQUTE, eine auf Arduino Mega 2560 basierende Messstation für Temperatur, pH, Trübung, TDS und DO. Die Daten wurden per Breadboard-Verkabelung erfasst, auf eine Micro-SD-Karte geschrieben und über ein LCD-Display visuell dargestellt. Die Station war jedoch nicht portabel, nur in vitro einsetzbar und an ein Netzteil gebunden.
Unser Projekt baut auf ARWAQUTE auf, erweitert es durch bessere Portabilität, Kalibrierung und Echtzeit-Webinterface und passt es an moderne Anforderungen an.

2.4 Konzeptionierung und Projektmanagement

Für die Arbeit wurde das Projektmanagement Tool Milanote verwendet um die Arbeitet strukturiert aufzuteilen und dort TODO sowie eine Arbeitsverteilung zu finden.

Um dies zu tun musste sich allerdings erst eine Arbeit finden und dafür brauchten wir ein Konzept. Die Idee war, dass man den Raspberry Pi mithilfe einer

Powerbank in einem 3D gedruckten Gehäuse betreiben könnte und die Software durch moderne Aspekte wie WebUIs und Modularisierung aufbessert.

Die Arbeitsverteilung war dann klar: Michelle war für die theoretischen Grundlagen und das 3D-Modell verantwortlich, während Leonie sich mit der Hardware und der Programmierung des Codes beschäftigte.

3. Theoretische Grundlagen und praktische Umsetzung

3.1 Orientierung

Der erste Schritt des Projektes war eine Analyse von ARWAQUTE, um herauszufinden, wie dieses Projekt erweitert werden kann und ob diese Erweiterungen sinnvoll für unseren Anwendungsbereich geeignet sind. Basierend auf dieser Analyse wurde die Messstation in drei wesentliche Bereiche unterteilt, die eine Erweiterung oder Verbesserung ermöglichen,

- Die Sensoren selbst, also was für Messparameter die Messstation auswertet,
- Die Hardware der Station, e.g. wie sie mit Strom versorgt wird und auf welchem Prozessor sie läuft, die beeinflusst wie tragbar, leistungsstark und wie effizient die Station ist,
- Die Software, wie z.B. die Interfaces über die die Daten ausgegeben werden, die hauptsächlich die Benutzerfreundlichkeit beeinflussten.

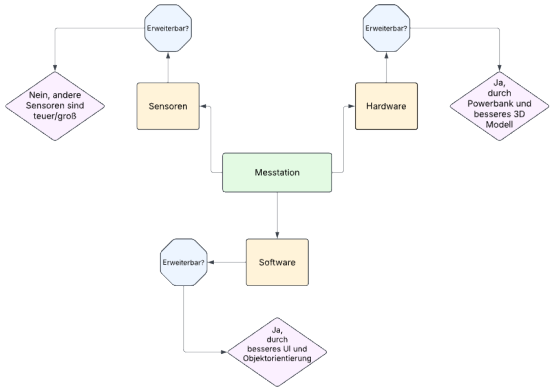


Abbildung 1: Erweiterbarkeit der Messstation

Die Sensoren wurden evaluiert indem betrachtet wurde, ob es andere Sensoren für grundlegende Wasserwerte wie Phosphat und Nitrat für den Raspberry Pi gab und ob diese in das Projektkonzept integriert werden können. Dabei stellte

sich heraus, dass viele dieser Sensoren entweder sehr teuer oder unportabel sind, was das kostengünstige und tragbare Konzept des Projekts nicht erfüllt hätte. Als Beispiel kann man den “Phosphate Sensor” von Dartmouth Oceans Technologies Inc. nehmen, dieser wiegt 5 kg in der Luft (2025 Dartmouth Ocean) was die Tragbarkeit deutlich eingeschränkt hätte.

Der Faktor der Hardware wurde evaluiert, indem betrachtet wurde, wie Erweiterbar die originale Hardware des Projektes „ARWAQUTE” ist und welche Schwächen diese hatte. Ein wesentlicher Nachteil war, dass ARWAQUTE nur über ein festes Netzteil betrieben werden konnte, was in vivo[1] Messungen quasi unmöglich gemacht hatte. Zur Verbesserung wurde die Hardware auf eine mobile, batteriebetriebene Lösung gewechselt und zudem konnte das 3D Modell optimiert werden, um die Tragbarkeit zu verbessern.

Der Software Aspekt der Messstation war einer, der stark verbesserbar war. Der ursprüngliche ARWAQUTE Quellcode war in einer einzigen Datei und die Messergebnisse wurden auf einem LCD-Bildschirm angezeigt. Um dies benutzerfreundlicher zu gestalteten wurde, ein objektorientierter Ansatz verfolgt, der die Entwicklung eines Web-Interfaces verfolgte. Dieses sollte Endnutzern erlauben einfach, die Messdaten in einer grafischen Oberfläche zu visualisieren und zu interpretieren.

3.2 Kommunikation mit dem Pi

3.2.1 Netzwerkkommunikation

Die Auslesung die per LCD die in „ARWAQUTE” stattfand, war unserer Meinung nach nicht sehr Nutzerfreundlich, genauso wenig, dass man Daten nur dann auslesen kann wenn man die Micro-SD

[1] in vivo: lat. „im Lebendigen”, d.h. dass Messungen im Lebendigen Organismus genommen werden

Karte manuell entfernt. Wir empfanden dies als wenig benutzerfreundlich, da man physische Nähe zur Station brauchte und da man annehmen kann, dass der Anfänger vll. Angst hätten die Micro-SD Karte zu entfernen. Daher hatten wir uns für ein anderes System entschieden, der Pi sollte einen Mobilen Hotspot starten. Auf diesem lokalen Netzwerk macht der Pi eine Website auf Port 8000 verfügbar, auf der die Daten der Sensoren in Echtzeit auf Graphen dargestellt werden. Die Website soll auch die Möglichkeit geben, den Pi zu kontrollieren. Im Notfall kann sich ein Endnutzer auch per SSH mit dem Pi verbinden, um ihn zu steuern. Aufgrund dieser vernetzten und Echtzeit Natur wurde sich dazu entschieden den Pi MORRIGAN zu nennen, welches für **M**obile **O**ptimized **R**eal-Time **R**esource **I**nterface **G**auge **A**nalysis **N**etwork steht.

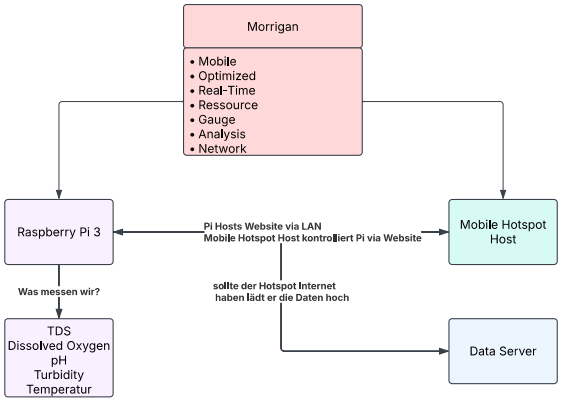


Abbildung 2: Konzept der Kommunikation

Die Umsetzung dieses Konzeptes ist wie folgt:

Die von den Sensoren gelesenen Daten sollen an zwei Orten gespeichert werden, zum einen sollen sie in eine .json Datei gespeichert werden die auf dem WebUI als Graph dargestellt werden und andererseits sollen die Dateien langfristig in einer SQLite Datenbank gespeichert werden. Um die Benutzerfreundlichkeit des WebUI's möglichst hoch zu halten hat das WebUI drei simple Knöpfe, einen um die Graphen neu zu starten, indem es die .json Datei leert, der 2. soll die Messungen starten indem er über ein Flask Backend die Datei zum Auswerten der Sensoren ausführt, der 3. dient dazu die Messungen zu stoppen indem er dem

Backend den Befehl gibt die Datei zu stoppen die die Sensoren auswertet. Darunter sind die einzelnen Graphen für die Sensoren.

3.2.2 Sensor Kommunikation

Sensoren sind unterteilt in digitale und analoge Sensoren. Digitale Sensoren geben ihre Daten als Bits aus d.h. sie sind frequenzmoduliert. Währenddessen geben analoge Sensoren ihre Werte als eine Spannung aus. Sie sind also amplitudenmoduliert.

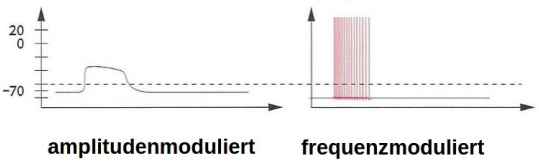


Abbildung 3: amplitudenmoduliert vs. frequenzmoduliert

Die Spannung die ein Sensor maximal ausgibt ist dabei Referenzspannung, in der die Amplitude am höchsten ist. Da der Raspberry Pi allerdings keine analogen Anschlüsse, auch genannt Pins oder Headers, hat braucht das Projekt einen sogenannten Analog-Digital-Wandler (ADC[2]). Dieses Projekt benutzt den ADS1115, welcher über einen 16bit ADC verfügt, d.h, dass die eingegebene Spannung mit einer Auflösung von 16 bits zu einem Byte umgerechnet werden. Des weiteren verfügt er über einen eingebauten 4 mal Multiplexer, d.h. das wir 4 analoge Sensoren anschließen können. Er kommuniziert über das I²C Protokoll und man benötigt die adafruit_ads1x15 um ihn in Python zu benutzen.

Die Standard Schaltung, per Breadboard, für diesen an den Raspberry Pi sieht aus wie folgt

[2] ADC: Analog-Digital-Converter engl. Analog-Digital-Wandler

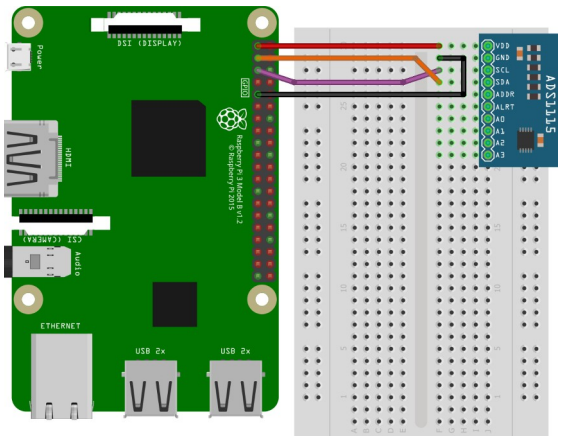


Abbildung 4: Schaltung ADS1115 an Pi

Diese Schaltung wird sehr kompliziert wenn man viele Sensoren an das Breadboard schließt, wie man in [TODO] sehen kann und die einzelnen extra Kabel machen es sehr unfreundlich für Anfänger die Station zu benutzen.

Um die Schaltung von analogen Sensoren zu vereinfachen und Kabelverbindungen zu reduzieren wurde eine Pi-HAT-Platine entwickelt. Diese ermöglicht es den ADS1115 in einen 10 Pin Header zu verbinden und daraufhin 3 Pin analoge Sensoren einfach in einen der 4 3 Pin Header zu stecken die mit dem ADS1115 verbunden sind.

Das Platinen Design sieht dafür wie folgt aus

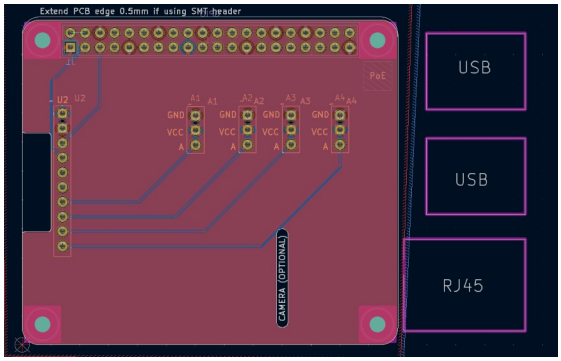


Abbildung 5: PCB für den Pi-HAT

Die Platine hat eine Kupferebene für die VCC 3V3 Schaltungen und eine Kupferebene für die GND Schaltungen. Über jedem analogen Pin ist eine Nummer um zu vereinfachen welchen analogen Pin man beim Programmieren angeben muss. Die Nummer die man beim Programmieren angeben muss ist die n-1 zu der Nummer die der Anschluss hat.

Der Schaltplan dafür ist wie folgt

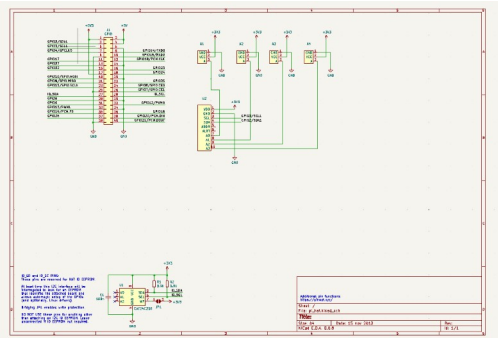


Abbildung 6: Schaltplan für Pi-HAT

Diese beiden kann man nun verändert in Fritzing umsetzen, hierbei

wird die logische Schaltung mithilfe von Pin Headern und Kabeln nachgestellt, um in den folgenden Teilen dieser Arbeit vereinfacht die Funktion der Sensor Schaltpläne darzustellen.

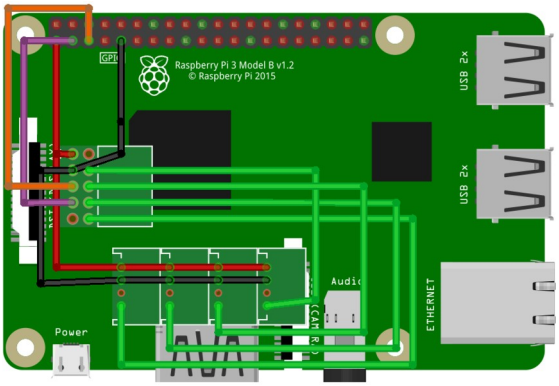


Abbildung 7: Pi-HAT in Fritzing

Dieser HAT ermöglicht es nun einfach Sensoren direkt mit dem Pi durch ihr mitgeliefertes Kabel zu verbinden, anstatt durch mehrere Jumper Kabel und ein Breadboard die oft zu losen Kabeln und extra Gewicht führen. Das PCB Design kann dann von einem PCB Hersteller hergestellt und geliefert werden. Im Rahmen der Arbeit wurde PCBWay als Hersteller gewählt welche die Platinen für 4,78€ herstellten.

3.2.3 Umsetzung in Code

3.2.3.1 Das Frontend

Das Frontend der Messstation wurde als Webinterface entwickelt, um Sensordaten in Echtzeit zu visualisieren. Die Umsetzung erfolgte mit HTML, CSS JavaScript und der Chart.js-Bibliothek. Die Messwerte werden regelmäßig aus einer JSON-Datei (data.json) geladen und in Diagrammen dargestellt.

Die Hauptfunktionen des Frontend sind folgendermaßen

3.2.3.1.1 Daten abrufe

Die Sensordaten werden mit fetch() aus einer JSON-Datei abgerufen. Falls die Datei nicht verfügbar ist, wird ein Fehler ausgegeben.

```
async function fetchData() {
  try {
    updateCharts(await (await
fetch("data.json")).json());
  } catch (e)
{ console.error(e); }
}
```

Die Funktion wird alle 10

Sekunden aufgerufen, um die Messwerte regelmäßig zu aktualisieren.

3.2.3.1.2 Diagrammerstellung

Die Sensordaten werden über die Chart.js-Bibliothek in Diagrammen dargestellt. Jedes Diagramm wird dynamisch für einen bestimmten Sensortyp erzeugt.

```
function createChart(ctx,
label, color, yLabel) {
  return new Chart(ctx, {
    type: "line",
    data: { labels: [],
datasets: [{ label, data: [],
borderColor: color,
backgroundColor:
color.replace("1)", "0.2)"),
borderWidth: 1 }] },
    options: { responsive:
true, plugins: { legend:
{ position: "top" }, title: {
display: true, text:
label } }, scales: { y:
{ title: { text: yLabel } },
x: { min: 0, max:
chartXDisplayMaxAmount,
title: { text:
chartXLabel } } },
maintainAspectRatio: false,
animation: enableAnimations }
  });
}
```

3.2.3.1.3 Datenaktualisierung

Die geladenen Daten werden in die bestehenden Diagramme eingefügt. Falls das Diagramm noch nicht existiert, wird es neu erstellt.

```
function updateCharts(data) {
 const sensorMapping =
{ temperature: "chartTemp",
PH: "chartPH", TDS:
"chartTDS", DO: "chartDO",
turbidity: "chartTurb", EC:
"chartEC" };


Object.entries(sensorMapping).
forEach(([key, chartId]) => {
 const sensorData =
data.filter((item) => key in
item);
 if (!charts[chartId]) {
 charts[chartId] =
createChart(document.getElemen
tById(chartId).getContext("2d"
), key, "rgba(54, 162, 235,
1)", key);
 }
 charts[chartId].data.labels =
sensorData.map((row) =>
row.seconds);

charts[chartId].data.datasets[
0].data = sensorData.map((row)
=> row[key]);

charts[chartId].update();
```

```
    });
}
```

Die Diagramme werden aktualisiert, indem neue Zeitstempel (seconds) als X-Werte und die jeweiligen Sensorwerte als Y-Werte eingetragen werden.

3.2.3.1.4 Interaktive Steuerung

Nutzer können über Buttons zwischen der Ansicht aller Datenpunkte und einer fokussierten Darstellung wechseln.

```
document.getElementById("viewAllPoints").addEventListener("click", () => {
 viewAllPoints = true;
 fetchData();
});

document.getElementById("viewSection").addEventListener("click", () => {
 viewAllPoints = false;
 fetchData();
});
```

Damit kann die Skalierung der X-Achse (Zeitachse) angepasst werden.

3.2.3.1.5 Bereitstellung des Frontends

Das Webinterface wird über einen lokalen Webserver gehostet, der mit dem folgenden Befehl gestartet wird:

python3 -m http.server 8000

Der Raspberry Pi stellt über seinen mobilen Hotspot die Webanwendung zur Verfügung, sodass die Sensordaten von jedem Gerät im Netzwerk eingesehen werden können.

3.2.3.2 Das Backend

Das Backend ist in 2 Dateien geteilt, um es modularer zu gestalten. Zum einem gibt es main.py welches zur Verarbeitung der Daten dient. Es liest die Sensoren über das ADS1115 Modul aus und speichert die Daten in einer SQLite-Datenbank sowie einer JSON-Datei zur direkten Visualisierung. Zum anderen gibt es server.py welches eine Flask-API mit der Flask und flask-cors library bereitstellt. Diese erlaubt das starten der Messungen durch eine HTTP-GET Anfrage.

3.2.3.2.1 main.py

main.py ist für die Erfassung der Sensordaten Verantwortlich. Beim starten des Skripts wird erst eine Datenbank mit dem folgenden Ausschnitt initialisiert sollte diese noch nicht existieren.

```
def init_db():
 conn =
sqlite3.connect(DB_PATH)
 cursor = conn.cursor()
 cursor.execute('''
 CREATE TABLE IF NOT EXISTS
sensor_logs (
 id INTEGER PRIMARY KEY
AUTOINCREMENT,
 timestamp TEXT,
 sensor TEXT,
 value REAL,
            voltage REAL,
            elapsed_seconds
INTEGER
        )
    ''')
    conn.commit()
    conn.close()
```

Diese Dient der Langzeit Speicherung der Daten um die Daten nach längerer Zeit nachvollziehen zu können. Die Messwerte werden also mit einem Zeitstempel, der Spannung und dem gemessenen Wert gespeichert.

Um zwischen den verschiedenen Sensoren zu wechseln wird auf die Multiplexing Funktion des ADS1115 zurückgegriffen. Mithilfe der switch_sensor Funktion wurde dieses dann in Code implementiert.

```
def switch_sensor(sensor):
 mapping = {
 "ph": ADS.P3, "TDS": ADS.P0,
"turbidity": ADS.P1, "DO":
ADS.P2
 }
 if sensor in mapping:
 ads.gain = 1
 return AnalogIn(ads,
mapping[sensor])
 else:
 raise ValueError("Ungültiger
Sensortyp")
```

Die Messungen werden daraufhin in einer Schleife durchgeführt, wobei Temperatur zuerst erfasst wird, da manche Sensorwerte Temperatur abhängig sind.

```
while True:
 temp = read_temp()
 log_data("temperature",
temp, None)

 for sensor in ["ph", "TDS",
"turbidity", "DO"]:
 chan = switch_sensor(sensor)
```

```
 value = globals()
[f"read_{sensor.upper()}"]
(chan.voltage)
 log_data(sensor, value,
chan.voltage)
 time.sleep(1)
```

Jeder Messwert wird sowohl in einer JSON Datei für das Webinterface als auch in der SQLite Datenbank gespeichert.

```
def log_data(sensor, value,
voltage):
    ts, elapsed =
datetime.now().isoformat(),
int(time.time() - start_time)
    data = {"seconds":
elapsed, "timestamp": ts,
"voltage": voltage, sensor:
value}
    json.dump(data,
open(DATA_FILE_PATH, "w"),
indent=4)

    with
sqlite3.connect(DB_PATH) as
conn:
        conn.execute("INSERT
INTO sensor_logs VALUES
(NULL, ?, ?, ?, ?, ?)",
tuple(data.values()))
```

3.2.3.2.2 server.py

server.py startet eine simple Web API, über die das Hauptskript per HTTP gestartet wird.

```
from flask import Flask,
jsonify
from flask_cors import CORS
import subprocess, os

app = Flask(__name__)
CORS(app)

SCRIPT_PATH =
os.path.join(os.path.dirname(
__file__), "main.py")

@app.route('/run-script',
methods=['GET'])
def run_script():
    try:
        result =
subprocess.run(["python",
SCRIPT_PATH],
capture_output=True,
text=True)
        return
jsonify({"output":
result.stdout.strip(),
"error":
result.stderr.strip()})
    except Exception as e:
        return
jsonify({"error": str(e)}),
500

if __name__ == '__main__':
    app.run(host='0.0.0.0',
port=5000, debug=True)
```

3.2.3.4 Ausführung beim Hochfahren

Der Mobile Hotspot des Pis konnte automatisch gestartet werden indem man die commands

sudo systemctl enable hostapd

und

sudo systemctl enable dnsmasq

während die Python Skripts beim Hochfahren mithilfe von crontab ausgeführt wurden. Dazu wurde mit dem command crontab -e die folgenden Linien hinzugefügt:

@reboot /usr/bin/python3 /home/water-pi/MORRIGAN/src/backend/server/main.py &

@reboot /usr/bin/python3 /home/water-pi/MORRIGAN/src/backend/server/server.py &

@reboot cd MORRIGAN/src/frontend/charts

python3 -m http.server

3.3 Temperatur

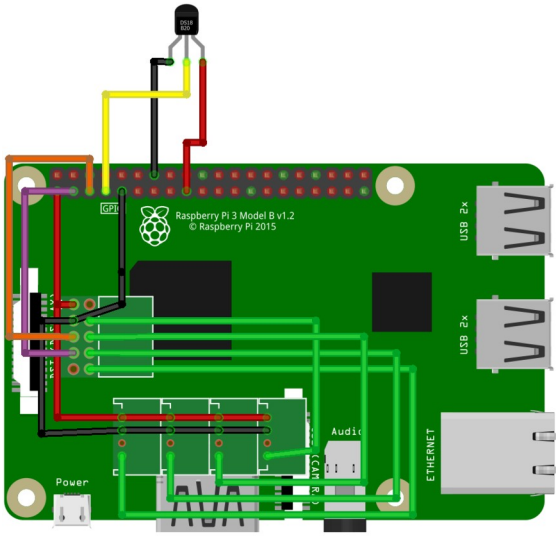
3.3.1 Theorie

Die Wassertemperatur hat einen direkten Einfluss auf chemische und biologische Prozesse in Gewässern. Beispielsweise bildet das Wasser, ab 4°C eine Eisschicht auf der Oberfläche, was dazu führt, dass Lebewesen ihren Wohnort wechseln müssen (FWU Institut 2025). Des weiteren nimmt die Löslichkeit von Sauerstoff mit steigender Temperatur ab, was den gelösten Sauerstoffgehalt reduziert und somit aquatische Ökosysteme beeinflusst (StudySmarter GmbH 2025). Da die meisten Fische nur in einem bestimmten Temperaturbereich von 10°C bis 20°C leben können, sterben sie außerhalb dieser Temperaturzone. Bei zu kaltem Wasser erfrieren die Lebewesen und bei zu hohen Temperaturen ersticken die Lebewesen, da zu viel Sauerstoff freigesetzt wird und sie ohne Sauerstoff

nicht atmen können (SEAWATER CUBE).

3.3.2 Verbindung mit dem Raspberry Pi

Um die Temperatur mithilfe des Raspberry Pi zu messen, benutzen wir das „Waterproof DS18B20 Temperature Sensor Kit“ von Gravity, dieses kostet 7.09€ bei DFRobot. Dieser ist bereits kalibriert und kommuniziert digital über das 1-Wire-Protocol und kann von -55 bis 125°C mit einer Genauigkeit von ±0.5°C Daten aufnehmen. Der Sensor hat eine eigene 64 bit-ID. Um den Sensor physisch zu verbinden verknüpfen wir den Sensor mit dem gemeinsamen GND und dem gemeinsamen VCC → 3V3, der letzte Pin DATA wird mit dem Pin GPIO 4 als GPCLK0 (general purpose clock 0) oder auch 1-Wire-Port verbunden. Das 1-Wire-Protocol muss dann in raspi-config eingeschaltet werden.



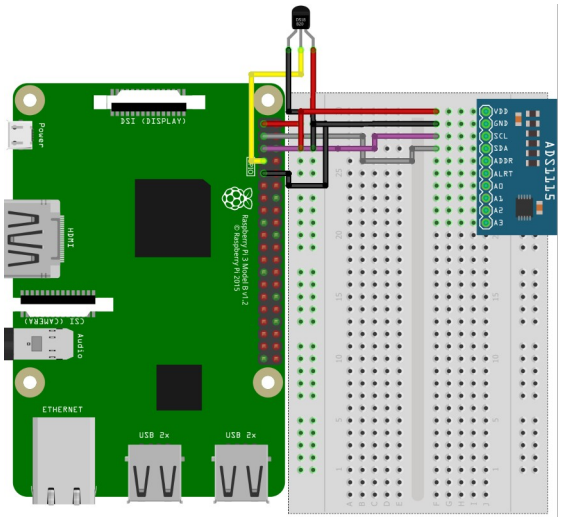


Abbildung 8: Schaltung des Temperatur Sensor mit PCB

Abbildung 9: Temperatur Sensor Schaltung normal

3.3.3 Code-Implementierung

Die folgende Implementierung basiert auf der Standard-Implementierung des Entwicklers. Zunächst werden mithilfe von `os.system` die beiden benötigten

Kernel-Module geladen. Anschließend beginnt der Sensor, seine Rohdaten in einem Unterordner von `/sys/bus/w1/devices/` zu speichern.

Um diesen Ordner zu finden, nutzen wir `glob`, um nach Verzeichnissen zu suchen, die mit „28“ beginnen. Dies liegt daran, dass die eindeutige ID des Sensors stets mit „28“ beginnt. Auf diese Weise können wir den Sensor Ordner finden, ohne die spezifische ID zu kennen. Die Datei, in die der Sensor seine Daten schreibt, heißt `w1_slave`.

Die Funktion `read_temp_raw()` liest den Inhalt dieser Datei aus. Anschließend verarbeitet `read_temp()` die Rohdaten: Zunächst wird geprüft, ob die Datei eine gültige Messung enthält, indem er überprüft, ob die erste Zeile der Datei „YES“ enthält, wenn nicht wiederholt er die Lesung. Danach wird die Temperaturmessung extrahiert und in Grad Celsius umgerechnet, indem der Wert durch 1000 geteilt wird. Schließlich wird das Ergebnis auf zwei Nachkommastellen gerundet zurückgegeben.

```
os.system('modprobe w1-gpio')
os.system('modprobe w1-
therm')
base_dir =
'/sys/bus/w1/devices/'
device_folder =
glob.glob(base_dir + '28*')
[0]
device_file = device_folder +
'/w1_slave'

def read_temp_raw():
f = open(device_file, 'r')
lines = f.readlines()
f.close()
return lines
def read_temp():
lines = read_temp_raw()
while lines[0].strip()[-3:] !
= 'YES':
time.sleep(0.2)
lines = read_temp_raw()
equals_pos =
lines[1].find('t=')
if equals_pos != -1:
temp_string = lines[1]
[equals_pos+2:]
temp_c = float(temp_string) /
1000.0
return round(temp_c,2)
```

3.4 pH-Wert

3.4.1 Theorie

Der pH-Wert beschreibt die Konzentration von Wasserstoffionen (H^+) in einer Lösung und wird auf einer logarithmischen Skala von 0 bis 14 angegeben. Ein niedriger pH-Wert weist auf eine hohe H^+-Konzentration hin (sauer), während ein hoher pH-Wert eine basische Lösung mit niedriger H^+-Konzentration signalisiert. In den Bereichen des pH-Wertes von 6,5 bis 9,5 gilt das Wasser als generell trinkbar. Dabei gilt ein pH-Wert von 7 als eine neutrale Lösung, da alles unter 7 säuerlich und alles über 7 basisch ist (UBA 2004). Wenn das Wasser sauer ist, verätzen die Lebewesen aufgrund der Säure und sterben daran, wenn aber das Wasser basisch ist, werden die Bakterien giftig und durch den Sauerstoffmangel sterben die Lebewesen ebenso daran. Die Temperatur hat einen großen Einfluss auf den pH-Wert, wenn die Temperatur steigt, sinkt der pH-Wert und wenn die Temperatur sinkt, steigt der pH-Wert. Da die beiden Werte stark zusammenhängen, ist es wichtig vorher die Temperatur zu messen, um den pH-Wert zu kompensieren (Wiley Analytical Science 2021). Das Säure-Base-Gleichgewicht hängt von der gleichen Teilchenanzahl von Oxonium- und Hydroxid-Ionen ab. Sobald sich die Temperatur verändert, wird das Säure-Base-Gleichgewicht verschoben und der pH-Wert ist nicht mehr neutral. Um den pH-Wert auszurechnen, wird die Konzentration von Oxonium-Ionen gebraucht. Dabei gilt die Formel: $\mathrm{pH} = -1\,g(c(H_3O^{+}))$ (FWU Institut 2025(2)). Eine andere Möglichkeit, den pH-Wert zu berechnen, ist durch die Nernst Gleichung. Dafür werden zwei Konzentrations-Halbzellen benötigt, hierfür wird das Redoxpaar: $H_3O^{+¿/H_2-¿}$

gebraucht. Die Oxonium-Ionen werden in Wasserstoff Gase umgewandelt, dabei finden in den Halbzellen die Reaktionen Oxidation und Reduktion statt. Die Reaktionen sehen folgend aus:

Oxidation:

$$2H_2O + H_2 \Rightarrow 2H_3O^{++2e^-}$$

Reduktion:

$$2H_3O^{++2e^-} \Rightarrow 2H_2O + 2H_2$$

Dadurch entsteht eine Konzentrationsdifferenz der Oxonium-Ionen mit einer Spannung, die man mit der Nernst Gleichung messen kann (Studyflix 2025 (2)). Beim Messen können Abweichungen vom eigentlichen Ergebnis entstehen, da die Nernst Gleichung nur von einer optimalen Reaktions Umgebung ausgeht und daher das Ergebnis verfälscht ist, wenn es bei einem Unwetter gemessen wird.

3.4.2 Verbindung mit dem Raspberry Pi

Zur Messung des pH-Wertes wurde das „Lab Grade Analog pH Sensor Kit for Arduino / Raspberry Pi“ von Gravity benutzt. Dieses kostet 37.31€ bei DFRobot. Die Sonde ist ein analoger Sensor, der eine Referenzspannung von 3V hat und einen potenziellen Fehlerwert von ±0.1 bei 25°C. Der Sensor kann entweder einen Slot auf dem Pi-HAT populieren oder per Breadboard an einen gemeinsamen GND- und VCC → 3V3 und einen der analogen Pins des ADC verbunden werden.

Dieser Schaltplan zeigt die Verbindung per Breadboard.

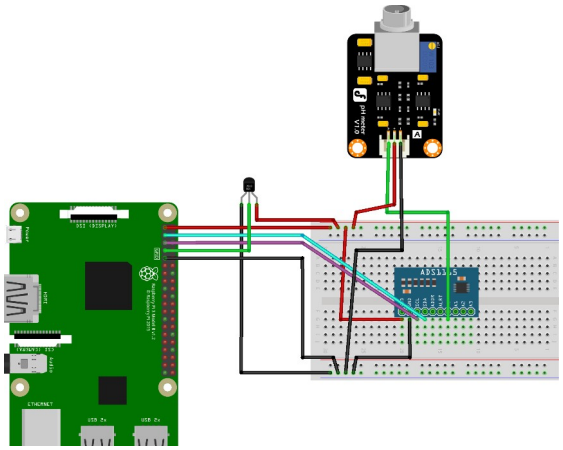


Abbildung 10: pH Schaltplan(Breadboard)

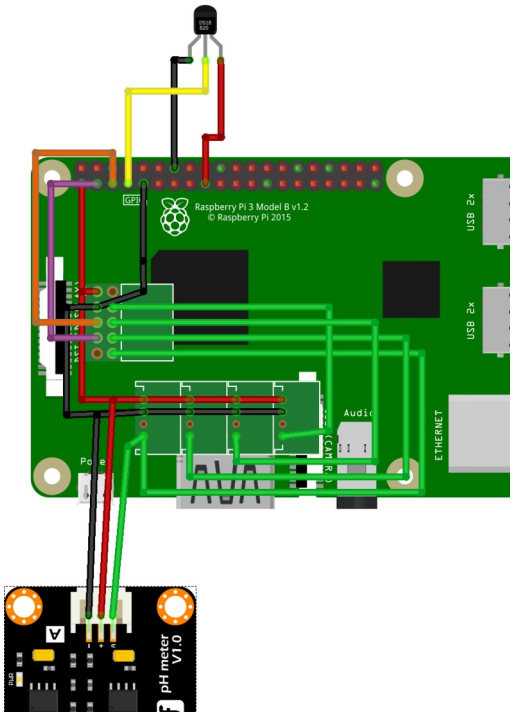


Abbildung 11: pH Sensor Schaltung PCB

3.4.3 Kalibrierung

Der pH-Wert wurde kalibriert indem zuerst der theoretische mV/pH mithilfe der Nernst-Gleichung gefunden wurde.

$$s_{theor.} = \frac{\ln(10) \cdot R \cdot T}{F}$$

wobei

$F = 96485\, As \cdot mol^{-1}; R = 8{,}31451\, J \cdot mol^{-1} \cdot K^{-1};$

$T = 273{,}15 + t[°C]$

Die Temperatur der benutzten Puffer betrug 20°C, es errechnet sich also

$$\frac{\ln(10) \cdot 8{,}3145 \cdot (273{,}15+20)}{96{,}485} = 58{,}17\, mV/pH$$

Um den praktischen Wert zu bestimmen, wurde die Sonde in einen pH 4 Citratpuffer und in einen pH 7 di-Natriumhydrogenphosphat Puffer hinein gesenkt um die Spannung zu messen. Da die Temperatur der Puffer 20°C betrug hatte der pH Puffer nun einen tatsächlichen pH Wert von 7,02.Im pH 7,02 Puffer wurde eine Spannung von 1533,125 mV gemessen und im pH 4 Puffer eine Spannung von 2049,125 mV. Vom Hersteller werden bereits die Formeln angegeben. Die Formel für die Steigung(m) war zuerst wie folgt

$$m = (7-4) \div ((U_{pH7} - 1500) \div 3 - (U_{pH4} - 1500) \div 3)$$

Wobei 3 die 3V Referenzspannung ist und 1500 die 1500 mV sind aus der Annahme, dass eine Hälfte des Spannungsspektrums für den pH unter 7 ist und eine für den pH über 7. Wenn man in die Formel nun unsere Messwerte einsetzt erhält man

$$m = (7{,}02-4) \div ((1533{,}125-1500) \div 3 - (2049{,}125$$

$-1500)\div 3) \quad = -\frac{151}{8600}\frac{pH}{mV}$

Der Kehrwert aus diesem ist, wie viel mV/pH wir praktisch haben.

$$-\frac{8600}{151}=56{,}95\frac{mV}{pH}$$

Dieses Entspricht nahezu dem Optimalwert mit einer Abweichung von 2,1% Daraufhin berechnet man anhand von der Formel des Herstellers den Y-Achsenabschnitt (Ys)

$$7{,}02-\frac{m\cdot(pH\,7(mV))-1500\,mV}{3\,V}$$

Wenn man die gemessenen Werte einsetzt erhält man

$$7{,}02-(-\frac{151}{8600}\frac{pH}{mV})\cdot\frac{1533{,}125-1500}{3}\approx 7{,}21$$

Wodurch man schließlich mit der Formel,

$$m\cdot(U[mV]-1500\,mV)\div 3\,V+Y_s$$

Wobei U eine Spannung ist,

Den tatsächlichen pH-Wert errechnen.

3.4.4 Code-Implementierung

Dies kann sehr simpel in das folgende Python Skript umgesetzt werden. Erst bestimmt man mit acidV und neutralV den Wert der in der pH 4 und pH 7,02 Lösung gemessen wurde. Die Funktion read_PH errechnet dann mit der bekannten Formel den pH Wert. Desweiteren wurde eine Error Funktion eingebaut die einen Fehlercode ausgibt sollte der pH Wert unter 0 oder über 14 sein, da das Spektrum der pH Werte nur von 0 bis 14 reicht.

```
    acidV=2049.12
neutralV=1533.12
def read_PH(voltage):
        global acidV
        global neutralV
        slope=(7.02-
4.0)/((neutralV-1500.0)/3.0-
(acidV-1500.0)/3.0)
        intercept=7.02-
slope*(neutralV-1500.0)/3.0

pH_value=slope*((voltage*1000
)-1500.0)/3.0+intercept
```

```
    return round(pH_value,2)
```

3.5. Total Dissolved Solids

3.5.1 Theorie

Total Dissolved Solids (TDS) bezeichnet die Menge an gelösten Stoffen in Wasser, darunter Salze, Mineralien und organische Verbindungen. TDS wird in Milligramm pro Liter (mg/L) oder parts per million (ppm) angegeben und ist ein wichtiger Indikator für die Wasserqualität. (OsmoFresh 2025). Da Salz Wasser entzieht, ist das ein Nachteil für die Lebewesen im Wasser, da die Lebewesen Wasser benötigen um zu leben.

3.5.2 Verbindung mit dem Raspberry Pi

Zur Messung des TDS Wertes wurde der „Analog TDS Sensor/ Meter for Arduino“

von Gravity benutzt. Dieser kostet 11.15€ bei DFRobot. Die Sonde ist ein analoger Sensor mit einer Referenzspannung von 2,3V. Um den Sensor zu Verbinden wird er entweder einen Slot auf dem Pi-HAT populieren oder per Breadboard an einen gemeinsamen GND- und VCC-3V3 und einen der analogen Pins des ADC verbunden werden. Er gibt die Menge an gelösten Feststoffen in ppm an. Er hat eine Messweite von 0 ~ 1000ppm mit einer Genauigkeit von ± 10% bei 25 °C.

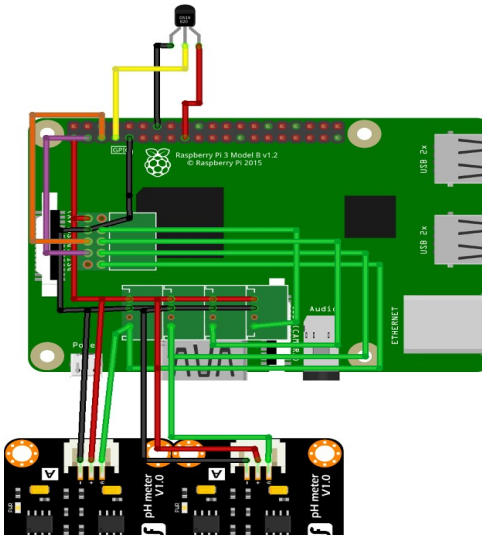


Abbildung 12: TDS Sensor mit PCB Schaltung

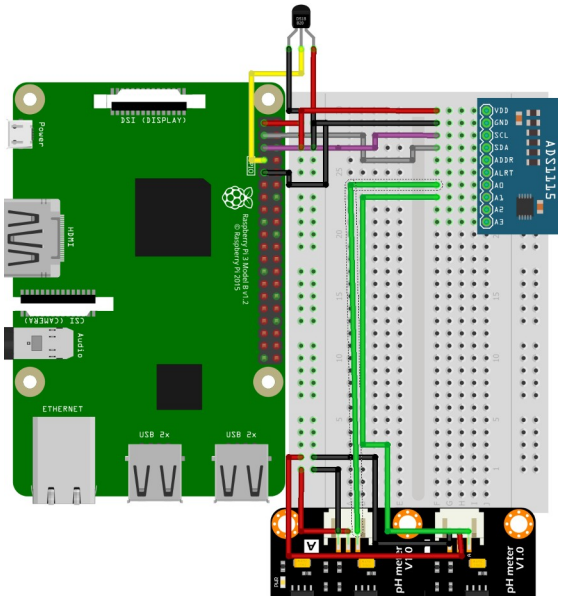


Abbildung 13: TDS Sensor Schaltung mit Breadboard

3.5.3 Kalibrierung

Der Sensor bestimmt die Menge an gelösten Feststoffen indem er die Leitfähigkeit misst, d.h., dass der Sensor nur bestimmen kann wie viele leitfähige Teilchen im Wasser gelöst sind. Um den Sensor zu kalibrieren wurde also eine Leitfähigkeitslösung von 0,01 mol/L KCl verdünnt und dann mit einem bereits kalibrierten Messgerät der BEG gemessen. Dabei wurde immer die Leitfähigkeit in Milisiemens pro cm notiert und dann die dazu gemessene Spannung notiert. Daraus bildet sich die folgende Tabelle

U[V]	EC[µS/cm]
0,066625	48
0,13325	101
0,256125	186
0,339625	277
0,470125	375
0,54775	440
1,594875	1400

Daraus folgt:

Leitfähigkeit in Abhängigkeit von Spannung

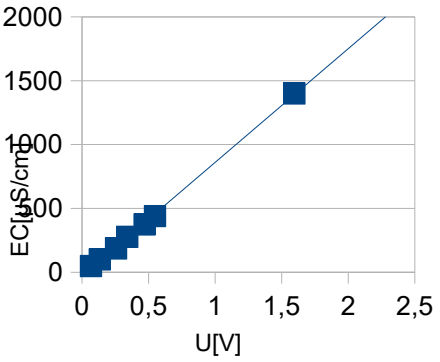


Abbildung 14: EC Graph

Die Funktion der Trendlinie ist f(x)= 890,419135176094 * x – 29,6974742651173

Mit einem Bestimmtheitsmaß von R^2 = 0,999009283277105

3.5.4 Code-Implementierung

Dies kann sehr simpel in das folgende Python Skript umgesetzt werden

```
def read_EC(voltage, temperature):
    EC_value = (890.419135176094 * voltage + 29.6974742651173) * (1.0 + 0.02 * (temperature - 25))
    return round(EC_value)

def read_tds(EC_value):
    TDS_value = EC_value * 0.64
    return round(TDS_value)
```

Hier wird erst eine Spannung und Temperatur an die read_EC Funktion übergeben die mit der bereits bekannten Funktion und einer Temperaturkompensationsformel von ECroh * 1 + 0,02 * (T[°C] - 25) die Leitfähigkeit berechnet dann berechnet read_tds die Menge an gelösten leitfähigen Feststoffen indem man es mit einem Umrechnungsfaktor von 0,64 multipiziert.

3.6 Dissolved Oxygen

3.6.1 Theorie

Dissolved Oxygen (DO) steht für den gelösten Sauerstoff im Wasser. Dabei ist es auch ein Maß für die Menge des gasförmigen Sauerstoffes im Wasser. Der gelöste Sauerstoff führt dazu, dass das Wasser sauber ist und Leben im Wasser stattfinden kann. Insgesamt gibt es drei Wege, wie gelöster Sauerstoff in das Wasser gelangen könnte: durch absorbierung aus der Atmosphäre, schnelle Bewegungen durch Wind erzeugt, Strömungen, mechanische Belüftung oder Photosynthese durch Pflanzen. Als Beispiel ist Phytoplankton eine wichtige Pflanze im Wasser, da sie tagsüber Sonnenenergie aufnimmt und dann Sauerstoff im Wasser erzeugt. Jedoch nachts kann die Phytoplankton keine Sonnenenergie aufnehmen, weshalb Fische nachts weniger Sauerstoff zur Verfügung

haben. Gleichzeitig dienen aber auch die Pflanzen als Nahrung für die Fische, die sie ebenso zum Leben im Wasser benötigen. Zum gelösten Sauerstoff spielen verschiedene Faktoren zum Beitrag mit. Einer der Faktoren ist der Druck, je höher der Druck in der Atmosphäre ist, desto mehr Sauerstoffmoleküle kann das Wasser aufnehmen. Ein weiterer Faktor ist die Temperatur, da bei hohen Temperaturen sich die Sauerstoff Moleküle schneller bewegen, was dazu führt, dass sie vom Wasser in die Luft abweichen. Auch die Fläche sowie die Tiefe sind wichtige Faktoren zum DO-Wert, da bei flachem Wasser die DO-Konzentration höher ist als bei tiefen Gewässern. Salz spielt ebenso eine wichtige Rolle, da bei einem niedrigen Salzgehalt die DO-Konzentration höher ist. Wenn die Bioaktivität von Mikroorganismen gering ist, steigt dadurch die Konzentration des gelösten Sauerstoffs. Der Grund dafür ist, dass Mikroorganismen Sauerstoff für die Atmung benötigen und desto weniger Mikroorganismen die Luft entziehen, kann auch mehr Sauerstoff ins Wasser gelangen (Hach 2025). Beim Messen können auch verschiedene Einflüsse den DO-Wert beeinflussen. Zum Beispiel können schnelle Bewegungen im Wasser dazu führen, dass der Sauerstoff übersättigt wird, da zu viele Gase freigesetzt werden.

3.6.2 Verbindung mit dem Raspberry Pi

Zur Messung des DO wurde das „Analog Dissolved Oxygen / DO Sensor Meter Kit for Arduino“ von Gravity verwendet. Dieses kostet 159.65€ von DFRobot und hat eine Referenzspannung von 3V. Der Sensor hat eine Messreichweite von 0~20mg/L mit einem Fehlerwert von 2% innerhalb von 90 Sekunden. Der Sensor kann entweder einen Slot auf dem Pi-HAT populieren oder per Breadboard an einen gemeinsamen GND- und VCC-3V3 und einen der analogen Pins des ADC

verbunden werden.

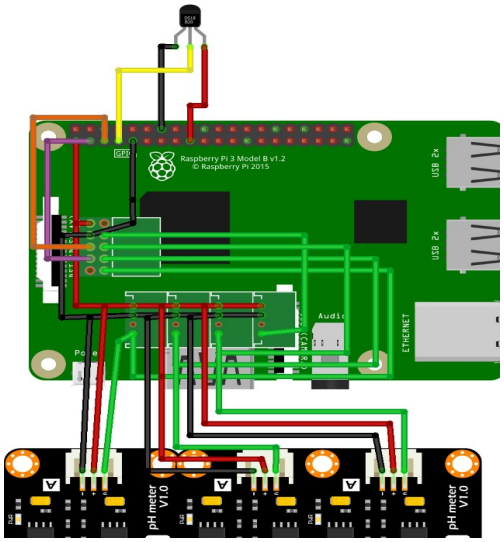


Abbildung 15: DO Sensor Schaltung PCB

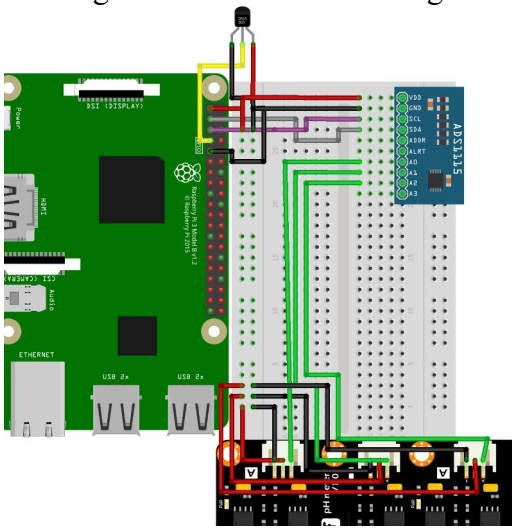


Abbildung 16: DO Sensor Schaltung Breadboard

3.6.3 Kalibrierung

Der Sensor wurde kalibriert indem eine Probe aus VE Wasser erst mit einem bereits kalibriertem BEG Messgerät gemessen, sobald diese einen stabilen DO Wert erreicht hatten, wurde die Spannung der Gravity Sonde in der selben Lösung gemessen. Da die Sonde immer unter Bewegung stehen muss um eine Sauerstoff zufuhr zu haben wurden die Sonden in einem Magnetrührer hinzugefügt der die Sonde aktiv mit Wasser angeströmt hat. Um den DO Wert auf fast 0 zu erniedrigen wurde Natriumsulfit im Wasser aufgelöst.

Damit hat sich die Folgende Tabelle gebildet

U[V]	O2[mg/L]
2,0085	18,4
1,80125	16,6
1,747	16,2
1,654	14,9
1,5395	14
1,451625	13,1
0,37225	1,7
0,108875	0,3

Daraus folgt:

Dissolved Oxygen in Abhängigkeit von Spannung

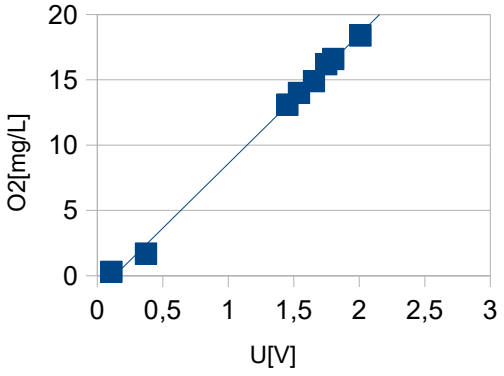


Abbildung 17: DO Graph

Diese Trendlinie hat eine Funktion $f(x)=9{,}8915916568544\,x-1{,}3089842087719$ wobei $R^2=0{,}9974607431071$

Dann wurde mithilfe von wasserdampfgesättigter Luft eine garantierte 100% Lösung von DO gebildet. Diese wurde bei fallender Temperatur gemessen um eine Temperaturkompensationsformel zu bilden. Daraus folgt:

T[°C]	U[V]
16,125	1,61025
15,125	1,570875
14,125	1,5305
13,125	1,47375
12,125	1,434375
11,125	1,376875

Daraus folgt:

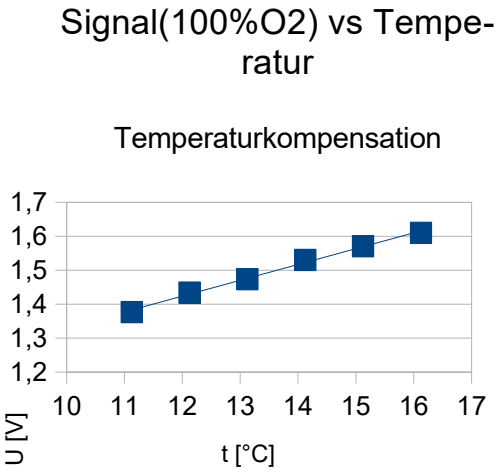


Abbildung 18: DO vs. Temperatur Graph

Die Trendlinie in diesem Graphen hat eine Funktion von f(x)= 0,0466607142857143 * x + 0,863685267857143. Diese Funktion hat ein Bestimmtheitsmaß von R²= 0,99579846620948

3.6.4 Code-Implementierung

Diese Funktion kann wie folgt in ein Python Skript implementiert werden. Hier gibt man an die Funktion read_DO die gelesene Spannung und die Temperatur und errechnet dann mit den 2 Funktionen die jeweiligen Werte. Schließlich teilt man die DO Konzentration durch den Temperatur Faktor um den eigentlichen DO Wert zu erhalten.

```
    def read_DO(voltage,
temperature)
    do_concentration =
9.8915916568544 * voltage -
1.30898420877194
    temp_comp_factor =
0.0466607142857143 *
```

```
temperature +
0.863685267857143
    compensated_do =
do_concentration /
temp_comp_factor
    return compensated_do
```

3.7 Turbidity

3.7.1 Theorie

Die Trübung, zeigt an, wie durchdringend das Licht im Wasser ist. Die Trübung entsteht entweder durch natürliche Ereignisse wie Regen, oder durch menschliche Ereignisse wie Bauarbeiten. Dies führt dazu, dass das Wasser trübe wird, was dazu führt, dass weniger Licht in das Wasser gelangt. Es bilden sich Sedimentschichten auf der Oberfläche, die dazu führen, dass das Wasser Licht reflektiert. Wenn sich genug Sedimentschichten bilden, kann diese Beschmutzung auch in das Trinkwasser gelangen (GlavierFresh 2024). Die Einheit die wir für Trübung benutzen sind FNU. Ein höherer FNU Wert weißt auf höhere Trübung und damit Beschmutzung und schwebenden Teilchen (Wasserfilteroase 2025). Eine starke Trübung ist gefährlich für die Lebewesen, denn ohne Licht können die Pflanzen keine Energie aufnehmen und sterben dadurch.

3.7.2 Verbindung mit dem Raspberry Pi

Zur Messung der Trübung wurde der „Analog Turbidity Sensor for Arduino“ von Gravity benutzt. Dieser kostet 9.35€ bei DFRobot und kann bis zu 3000 NTU messen, der Sensor hat dabei eine Genauigkeit von ±0,3V und eine Referenzspannung von 4,5V bei 5V Betriebsspannung. Es wird wie die anderen Sensoren angeschlossen.

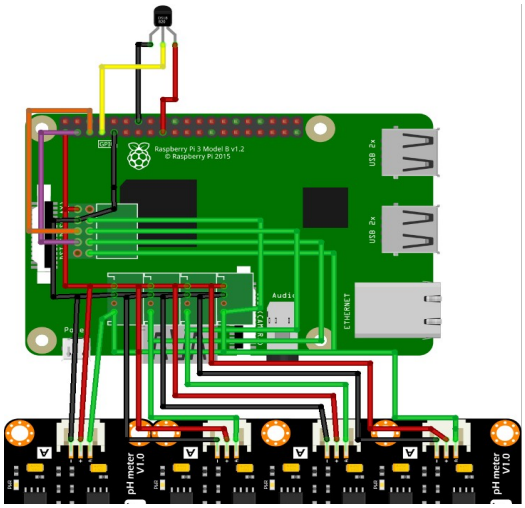


Abbildung 19: Turbidity Schaltplan(PCB)

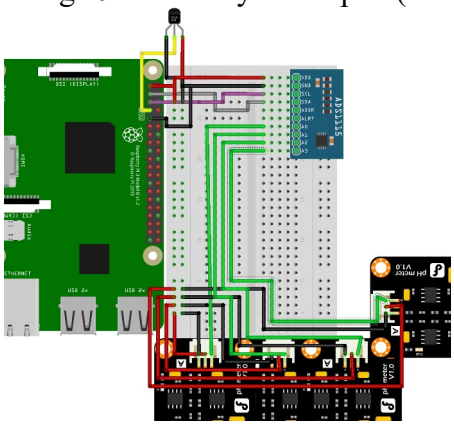


Abbildung 20: Turbidity Schaltplan(Breadboard)

3.7.3 Kalibrierung

Zur Kalibrierung wurde ein bereits kalibriertes FNU Messgerät der BEG verwendet. Um eine Trübung zu erzeugen wurden Klärwasser Proben mit VE Wasser gemischt, diese wurden dann zuerst im kalibrierten Messgerät gemessen und dann wurde die Spannung unseres Messgerätes gemessen. Daraus bildete sich folgende Tabelle:

U[V]	Turbidity[FNU]
1,93	96
1,874	193
1,683	400
1,653	500
1,573	530
1,442	850

Daraus folgt:

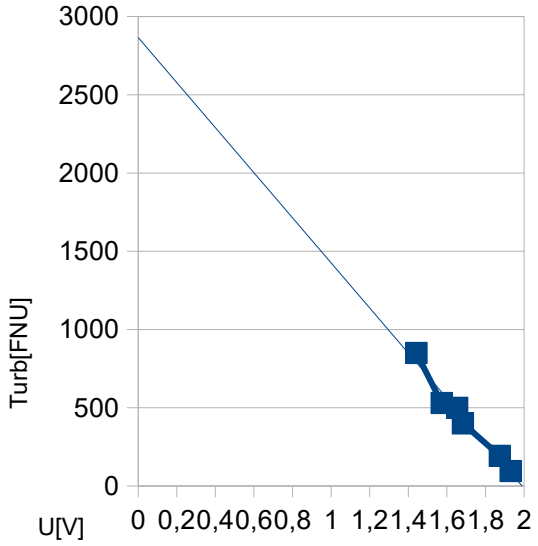


Abbildung 21: Graph Turbidity

Die Trendlinie hier hat eine Funktion von f(x) = -1,44010724307339 * x + 2865,54817556838. Hierbei muss notiert werden das der Sensor eine deutlich niedrigere Referenzspannung zeigt, da der Raspberry Pi ihn bei 3,3V Betriebsspannung benutzt und nicht 5V.

3.7.4 Code-Implementierung

Diese Funktion ist simpel in ein Python Skript umzusetzen

```
def read_turbidity(voltage):
turbidity_value=-1458.233516846
57*voltage+3011.70589885467
 return round(turbidity_value)
```

Neben den bereits benannten fortschritten für Tragbarkeit, wie das PCB wurde auch eine Powerbank gekauft und ein 3D-Modell entwickelt. Dadurch das der Pi so viel Strom verbraucht musste eine 20.000mAH Powerbank gekauft werden um Langzeit Operation zu Garantieren. 3D-Modell wurde mit OnShape entwickelt. Der Grundriss hat die Form eines Quaders, der nach und nach überarbeitet wurde. Die Box wurde mit ein paar Abrundungen an den Kanten, anschaulicher gestaltet. Damit auch in die Box Inhalte wie der Raspberry Pi oder Powerbank reinpasst, wurde ein Hohlraum von 5 mm Bodenabstand eingefügt und eine Wanddicke von 2,4 mm. Damit die Raspberry Pi nicht beim Transport verrutscht, wurden am Boden 4 Löcher

4. Konzept für Tragbarkeit und Umsetzung

eingefügt, mit einem Radius von 3 mm, wo die Raspberry Pi angeschraubt werden kann. Um einen Platz für die Sensoren zu schaffen, wurde ein Deckel entwickelt, der mit einem Lochabstand von 5 mm ermöglicht, beliebig die Sensoren an den Deckel anzubringen. Des Weiteren wurde eine Stufe an den Deckel angebracht, um zu verhindern, dass der Deckel beim Transport verrutscht. An der Box wurde ein Loch eingefügt, das es ermöglicht, die Sensoren außerhalb der Box einzusetzen. Zum Schluss wurden noch zwei weitere Löcher an der Box hinzugefügt, damit die Powerbank und der Raspberry Pi angeschlossen werden können. Das 3D Modell kann über einen 3D Drucker ausgedruckt werden und ist dann für eine portable Messstation einsatzbereit.

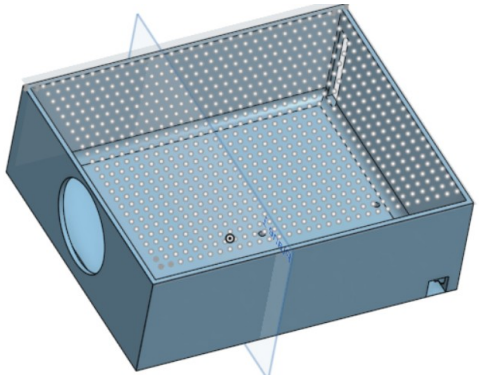


Abbildung 22: Das 3D Modell

5. Ergebnis Evaluation: Vergleich von ARWAQUTE und MORRIGAN

Um die technischen Verbesserungen von MORRIGAN zu bewerten, wurde ein Vergleich mit dem ursprünglichen ARWAQUTE-System durchgeführt. Die wichtigsten Unterschiede sind in der folgenden Tabelle dargestellt:

Merkmal	ARWAQUTE	MORRIGAN
Portabilität	Stationär, benötigt Netzteil	Tragbar, batteriebetrieben
In vivo Messungen	Schwierig, da fester Aufbau	Einfach durch mobile Nutzung
Prozessor	Weniger leistungsstark	Raspberry Pi mit mehr Leistung
Sensoren	Nicht kalibriert	Kalibrierte Sensoren für präzisere Messungen
Software	Monolithische r Code	Objektorientierte Programmierung
Hardware	Standardverkabelung	Custom HAT für einfachere Anschlüsse
Energieverbrauch	Geringer	Höher durch Raspberry Pi
ADC	Integriert	Externer ADC notwendig
Entwicklungsaufwand	Weniger, da bestehender Code	Höher, da Python-Support für Sensoren fehlte

Der Vergleich zeigt, dass MORRIGAN signifikante Fortschritte in Portabilität, Benutzerfreundlichkeit und Sensorgenauigkeit gemacht hat. Besonders die Einführung eines WebUI für die Datenanzeige und ein modulares Sensorkonzept sind große Vorteile. Allerdings bringt die höhere Leistung des Raspberry Pi auch Herausforderungen mit sich: Der Energieverbrauch ist gestiegen, sodass eine leistungsstarke Powerbank erforderlich ist. Zudem wurde ein externer ADC nötig, da der Raspberry Pi keine analogen Signale direkt verarbeiten kann.

6. Fazit

In dieser Arbeit wurde die Entwicklung eines portablen, modularen und benutzerfreundlichen Messsystems zur Analyse von Wasserqualität vorgestellt, basierend auf der Frage: „Welche technischen und gestalterischen Verbesserungen sind notwendig, um eine Wassermessstation portabler und anwenderfreundlicher zu gestalten?“ Die Antwort auf diese Frage wurde in den Verbesserungen von MORRIGAN im Vergleich zu ARWAQUTE gefunden.

MORRIGAN zeichnet sich durch eine höhere Portabilität, leistungsfähigere Hardware und eine verbesserte Benutzeroberfläche aus. Besonders die Integration eines Webinterfaces zur Echtzeitvisualisierung der Messwerte sowie die Nutzung eines speziellen Pi-HATs zur Optimierung der Verkabelung und Sensorintegration machen die Handhabung deutlich einfacher. Dies führte zu einer erhöhten Anwenderfreundlichkeit, da die Anzahl der notwendigen Schritte für den Endnutzer reduziert wurde – der Pi-HAT verringert den Verkabelungsaufwand und das WebUI ermöglicht eine einfachere Datenanzeige.

Des Weiteren zeigte sich, dass 3D-gedruckte Gehäuse die Portabilität erheblich verbessern, da sie den Transport der Messstation vereinfachen. Ein weiteres Ergebnis war die Feststellung, dass modulare Anpassungen (wie das Custom HAT und die Integration eines Webinterfaces) die Flexibilität und Benutzerfreundlichkeit im Vergleich zu bestehenden Systemen wie ARWAQUTE steigern.

Allerdings gab es im Rahmen der Arbeit auch einige Herausforderungen und Einschränkungen: Der höhere Energieverbrauch des Raspberry Pi im Vergleich zu anderen Systemen erschwert

den mobilen Einsatz ohne Netzstromversorgung. Zudem war die Entwicklung auf einige Python-basierte Sensoren angewiesen, da viele Sensoren keine unterstützenden Treiber hatten. Weitere Herausforderungen bestanden in der Kalibrierung der Sensoren und der Notwendigkeit, externe ADCs zu verwenden, um die analogen Signale zu verarbeiten.

Trotz dieser Herausforderungen zeigt die Arbeit, dass MORRIGAN eine solide Grundlage für zukünftige Entwicklungen und Optimierungen im Bereich der tragbaren Wassermessstationen darstellt. Zukünftige Arbeiten könnten sich auf die Verbesserung der Energieeffizienz, die Erweiterung des Systems um zusätzliche Sensoren und die Verbesserung der Langzeitauswertung konzentrieren.

7. Quellenverzeichnis

https://dartmouthocean.com/products/phosphate-sensor

Bishop, I.J., Warner, S., van Noordwijk, T.C.G.E., Nyoni, F.C., and Loiselle, S. (2020) Citizen science monitoring for sustainable development goal indicator 6.3.2 in England and Zambia.Sustainability, 12(24), pp. 1–15. https://doi.org/10.3390/su122410271 (26.01.2025)

Fraisl, D., Hager, G., Bedessem, B. *et al.* Citizen science in environmental and ecological sciences. *Nat Rev Methods Primers* **2**, 64 (2022). https://doi.org/10.1038/s43586-022-00144-4 (26.01.2025)

FWU Institut 2025: https://www.leifiphysik.de/waermelehre/ausdehnung-bei-erwaermung/grundwissen/anomalie-des-wass ers (23.01.2025, 9 Uhr)

FWU Institut 2025 (2): https://www.leifichemie.de/saeuren-und-basen/saeure-base-gleichgewicht/grundwissen/ph-wert-berech nung (23.01.2025, 17 Uhr)

GlavierFresh 2024: https://glacierfreshfilter.com/de/blogs/news/understanding-turbidity-in-drinking-water-why-it-matters-and-how-to-reduce-it?srsltid=AfmBOoofyhSBHgyn5OLm75oEWP2ZNqzPgdEeHNARCqCwo6auiuR9 OiOl (23.01.2025, 21 Uhr)

Hach 2025: https://at.hach.com/parameters/dissolved-oxygen (25.01.2025, 13 Uhr)

Kosmala, M., Wiggins, A., Swanson, A. and Simmons, B. (2016) Assessing data quality in citizen science, Frontiers in Ecology and the Environment, 14(10), pp. 551–560 https://doi.org/10.1002/fee.1436 (26.01.2025)

OsmoFresh 2025: https://www.osmofresh.de/service/blog/Was-misst-e in-TDS-Messgeraet (23.01.2025, 20 Uhr)

OpenedTech 2024: https://openedtech.ellak.gr/robotics2024/arwaqute- metrisi-elegchos-paragonton-piotitas-nerou-me-ard uino/ (26.01.2025)

Kolade Olatunde, Susan Kane Patton, Laura Cameron, Tony Stankus, Plangkat James Milaham. Factors Affecting the Quality of Drinking Water in the United States of America: A Ten-Year Systematic Review. *American Journal of Water Resources*. Vol. 10, No. 1, 2022, pp 24-34. https://pubs.sciepub.com/ajwr/10/1/4 (26.01.2025)

Pellerin, Brian A., Beth A. Stauffer, Dwane A. Young, Daniel J. Sullivan, Suzanne B. Bricker, Mark R. Walbridge, Gerard A. Clyde, Jr., and Denice M. Shaw, 2016. Emerging Tools for Continuous Nutrient Monitoring Networks: Sensors Advancing Science and Water Resources Protection. *Journal of the American Water Resources Association* (JAWRA) 52(4): 993–1008. https://doi.org/10.1111/1752-1688.12386 (26.01.2025)

Rainear, Adam. (2025). Publics and Citizen Science. http://dx.doi.org/10.1007/978-3-031-74062-6_40 (26.01.2025)

SEAWATER Cube: https://seawatercubes.de/wassertemperatur-fischwohl/#:~:text=Die%20g%C3%A4ngigen%20Speisefische%20Wolfsbarsch%20und,C%20sollten%20nicht%20%C3%BCberschritten%20werden. (23.01.2025, 9 Uhr)

SEBA Hydrometrie GmbH & Co. KG 2019: https://www.seba-hydrometrie.com/news/wie-haeng t-ec-mit-tds-und-salzgehalt-zusammen (23.01.2025,20 Uhr)

Studyflix 2025: https://studyflix.de/chemie/aggregatzustand-einfach-erklart-4087 (23.01.2025, 10 Uhr)

Studyflix 2025 (2): https://studyflix.de/chemie/nernst-gleichung-1575 (23.01.2025, 23 Uhr)

StudySmater GmbH 2025:https://www.studysmarter.de/ausbildung/ausbi ldung-in-chemie/biologielaborant-ausbildung/wasse rtemperatur/ (23.01.2025, 10 Uhr)

Thomei 2023: https://www.wasserfilteroase.de/blogs/glossar/neph elometric-turbidity-units?srsltid=AfmBOor0ESKnv9IybImHgqNjNh0A2lbrcctgQqBlwpmNiTTr03gMr n2O (25.01.2025, 19 Uhr)

UBA 2004: https://www.umwelt.niedersachsen.de/startseite/themen/wasser/grundwasser/grundwasserbericht_niedersachsen/grundwasserbeschaffenheit/guteparameter/grundprogramm_des_nlwkn/ph_wert/pH-Wert-137608.html#:~:text=Die%20Trinkwasserverordnung%20sieht%20f%C3%BCr%20den,alkalischen%20Charakter%20einer%20w%C3%A4ssrigen%20L%C3%B6sung (23.01.2025, 16 Uhr)

Ullrich, National Geographic 2024:https://education.nationalgeographic.org/resource/citizen-science-article/ (26.01.2025)

Weigelhofer, G. and Pölz, E.-M. (2016) Data quality in citizenscience projects: challenges and solutions. Frontiers in Environmental Science, 4. https://doi.org/10.3389/conf.fenvs.2016.01.00011 (26.01.2025)

Wiley Analytical Science 2021: https://analyticalscience.wiley.com/content/article-d o/ph-und-temperatur-zwei-un-shy-trenn-shy-bare-gr%C3%B6%C3%9Fen (23.01.2025, 17 Uhr)

Xylem 2024: https://www.xylemanalytics.com/de/unternehmen/blog/xylem-analytics-blog/2022/09/kalibrierung-und- justierung-einer-ph-elektrode (24.01.2025)

8. Anhang