

Audit 4

Entwicklung eines Systems zur Überwachung und Automatisierung von Aquaponikanlagen

Anne Germund und Verena Heissbach





Feedback aus Audit 3

1. Architekturdiagramm für Audit 4
2. Installation eines Displays im Grafana-Kiosk-Modus
3. Zugriff via SSH testen
4. Automatisches Durchführen der Messungen nach Start des Raspberry Pi
5. Code sinnvoll aufteilen, Clean Code
6. GUI für Grundeinstellungen durch Anlagenbetreiber*innen (Min- und Max-Werte, Fülldauer etc.)

**Durchgeführte Proof-of-Concepts und
Architekturdiagramm**



Proof-of-Concept #8 | Durchführung Use Case #2



Offizieller Raspberry Pi 7-Zoll Touchscreen
Quelle: <https://tutorials-raspberrypi.de/testbericht-zum-7-raspberry-pi-touchscreen-display/c>

Visualisierung der Messdaten auf lokalem Display

Leider mussten wir beim ersten Anschluss-Versuch des vorhandenen LCD-Displays feststellen, dass dieses nicht mit dem verwendeten Raspberry Pi kompatibel zu sein scheint. Nach der Verbindung der Komponenten ließ sich der Raspberry Pi nicht mehr starten und war erst nach Entfernen des Displays wieder einsatzfähig. Aus diesem Grund soll für den endgültigen Einsatz im Institut ein größeres und moderneres Touch-Display (siehe Foto) verwendet werden, welches jedoch für das 4. Audit nicht rechtzeitig geliefert werden konnte. Für den funktionalen Prototyp wurde daher ein Bildschirm per HDMI-Kabel angeschlossen

Sobald der Raspberry Pi hochgefahren wird, öffnet sich Grafana automatisch im Kiosk-Modus. Dies wird durch eine Service Datei ermöglicht, welche von systemd nach dem Booten gestartet wird.

Verwendete Tutorials:

- <https://www.loganmarchione.com/2020/05/always-on-grafana-dashboard-using-raspberry-pi-touchscreen/>
- <https://github.com/grafana/grafana-kiosk>
- <https://blog.muffn.io/making-a-raspberry-pi-grafana-monitor/>



Proof-of-Concept #9

GUI für Werkseinstellungen

Beschreibung

Über eine GUI können die Werkseinstellungswerte, Wasser GPIO, Wasser Channel, Wasserstand Max. und Min, Temperatur Max. und Min. angepasst werden. Der Nutzer muss diese Werte dann nichtmehr im Code selber ändern. Damit besteht auch keine zusätzliche Gefahr, dass Code verändert werden könnte.

Verwendete Tutorials: https://www.python-kurs.eu/tkinter_entry_widgets.php,
<https://pythonbuch.com/gui.html#menu-und-messagebox>

Schritte

- Tkinter Modul importieren
- Key-Value Store in Form eines Dictionary implementieren
- GUI coden: Fenster, Label und Eingabefelder erstellen. Bestätigungsbutton initialisieren
- Button Action definieren → eingegebene Werte überschreiben die Setting werte
- Bestätigungslabel gibt Feedback

Exit Kriterien

Es öffnet sich ein GUI Fenster mit Eingabefeldern, nach der Eingabe und dem betätigen des Bestätigen Buttons werden die Werte in die Settings Werte korrekt übernommen (überschrieben). Mit dem Klick auf das X wird das Fenster geschlossen.

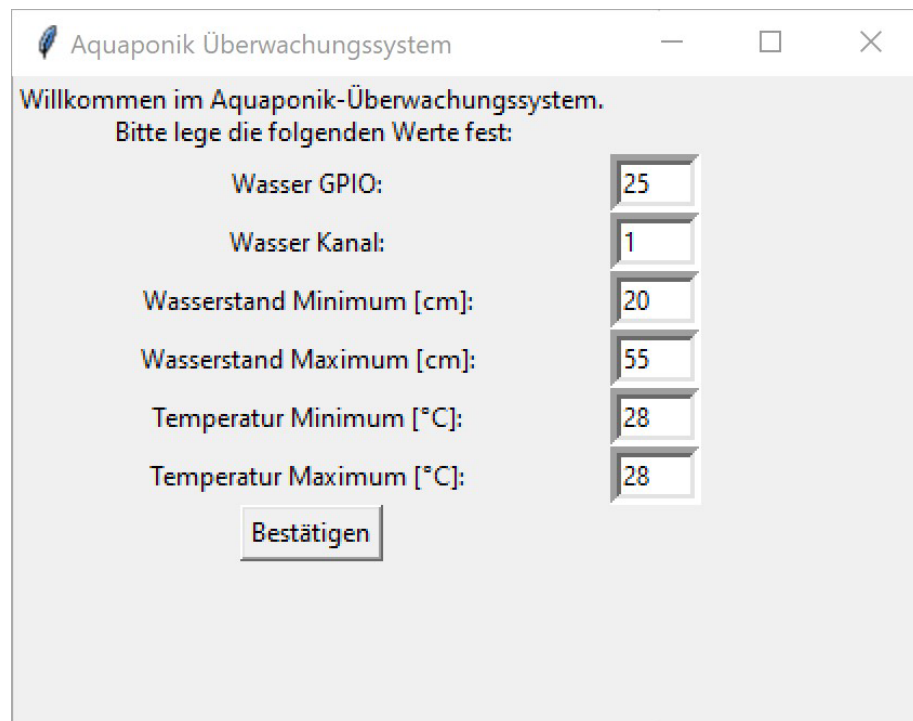
Fail Kriterien

Es öffnet sich kein Fenster, es ist keine Eingabe möglich und oder die Daten lassen sich nicht in den Key-Value-Store übernehmen.

Fallbacks

Änderungen der Werte müssen weiterhin im Code selbst angepasst werden.

Proof-of-Concept #9 | Durchführung



Eigene GUI für Werkseinstellungen
Quelle: Eigene Darstellung

GUI für Werkseinstellungen

Mithilfe des tkinter Moduls welches bereits in python enthalten ist wurde eine einfache GUI erstellt. Es wurden die passenden Labels und Eingabefelder erzeugt.

Als Basis der Werte wurde ein key-value-store in Form eines Dictionary angelegt.

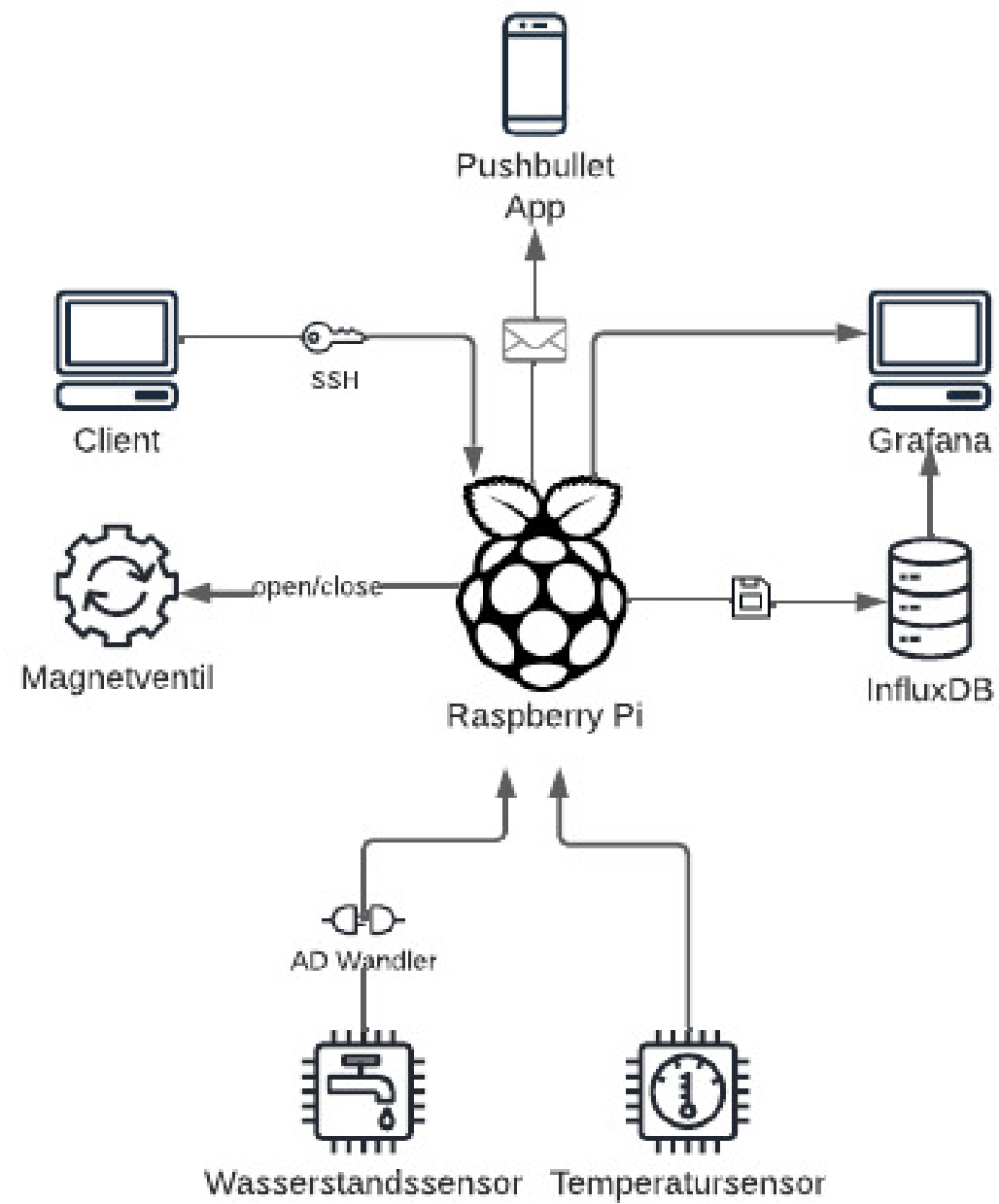
Mittels der get() Methode konnten die Eingabewerte ermittelt werden. Über insert() werden die Setting Werte in den Eingabefeldern automatisch angezeigt.

Anschließend wurde die Button Action für den Bestätigungsbutton definiert. Die eingegebenen Werte sollen die vorhandenen Daten des kvs überschreiben.

Zum Schluss wurden die einzelnen Komponenten zum Fenster hinzugefügt und über die Funktion mainloop() eine Endlosschleife implementiert die auf eine Aktion des Nutzers wartet. Über einen Klick auf das X wird das Fenster geschlossen und die GUI Anwendung beendet.

Zusätzlich wurde eine einfache Validierung der Eingabewerte implementiert, um sicherzustellen, dass nur numerische Werte und im passenden Wertebereich liegende Werte gespeichert werden.

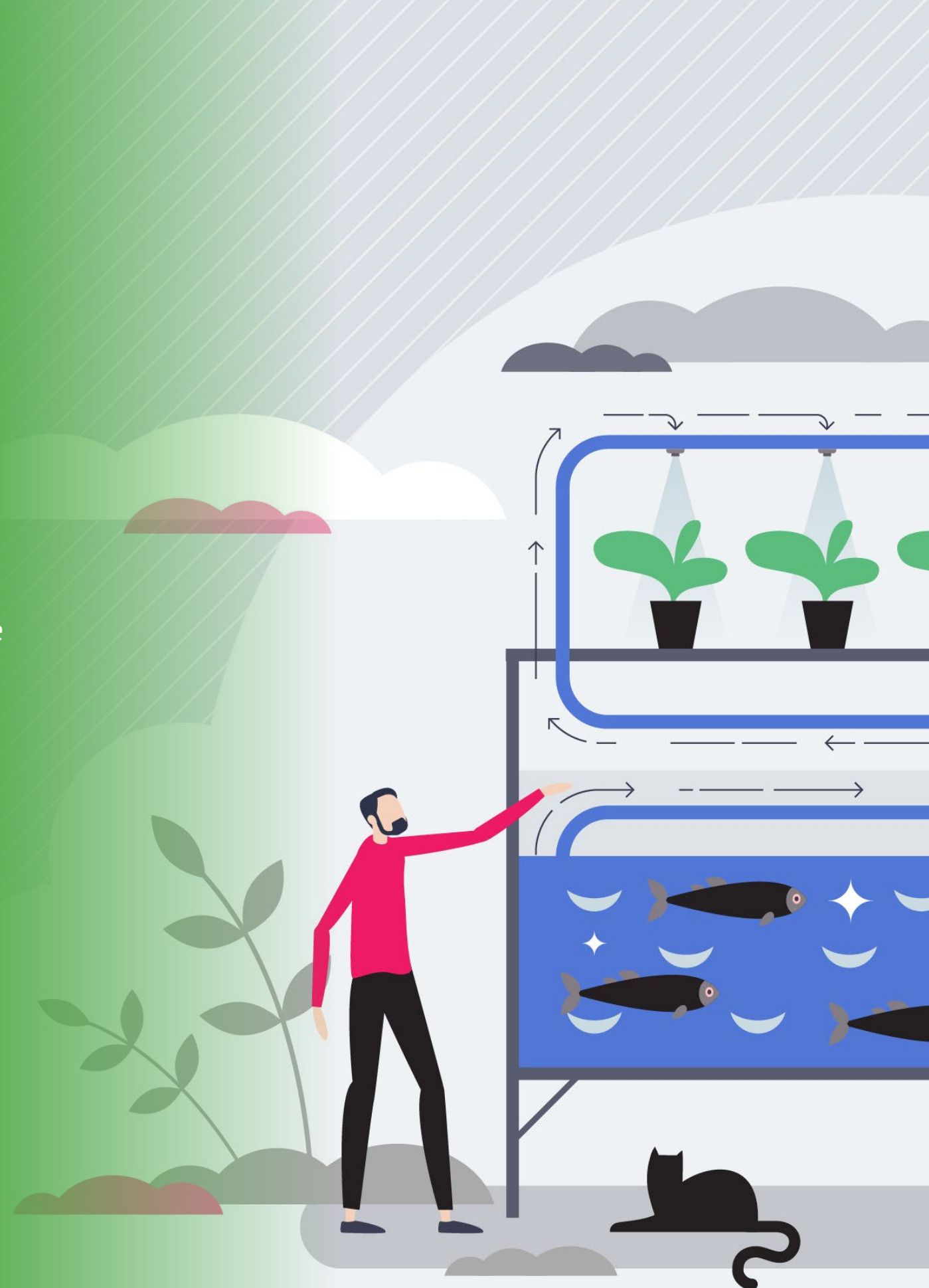
Architekturdiagramm



Software-Architektur-Diagramm
Quelle: Eigene Darstellung

Funktionaler Prototyp

Darstellung der Implementierung der modellierten Konzepte

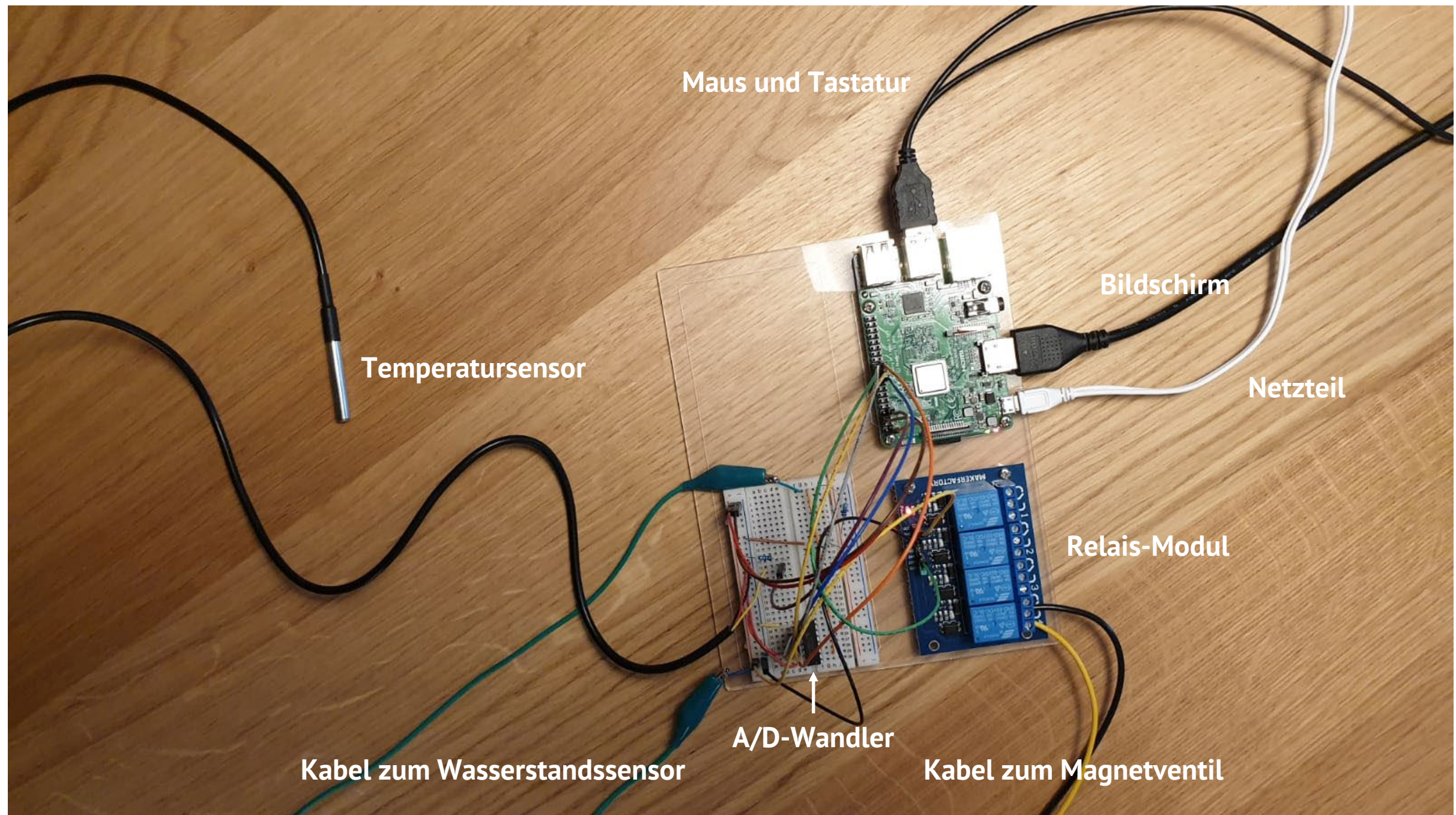


Demonstration des funktionalen Prototyps



Raspberry Pi mit angeschlossenem Bildschirm im Grafana-Kiosk-Modus.
Quelle: Eigene Darstellung

Demonstration des funktionalen Prototyps

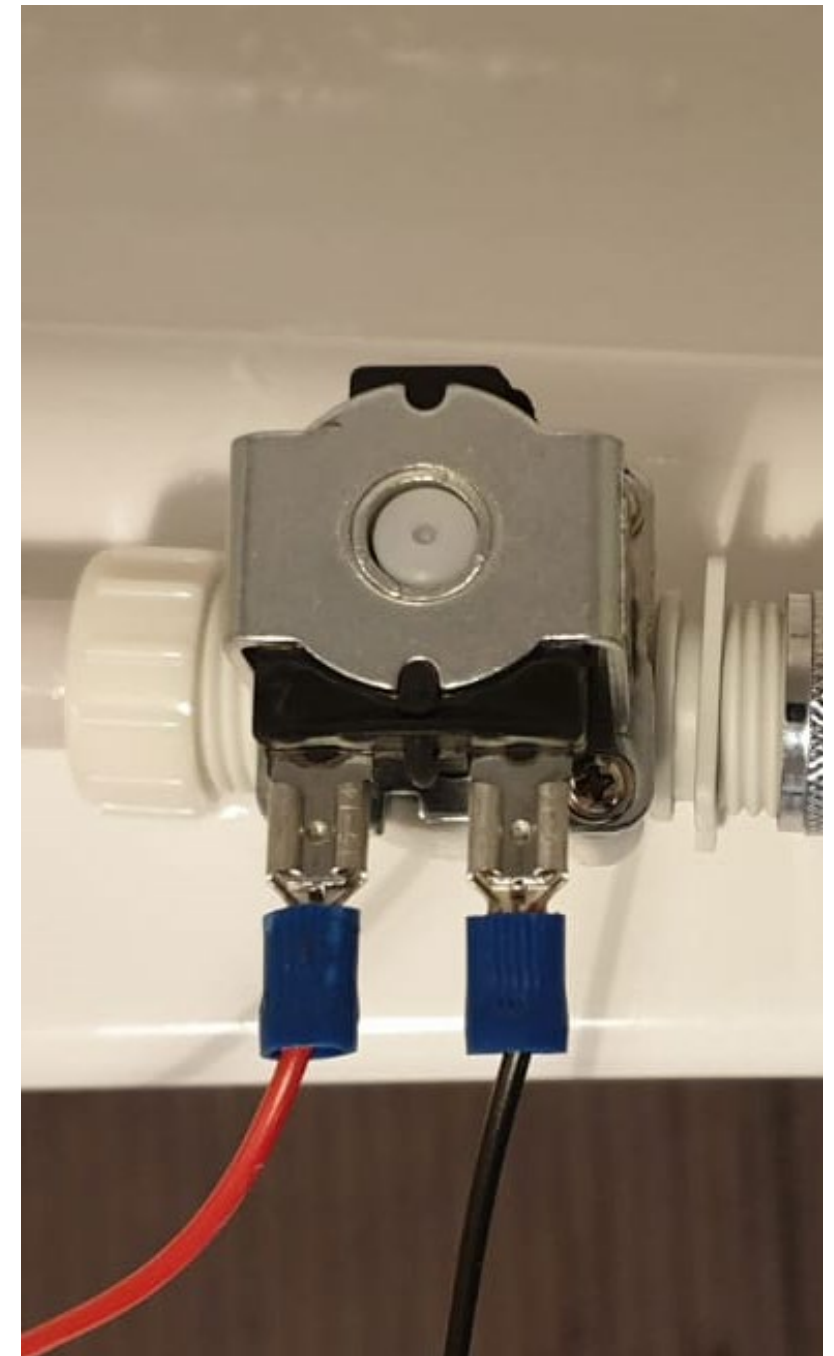


Verkabelung des Raspberry Pi.
Quelle: Eigene Darstellung

Demonstration des funktionalen Prototyps

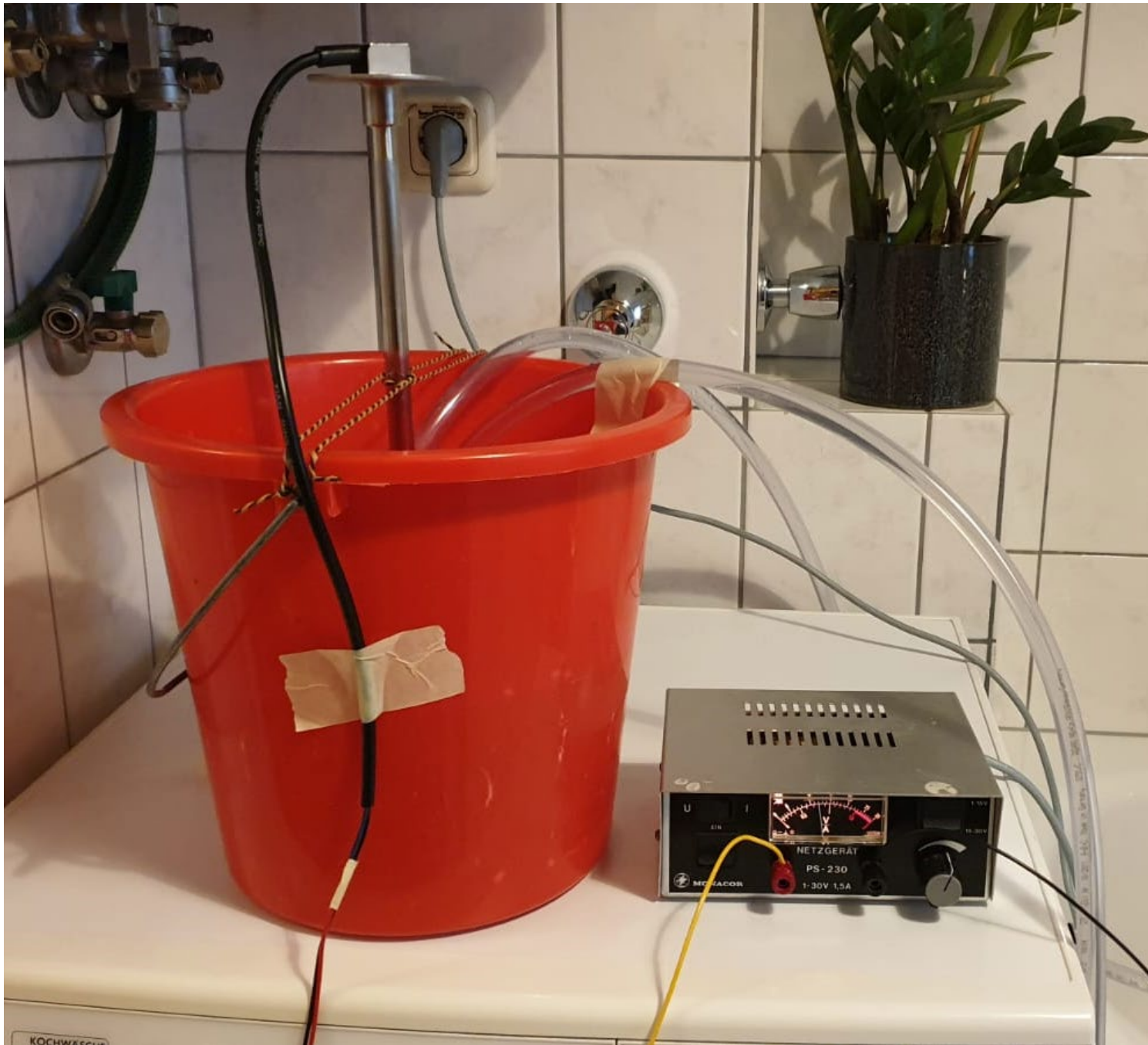


Prototyp des Fischbeckens.
Quelle: Eigene Darstellung



Detailansicht des Magnetventils.
Quelle: Eigene Darstellung

Demonstration des funktionalen Prototyps



Wasserstandssensor im prototypischen Fischbecken.
Quelle: Eigene Darstellung

Demonstration des funktionalen Prototyps



Video zum funktionalen Prototypen.
Quelle: Eigene Darstellung

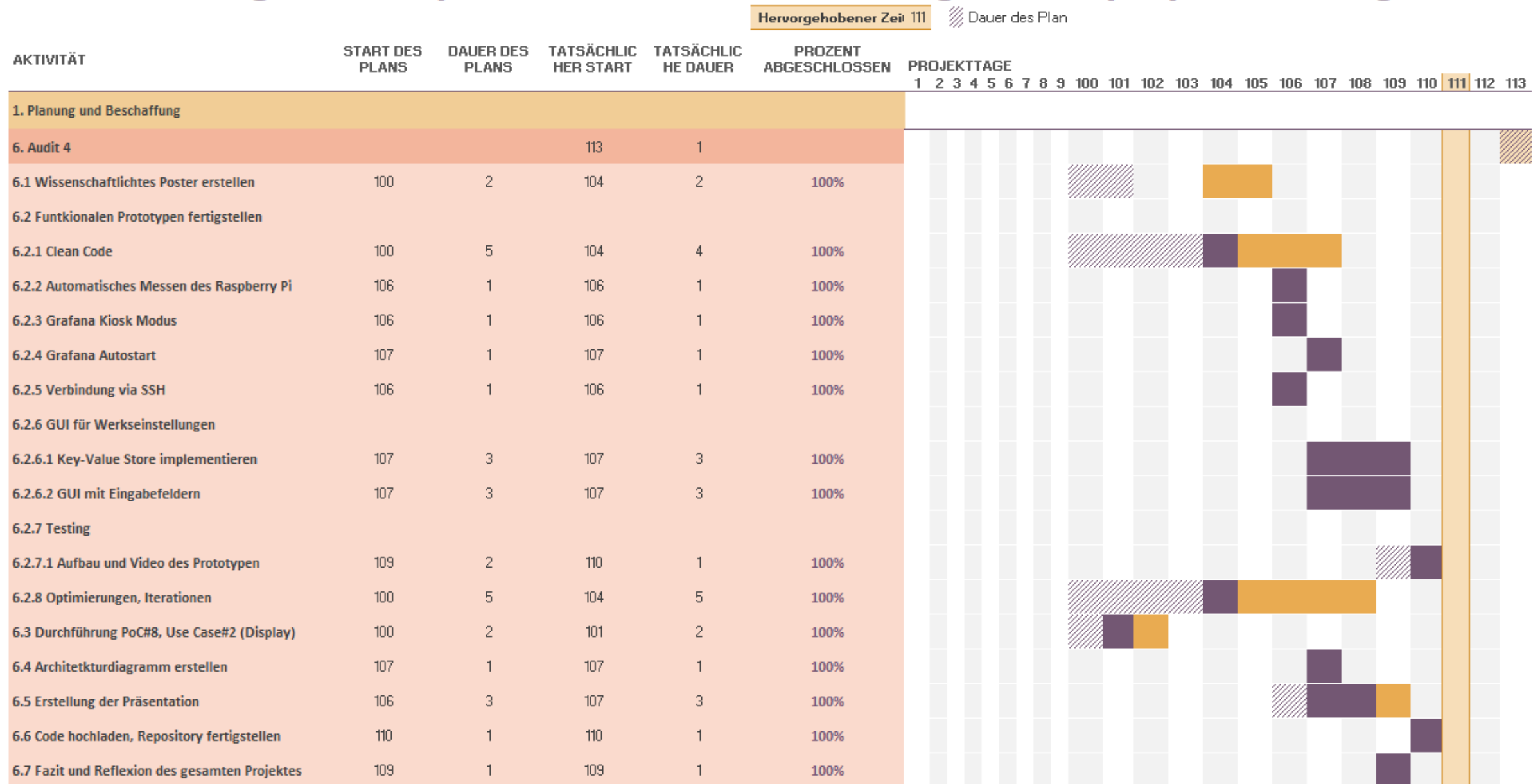
Projektplan

kritisch reflektiertes Prozessassessment des gesamten Projektes anhand der ursprünglichen Zielsetzung



Gantt-Diagramm zur Projektplanung

Entwicklung eines Systems zur Überwachung von Aquaponikanlagen



Projektplan Fokus Audit 4.
Quelle: Eigene Darstellung

Reflexion und Ausblick

kritisch reflektiertes Prozessassessment des gesamten Projektes
anhand der ursprünglichen Zielsetzung



Evaluation der Zielsetzungen

Es soll ein System zur Fernüberwachung von Aquaponikanlagen entstehen, welches den Betrieb, die Kontrolle und Wartung erleichtert und somit auch die Gesamtsicherheit der Anlage erhöht.



Die Fernüberwachung läuft noch nicht optimal, da der Zugriff auf Grafana-Daten zurzeit nur über das lokale Netz funktioniert. Dafür wurde ergänzend ein lokales Display angebracht, welches die Kontrolle vor Ort maßgeblich erleichtert. Sobald der Raspberry Pi gestartet ist, werden die Messungen automatisch durchgeführt und auf dem Grafana-Dashboard ausgegeben.

Ein System von Sensoren und Schnittstellen übernimmt die Überwachung der Messwerte und benachrichtigt die Betreiber der Anlage bei einem Eintritt in kritische Wertebereiche.



Nach anfänglich Kompatibilitätsproblemen konnten die Hardwarekomponenten letztlich zuverlässig ans Laufen gebracht werden. Zurzeit erfolgt die Beurteilung der kritischen Werte nur anhand von Minimal- und Maximal-Werten, das Oszillieren des Wasserstands an der Institutsanlage wird noch nicht berücksichtigt (weitere Codeanpassungen notwendig). Sobald kritische Messwerte festgestellt werden, erhalten registrierte Geräte zuverlässig eine Warnung.

Zusätzlich soll es möglich sein, dass das System selbst aktiv wird, um auf kritische Werte angemessen zu reagieren.



Bei einem zu geringen Wasserpegel wird das Magnetventil geöffnet und es wird so lange Wasser nachgefüllt, bis der gewünschte Wasserstand erreicht ist. Hier müssen für den endgültigen Aufbau der Anlage noch zusätzliche Sicherheitsmechanismen angebracht werden, um eine Überflutung zu verhindern → bspw. Niveauschalter oder Wasserwächter

Eine manuelle Kontrolle der Messwerte durch die Betreiber wäre somit nicht mehr notwendig. Auch eine tägliche visuelle Kontrolle der Anlage müsste auf diese Weise nicht mehr stattfinden.




Eine manuelle Kontrolle der Messwerte ist durch den Einsatz der Sensoren nicht mehr notwendig. Eine visuelle Kontrolle sollte jedoch weiterhin regelmäßig stattfinden – besonders hinsichtlich der Wasserversorgung –, bis eine Videoüberwachung in das System integriert wurde.



Evaluation der Zielsetzungen

Strategische Ziele:

- Z_S1: Fernüberwachung der Aquaponikanlage 
- Z_S2: Automatisierung der Anlage
- Z_S3: Vereinfachung der Kontrolle
- Z_S4: Reduzierung des zeitlichen Aufwands
- Z_S5: Zugänglichkeit der Anlagen erhöhen
- Z_S6: Warnsystem der Anlage (mittels Benachrichtigung der Nutzer)



Taktische Ziele:

- Z_T1: Aufbau eines Internet of Things 
- Z_T2: Raspberry Pi als Schnittstelle nutzen
- Z_T3: Sensoren zur Aufnahme physikalischer und chemischer Messwerte
- Z_T4: Kompetenzen im Bereich Web Development erwerben/nachholen



Operative Ziele:

- Z_O1: Erste Übungen am Raspberry Pi durchführen
- Z_O2: Python lernen (Codecademy etc.)
- Z_O3: Komponenten bestellen
- Z_O4: Raspberry Pi einrichten
- Z_O5: Sensoren anschließen, mit Raspberry Pi verbinden
- Z_O6: Erste Datenübertragungen testen
- Z_O7: Installation der Hardware-Komponenten



Fazit

Da wir uns bereits im Modul MCI mit der Digitalisierung von Aquaponikanlagen beschäftigt hatten und uns dieses Thema weiterhin sehr interessierte, wollten wir unsere bestehenden Entwürfe in einen praktischen Prototypen umsetzen. Durch dieses und vorangegangenen gemeinsame Uniprojekte wussten wir, dass wir als Team gut harmonieren. Zusätzlich konnte uns unsere Motivation für das Thema sehr dabei helfen, die Aufgaben trotz der fehlenden Vorkenntnisse im Bereich Web Development zu bewältigen. Dabei griffen wir auf bereits erlernte Methoden zurück und konnten Probleme meist schnell mit Hilfe von Tutorials oder Issues bei Github lösen. Aufgrund der aktuellen Situation lief ein Großteil der Projektarbeit online ab, was durch Tools wie Zoom, Code with Me von PyCharm, Figma, PowerPoint und GitHub aber trotzdem gut umzusetzen war. In regelmäßigen Meetings konnten wir Schritt für Schritt die definierten Ziele realisieren und unseren Prototyp durch Anregungen aus den Open Spaces und Audits erweitern und iterieren. Angesichts des hinzugewonnenen Wissens und neuen Kompetenzen hat sich der Schritt in diese neue Domäne für uns gelohnt.

Ausblick

- Zugriff über öffentliches Netz ermöglichen
- GUI Optimierungen
- Validierung der Eingaben im GUI erweitern
- Mobile Überwachung
- kleineres Touch Display installieren
- Kamera bzw. weitere Hardware (Sensoren etc.) installieren
- Datensicherheit, Zugriffsrechte
- ausreichende Sicherheitsvorkehrungen treffen (Überflutung, Stromausfall, keine Internetverbindung etc.)
- Installation des Systems an der Aquaponikanlage
- Testing und Evaluation