Pflichtenheft

Automatische Balkonbewässerung

Autor: Dzaid Abiyyu Siregar, Zul Fahmi Nur Vagala, Johannes Berg
Letzte Änderung: . 22 Mai 2025
Dateiname: Pflichtenheft_Automatische_Balkonbewässerung.docx

Version: 1.3

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Copyright

Version Historie

Version:	*Datum:*	*Verantwortlich*	*Änderung*
1.0	16.05.2025	Johannes Berg	Überblick, Ziele, Annahmen und Abgrenzungen
1.1	22.05.2025	Zul Fahmi Nur Vagala	Funktionalität
1.2	22.05.2025	Dzaid Abiyyu Siregar	Workflow & Funktionalität
1.3	23.05.2025	Dzaid & Zul & Johannes	Abgabe

1 Vorhandene Dokumente

Tabelle 1: Vorhandene Dokumente

Dokument	Autor	Datum
Lastenheft	Dzaid Abiyyu Siregar, Zul Fahmi Nur Vagala, Johannes Berg	25.04.2025

2 Überblick

Das Projekt „Automatische Bewässerung für Balkonpflanzen“ hat das Ziel, eine intelligente und kostengünstige Lösung zur automatisierten Pflanzenbewässerung zu entwickeln. Der Fokus liegt auf der Anwendung in einem Balkonumfeld, wobei mehrere Pflanzen in einem gemeinsamen Pflanzkasten untergebracht sind.

Das System erkennt mithilfe von Bodenfeuchtesensoren, wann die Pflanzen Wasser benötigen. Eine Steuereinheit (Microcontroller) wertet die Sensorwerte aus und aktiviert bei Bedarf eine Pumpe, um Wasser aus einem externen Behälter gezielt zu den Pflanzen zu leiten. Jede Pflanze kann individuell versorgt werden. Die Benutzer können über eine App den aktuellen Zustand der Pflanzen überwachen (z. B. Feuchtigkeitswerte) und grundlegende Einstellungen vornehmen.

Die gesamte Technik ist platzsparend im oder am Pflanzkasten untergebracht. Dabei wird darauf geachtet, dass Schläuche und Kabel ordentlich verlegt und ggf. verkleidet werden. Das System ist modular aufgebaut, sodass es zukünftig problemlos erweitert oder an andere Pflanzkästen angeschlossen werden kann.

Ziel ist es, die Pflanzenpflege zu erleichtern, den Wasserverbrauch zu optimieren und die Bedienung auch für technisch weniger erfahrene Nutzer möglichst einfach zu gestalten.

3 Hauptziele

Tabelle 2: Hauptziele

#	Ziel	Beschreibung der Implementation
1	Automatisierte Bewässerung	Feuchtigkeitssensoren erkennen den Bedarf, Pumpen und Ventile reagieren automatisch
2	Einfache Bedienung	Intuitive App-Oberfläche (siehe GUI-Skizze) für Android und IOS Einfache Inbetriebnahme und automatisierte Abläufe
3	Kostengünstiges System	Verwendung günstiger Komponente, wie z.B. Microcontroller, einfache Sensoren und 3D-gedruckte Teile
4	Zuverlässige Erkennung	Pro Pflanze ein Sensor zur genauen Feuchtigkeitsmessung
5	Anzeige der Daten über App	Entwicklung einer App, um Zustandsdaten grafisch darzustellen
6	Modularität und Erweiterbarkeit	Microcontroller erlaubt Anbindung weiterer Pflanzkästen oder Sensoren

4 Annahmen und Abgrenzungen

Tabelle 3: Annahmen und Abgrenzungen

#	**Annahmen (fachliche und technische Annahmen)**
1	Nutzer hat Zugriff zu WLAN oder Bluetooth (App-Kommunikation)
2	System wird in einem geschützten Außenbereich installiert
3	Dauerhafte Stromversorgung (z.B. über USB (evtl. Solar))
4	Pflanzen haben ähnlichen Wasserbedarf
5	Nutzer befüllt Wasserbehälter regelmäßig (manuell)

#	**Abgrenzungen (Was ist in dieser Lösung nicht enthalten bzw. abgedeckt)**
1	Gehäuse schützt vor Spritzwasser, aber nicht vor starkem Regen oder direktem UV-Licht
2	Keine Düngung o.ä.
3	System trifft Entscheidungen rein regelbasiert, keine Bilderkennung oder anderes KI-gestütztes Monitoring
4	Keine autonome Wasserbefüllung
5	App verfügt über keine Datenweiterleitung in z.B. Cloud

5 Workflow

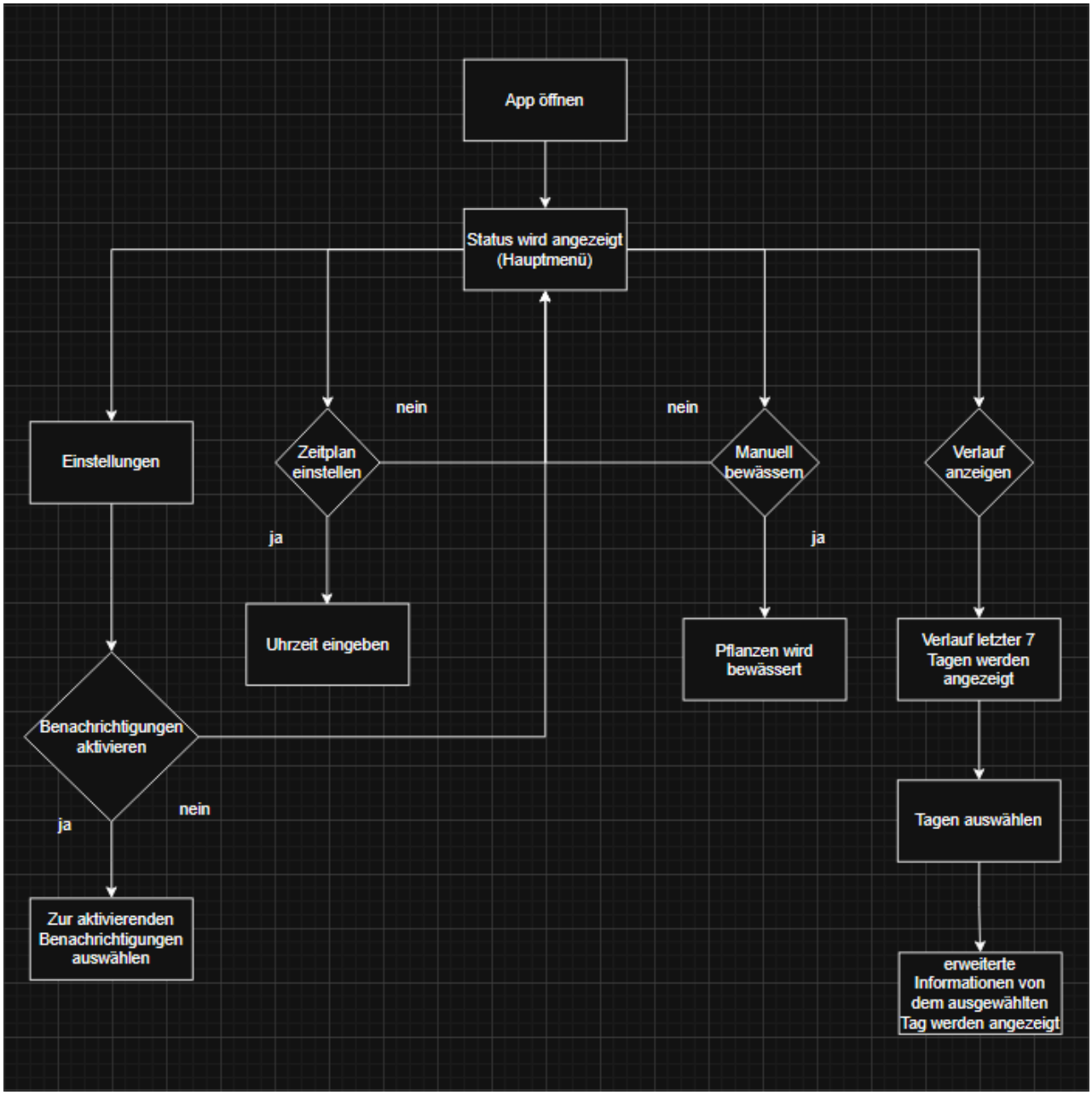

Abbildung 1: Workflow

6 Funktionalität

6.1 Überblick

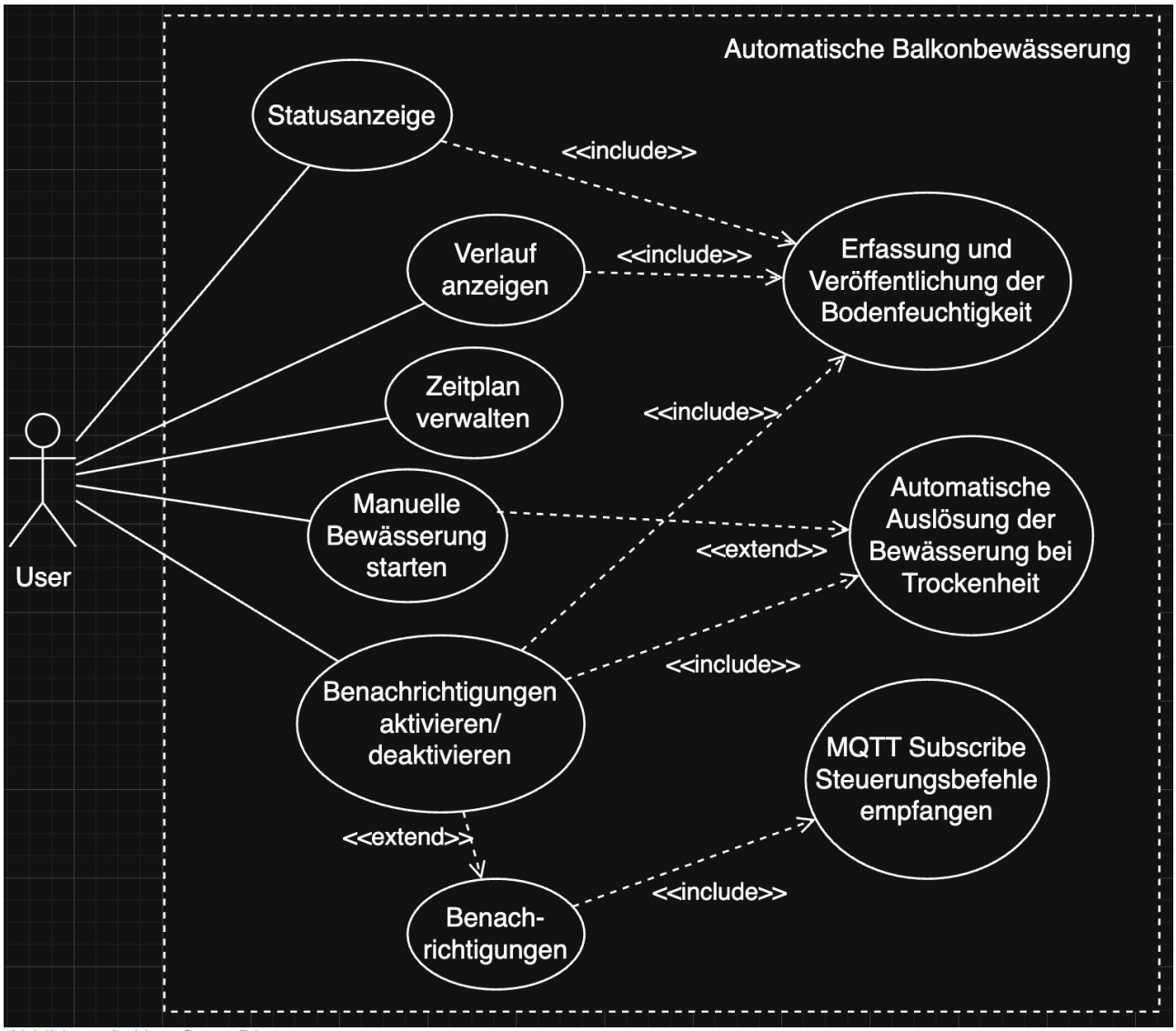

Abbildung 2: Use-Case-Diagramm

6.2 Statusanzeige

Tabelle 4: Statusanzeige

Zweck/Ziel	Anzeige von aktuellen Sensorwerten: Bodenfeuchtigkeit, Wasserstand, letzte Bewässerung.
Akteur/Auslöser	Benutzer beim Öffnen der App
Berechtigung	Keine Anmeldung nötig
Vorbedingung	1. Sensorwerte müssen empfangbar sein 2. Verbindung zum Mikrocontroller besteht
Daten-Input	Werte der Sensoren (z. B. Bodenfeuchtigkeit: 45 %, Wasserstand: 75 %)
Verarbeitungsschritte	1. App empfängt Sensorwerte vom Mikrocontroller 2. Werte werden visuell in der Startansicht angezeigt
Ergebnis	Live-Status der Pflanzenpflege ist sichtbar
Plausibilitäten	1. Feuchtigkeit: 0–100 % 2. Wasserstand: 0–100 % 3. Letzte Bewässerung: heutiges Datum ≤ aktuelles Datum
Fehlerhandling	Bei fehlender Verbindung: Anzeige „Keine Verbindung“
Folgeprozess	1. Möglichkeit zur manuellen Bewässerung 2. Trigger für automatische Benachrichtigung bei kritischen Werten
Out of Scope	1. Historische Datenanalyse 2. GPS-/Standortinformationen
Test Cases	1. Sensorwert < 10 % → Warnmeldung 2. Verbindung unterbrochen → Offline-Anzeige erscheint

6.3 Manuelle Bewässerung starten

Tabelle 5: Manuelle Bewässerung starten

Zweck/Ziel	Direktes Auslösen der Pumpe durch den Nutzer
Akteur/Auslöser	Benutzer klickt auf „Jetzt bewässern“
Berechtigung	Nutzer mit App-Zugriff
Vorbedingung	Verbindung zur Pumpe via Mikrocontroller
Daten-Input	Buttonklick + voreingestellte Menge (z. B. 150 ml)
Verarbeitungsschritte	1. Klicksignal → Mikrocontroller 2. Pumpe wird aktiviert 3. Protokollierung im Verlauf
Ergebnis	Pumpe läuft und Pflanzen werden bewässert
Plausibilitäten	1. Max. Menge: 300 ml 2. Mindestintervall: 30 Min. Abstand zwischen 2 Bewässerungen
Fehlerhandling	1. Kein Wasser: Push-Benachrichtigung 2. Timeout: Hinweis „Pumpenfehler“
Folgeprozess	Eintrag im Verlauf, ggf. Statusaktualisierung
Out of Scope	1. Teilweise Zonenbewässerung 2. Mengenregelung durch Benutzer
Test Cases	1. Bewässerung erfolgreich → Protokolleintrag erscheint 2. Kein Wasser → App zeigt Warnmeldung

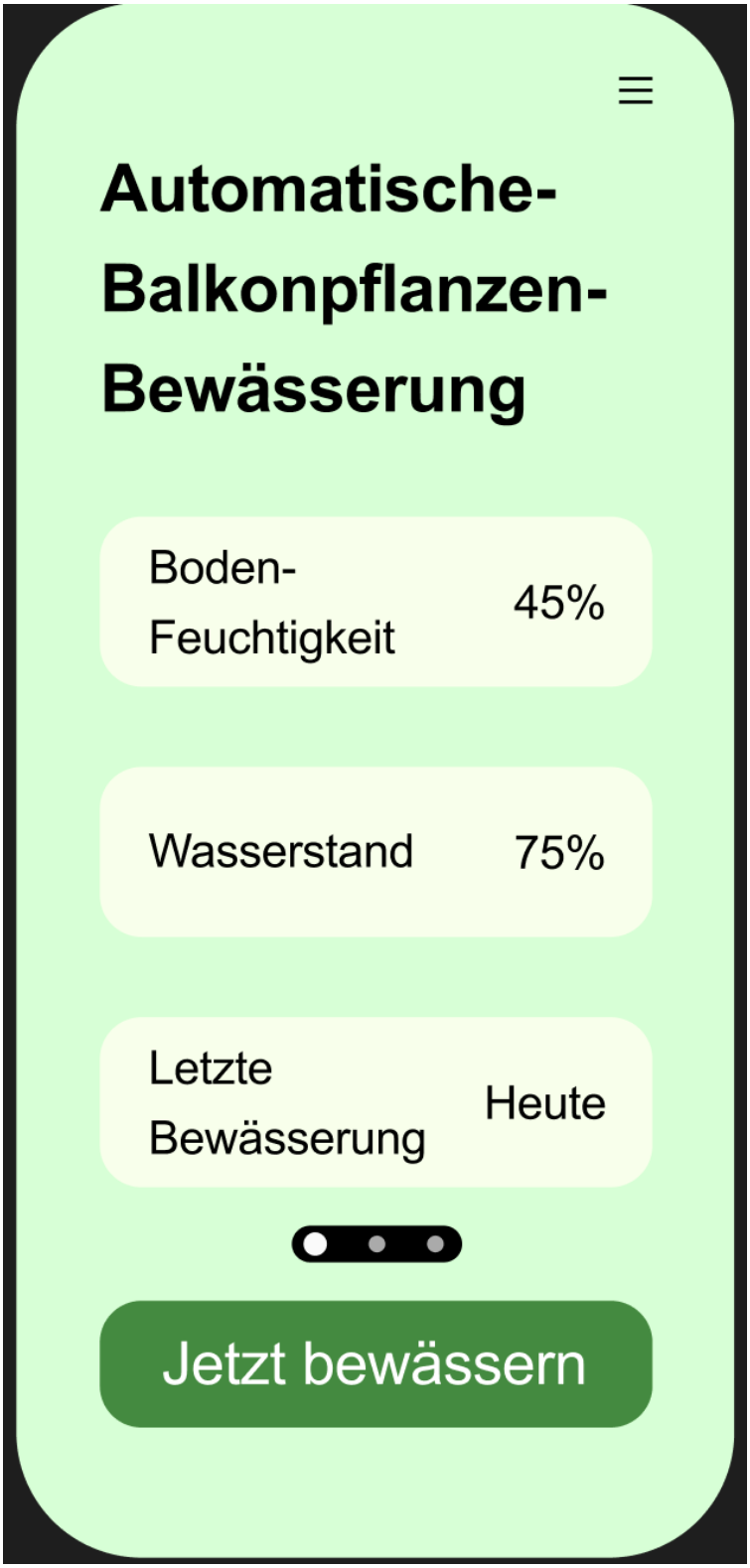

Abbildung 3: Hauptmenü

6.4 Verlauf anzeigen

Tabelle 6: Verlauf anzeigen

Zweck/Ziel	Tägliche Protokollierung von Sensorwerten und Aktionen
Akteur/Auslöser	Benutzer navigiert zu „Verlauf“
Berechtigung	Alle Benutzer
Vorbedingung	Mindestens 1 Eintrag im Verlauf vorhanden
Daten-Input	Tagesauswahl in den letzten 7 Tagen (Mo–So)
Verarbeitungsschritte	1. Datenabruf aus interner Datenbank 2. Anzeige als Liste chronologisch
Ergebnis	Tagesansicht mit Ereignissen
Plausibilitäten	Einträge sortiert nach Zeit
Fehlerhandling	Keine Daten: Anzeige „Keine Ereignisse vorhanden“
Folgeprozess	ggf. Verweis auf die direkt folgenden Prozesse bzw. Funktionen im Workflow
Out of Scope	1. Exportfunktion 2. Statistik oder Diagramm
Test Cases	1. Klick auf Freitag → zeigt 5 Aktionen korrekt 2. Kein Eintrag → „Keine Daten“ wird angezeigt

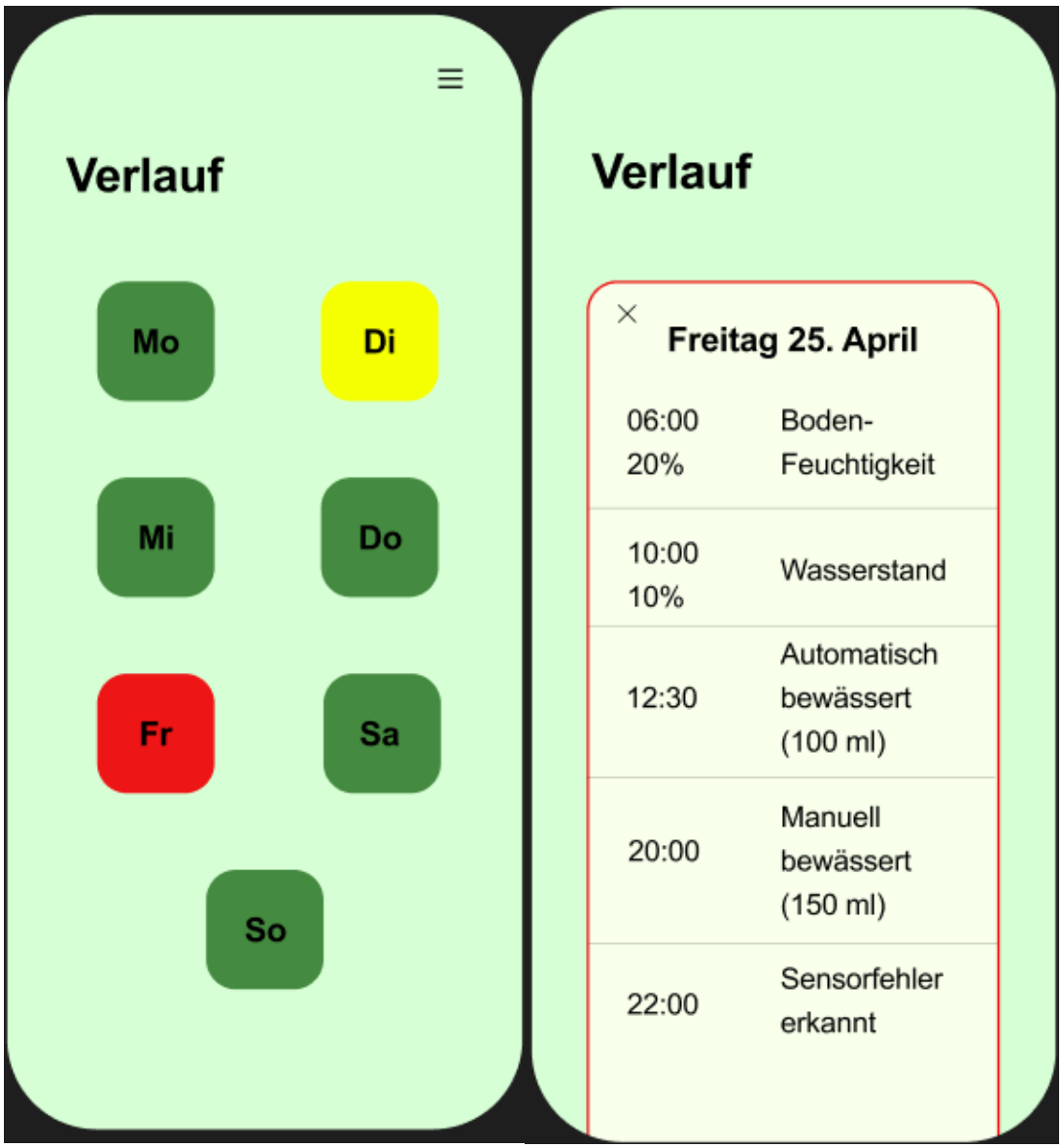

Abbildung 4: Verlauf anzeigen

Tabelle 7: Zeitplan verwalten

6.5 Zeitplan verwalten (als Backup)

Zweck/Ziel	Festlegen einer täglichen Uhrzeit für automatische Bewässerung
Akteur/Auslöser	Benutzer aktiviert Zeitfunktion
Berechtigung	Alle Benutzer
Vorbedingung	App ist mit System verbunden
Daten-Input	Uhrzeit (z. B. 08:00)
Verarbeitungsschritte	1. Auswahlzeit in UI 2. Speicherung im Mikrocontroller 3. Auslösung zur definierten Zeit
Ergebnis	Bewässerung erfolgt automatisch zur festgelegten Uhrzeit
Plausibilitäten	1. Gültige Uhrzeitformate (hh:mm) 2. Nur eine aktive Zeit erlaubt
Fehlerhandling	1. Keine Speicherung möglich → Fehlermeldung 2. System nicht erreichbar → Warnung
Folgeprozess	Automatische Pumpe + Eintrag im Verlauf
Out of Scope	1. Mehrere Zeiten 2. Wochenplan
Test Cases	1. Zeit auf 08:00 gesetzt → Bewässerung erfolgt korrekt 2. Uhrzeit fehlerhaft → Speicherung verweigert

Abbildung 5: Zeitplan verwalten

6.6 Benachrichtigungen aktivieren/deaktivieren

Tabelle 8: Benachrichtigungen aktivieren/deaktivieren

Zweck/Ziel	Nutzer erhält Push-Nachrichten bei Ereignissen
Akteur/Auslöser	Benutzer schaltet Benachrichtigungen-Einstellung ein
Berechtigung	Alle Benutzer
Vorbedingung	1. Benachrichtigungen sind im Gerät und in der App erlaubt 2. WLAN-/Bluetooth-Verbindung besteht (?)
Daten-Input	Fehlercodes, Sensordaten, Pumpenstatus
Verarbeitungsschritte	1. Sensor meldet kritischen Zustand oder Systemereignis 2. Mikrocontroller analysiert Daten 3. Ereignis wird als Push-Nachricht übermittelt
Beanchrichtigungstypen	Kategorie: Nachricht (Beispieltext) 1. Feuchtigkeit: „Achtung! Bodenfeuchtigkeit unter 20 %. Gießen empfohlen.“ 2. Wasserstand: „Wassertank fast leer. Bitte nachfüllen.“ 3. Sensorfehler: „Fehler: Bodenfeuchtesensor nicht erreichbar.“ 4. Pumpenfehler: „Warnung: Pumpe konnte nicht aktiviert werden.“ 5. Automatische Aktion: „Automatische Bewässerung wurde um 08:00 Uhr erfolgreich durchgeführt.“ 6. Manuelle Aktion: „Manuelle Bewässerung wurde ausgeführt (150 ml).“ 7. System online/offline: „Verbindung zum Bewässerungssystem wiederhergestellt.“ oder „Verbindung verloren.“
Ergebnis	1. Der Nutzer wird zeitnah über den Zustand des Systems informiert 2. Möglichkeit, über App oder manuell zu reagieren
Plausibilitäten	1. Max. eine Benachrichtigung pro Ereignis alle 5 Minuten 2. Nur bei tatsächlichem Problem (z. B. Schwellenwert unterschritten)
Fehlerhandling	Push fehlgeschlagen → App zeigt Hinweis
Folgeprozess	Je nach Typ: Nutzer kann System prüfen, Wasser auffüllen, manuell starten etc.
Out of Scope	1. E-Mail-Versand 2. Fehlerbehebung direkt in App
Test Cases	1. Sensorwert < 20 % → Push-Benachrichtigung innerhalb 30 Sekunden 2. Tank leer → Nutzer erhält Warnmeldung mit Aufforderung zum Nachfüllen 3. Pumpe reagiert nicht → App zeigt „Pumpenfehler“ 4. Manuelle Aktion → Push bestätigt erfolgreiche Ausführung 5. System offline → App meldet Verbindungsfehler

Abbildung 6: Benachrichtigungen aktivieren

6.7 Erfassung und Veröffentlichung der Bodenfeuchtigkeit

Tabelle 9: Erfassung und Veröffentlichung der Bodefeuchtigkeit

Zweck/Ziel	Automatisches Erfassen der Bodenfeuchte und Veröffentlichung per MQTT, um die aktuellen Messwerte der Pflanze für die App/Dashboard bereitzustellen.
Akteur/Auslöser	ESP und MQTT
Vorbedingung	WLAN- und MQTT-Verbindung aktiv, Sensor korrekt angeschlossen
Daten-Input	Analogsignal vom Bodenfeuchtigkeitssensor
Verarbeitungsschritte	1. Sensorwert einlesen 2. Zeitstempel erzeugen 3. JSON-Nachricht erzeugen 4. MQTT-Nachricht an einem bestimmtenTopic senden
Ergebnis	Feuchtigkeitswert im MQTT-Broker verfügbar, App kann anzeigen
Plausibilitäten	Gültiger Wertebereich (z. B. 0–4095), ungewöhnliche Werte werden gekennzeichnet
Fehlerhandling	0 oder ungültige Werte → als Fehler kennzeichnen MQTT offline → reconnect-Mechanismus Fallback: Retry nach Zeitintervall
Folgeprozess	Nächster Erfassungszyklus wird gestartet (z. B. alle 3 Min.)
Test Cases	- Sensor liefert gültigen Wert → MQTT publiziert - Sensor fehlt → Fehlerstatus - MQTT nicht erreichbar → reconnect - JSON-Format korrekt

6.8 Automatische Auslösung der Bewässerung bei Trockenheit

Tabelle 10: Automatische Auslösung der Bewässerung bei Trockenheit

Zweck/Ziel	Automatische Bewässerung, wenn der Boden zu trocken ist
Akteur/Auslöser	ESP und Wasserpumpen
Vorbedingung	- Sensorwert unter Grenzwert - Kein aktiver Sperrzeit-Timer - Stromversorgung verfügbar
Daten-Input	Letzter Sensorwert, Schwellwert, Systemzeit
Verarbeitungsschritte	1. Vergleich mit Schwellenwert 2. Aktivierung Relais (Pumpe) 3. Einstellbare Deaktivierungszeit 4. MQTT-Statusnachricht
Ergebnis	Pflanze wird gegossen
Plausibilitäten	Relais-Zeitfenster z. B. 10s Vermeidung von Dauerbetrieb
Fehlerhandling	Relaisausfall → Fehlerausgabe Fehlerhafte Messung → keine Auslösung Sperrzeit → automatische Blockade
Folgeprozess	Neue Sperrzeit startet System wartet auf nächsten Sensorzyklus
Test Cases	- Sensorwert unter Grenzwert → Pumpe aktiv - Sperrzeit aktiv → keine Pumpe - MQTT sendet Statusmeldung - Pumpe läuft maximal x Sekunden

6.9 MQTT Publish – Sensor- & Statusdaten senden

Tabelle 11: MQTT Publish - Sensor- & Statusdaten senden

Zweck/Ziel	Der ESP sendet Sensorwerte und Statusmeldungen über MQTT an spezifische Topics, damit externe Systeme (App, Dashboard) Daten empfangen können.
Akteur/Auslöser	Intern: nach Messung, nach Pumpenvorgang, bei Fehlern
Berechtigung	ESP32 besitzt Publish-Rechte (MQTT-Login mit Schreibzugriff)
Vorbedingung	Aktive WLAN- & MQTT-Verbindung, gültiger Topic-Name
Daten-Input	Sensordaten, Systemstatus, Zeitstempel
Verarbeitungsschritte	1. JSON-Nachricht erzeugen 2. Verbindung zur MQTT bestimmen. 3. Nachricht an festgelegtes Topic senden:
Ergebnis	Daten im Broker verfügbar für alle Subscriber (z. B. App)
Plausibilitäten	Nachrichten müssen gültiges JSON enthalten Topics müssen existieren bzw. zulässig sein MQTT-Verbindung muss richtig konfiguriert sein
Fehlerhandling	Kein Brokerkontakt → reconnect Fehlerhafte Daten → verworfen und geloggt
Folgeprozess	Nächster Erfassungszyklus wird gestartet (z. B. alle 3 Min.)
Test Cases	- Wert publiziert erfolgreich - JSON lesbar im MQTT-Viewer - Fehlerfall: Nachricht nicht gesendet - Reconnect funktioniert korrekt

6.10 MQTT Subscribe – Steuerungsbefehle empfangen

Tabelle 12: MQTT Subscribe - Steuerungsbefehle empfangen

Zweck/Ziel	Der ESP empfängt Steuerbefehle (z. B. zur manuellen Pumpensteuerung) über ein MQTT-Topic und führt die gewünschten Aktionen aus.
Akteur/Auslöser	Nutzer sendet MQTT-Befehl an bestimmtes Topic
Berechtigung	MQTT-Client des ESP ist mit Leseberechtigung verbunden
Vorbedingung	WLAN- & MQTT-Verbindung aktiv, Topic korrekt abonniert
Daten-Input	MQTT-Nachricht (JSON-Datei) von ESP
Verarbeitungsschritte	1. ESP ist Subscriber von einem bestimmten Topic 2. Nachricht wird empfangen 3. Inhalt geprüft und interpretiert 4. Pumpe aktiviert oder deaktiviert 5. Rückmeldung über Status-Topic
Ergebnis	Steuerbefehl wird ausgeführt, Status wird rückgemeldet
Plausibilitäten	Nur vordefinierte Nachricht akzeptiert (bei Pumpensteurung nur “on” und “off”)
Fehlerhandling	Ungültiger Inhalt → ignorieren & loggen Verbindungsabbruch → automatisch neuverbinden
Folgeprozess	Zustand bleibt bestehen bis neuer Befehl oder nächste automatische Aktion
Test Cases	- "on": Pumpe läuft - "off": Pumpe stoppt - Falscher Befehl → ignoriert - Neuverbinden nach Trennung - Kontrolle über App sichtbar

2020