

Dokumentation Mikrocontroller

Gruppe:

04

Datum:

17.01.2026

Teammitglieder:

Christian Fischer

David Schüring



GREEN HUB

GREEN GARDENING

Inhalt

Einleitung.....	3
Arbeitsverteilung	4
Beschreibung der Lösung.....	5
Elektrischer-Schaltplan	7
Softwaretechnischer Schaltplan	Fehler! Textmarke nicht definiert.
Software:	Fehler! Textmarke nicht definiert.
Code.....	Fehler! Textmarke nicht definiert.
Fortschritte	12
Reflexion	13
Zusammenfassung.....	14

Einleitung

Das Projekt Green Gardening entstand im Rahmen des Fachs Mikrocontroller mit dem Ziel, die Bewässerung des Hochschulgartens automatisiert, ressourcenschonend und datenbasiert zu gestalten. Unsere Teams arbeiteten in vier Untergruppen — Pumpe, Stromversorgung, Sensorik und Software — und entwickelten die Hardware und Software modular, um Wartung und Erweiterung zu erleichtern. Technisch basiert das System auf einem vernetzten Ökosystem aus Mikrocontrollern (ESP-Module), einem zentralen Web-Server mit Datenbank und einer benutzerfreundlichen Weboberfläche. Sensoren erfassen Bodenfeuchte und weitere Umweltdaten; ein Host-ESP sammelt Messwerte über ESP-Now/Bluetooth-Netzwerke und überträgt sie an den Server. Die Weboberfläche visualisiert Messwerte, ermöglicht gesicherte Konfigurationsänderungen und steuert Bewässerungszeitpunkte ohne zusätzliche Programmierung. Durch die datenbasierte Steuerung reduziert das System den Wasserverbrauch, erleichtert die Pflege des Gartens und schafft eine Grundlage für zukünftige Erweiterungen sowie für Lehr- und Forschungszwecke.

Arbeitsverteilung

Teammitglieder:

Christian Fischer

David Schüring

1. Projektleitung: Christian Fischer

Aufgaben:

- ESP-Kommunikationsprotokoll implementieren
- Web-Frontend Anbindung
- Durchführung von Tests

2. Testumgebung und Dokumentation: David Schüring

Aufgaben:

- Erstellung der Dokumentation und des Videos
- Visuelles Feedback über LEDs (Status- und Aktivitätsanzeige)

Beschreibung der Lösung

1. Architektur und Komponenten

Das System besteht aus mehreren ESP-Geräten mit klaren Rollen: Sensor-Clients messen und senden Daten, Pumpen-Clients empfangen Steuerbefehle, und ein Server-ESP vermittelt zwischen dem lokalen Netzwerk und dem Web-Server. Der Web-Server hostet die Datenbank und das Frontend; dort laufen Authentifizierung, Konfigurationsverwaltung und Historisierung der Messwerte. Konfigurationen werden per API-Token gesichert übertragen; Geräte speichern Einstellungen lokal im Flash, um nach Neustarts oder Ausfällen autonom weiterzuarbeiten.

2. Kommunikationsfluss und Protokolle

Die Geräte kommunizieren lokal über ESP-Now/Bluetooth und nutzen den Server-ESP als Gateway zum Internet. Bei Konfigurationsänderungen informiert die API den Server-ESP, der betroffene Sensoren zur Abfrage neuer Einstellungen anstößt.

3. Energieeffizienz und Ausfallsicherheit

Um Energie zu sparen, koordinieren die Mikrocontroller ihre Kommunikationsfenster und verbleiben außerhalb dieser Intervalle im DeepSleep. Lokale Pufferung auf dem Server-ESP und kontrolliertes Löschen ältester Einträge verhindern Datenverlust bei kurzzeitigen WLAN-Ausfällen. RTC-gespeicherte Zustände und Flash-Persistenz sorgen dafür, dass Geräte nach Stromausfällen oder DeepSleep-Phasen wieder konsistent arbeiten. Retry-Limits, Zeitstempel-Synchronisation (NTP) und Monitoring-Logs unterstützen die Fehlerdiagnose.

4. Bedienung, Wartung und Nutzen

Die Weboberfläche bietet eine einfache, passwortgeschützte Bedienoberfläche zur Anzeige historischer Messwerte, Anpassung von Mess- und Bewässerungsintervallen sowie zur Überwachung des Systemzustands. Zusätzlich ermöglicht die Oberfläche ein einfaches Hinzufügen weiterer ESP-Geräte, Pumpen und Sensoren: neue Geräte lassen sich über ein Formular registrieren, mit Namen und Typ versehen. Für jedes Gerät können zeitliche Abläufe konfiguriert werden, etwa Messintervalle für Sensoren und Schaltzeiten oder Verzögerungen für Pumpen. Die Oberfläche zeigt den aktuellen Status der Pumpe mit einem Zeitstempel. Auf der Administrationsseite, auf der neue Geräte hinzugefügt werden können, lassen sich Pumpen auch zeitgesteuert aktivieren – wahlweise in Minuten- oder Stundenintervallen oder zu wiederkehrenden Tageszeiten. Der Vorteil liegt in einem reduzierten Wasserverbrauch, einem geringeren Pflegeaufwand, einer einfacheren Erweiterbarkeit, einer besseren Planbarkeit und einer soliden Grundlage für Lehre und Forschung

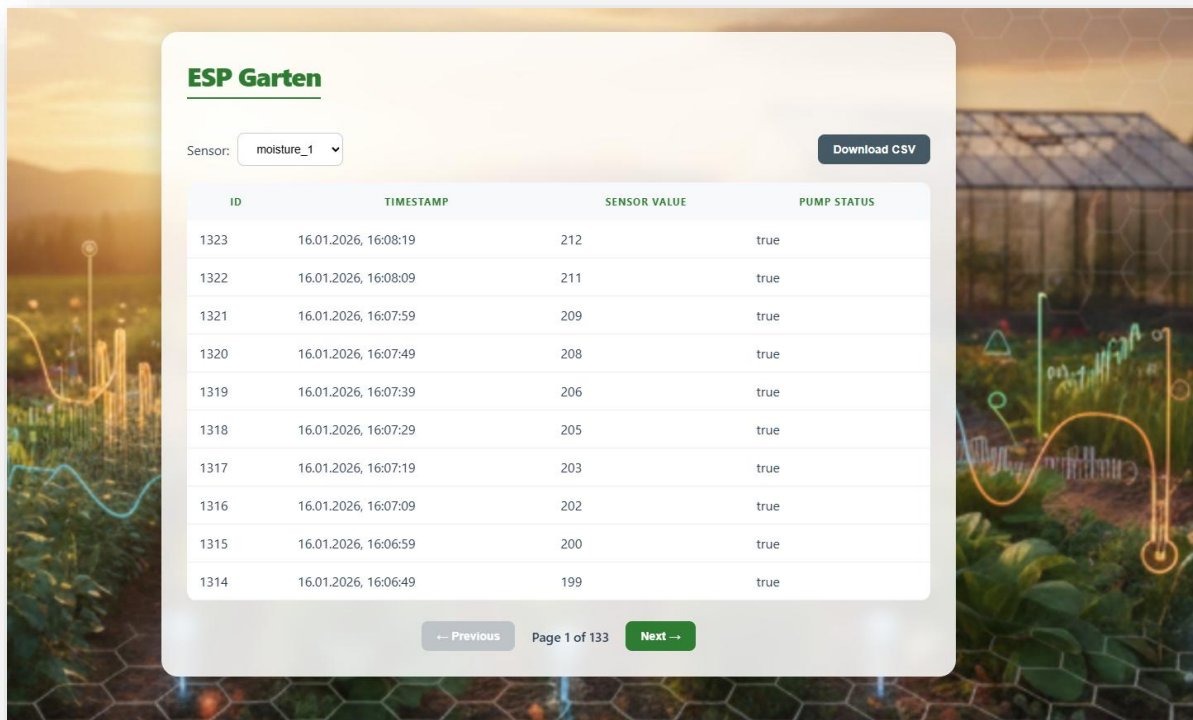


Abbildung 1: Dashboard-Ansicht des ESP-Garten-Systems mit Live-Daten zur Bodenfeuchte und Pumpenstatus

Elektrischer-Schaltplan

Zur visuellen Rückmeldung während der Demonstration werden zwei Leuchtdioden direkt über die GPIO-Pins des ESP32-C3 angesteuert. Die LEDs dienen ausschließlich als Statusanzeige für den Pumpenbetrieb und ermöglichen es, den Systemzustand ohne zusätzliche Software-Ausgabe sofort zu erkennen.

GPIO	LED-Farbe	Zustand der Pumpe aus	Zustand der Pumpe an
GPIO8	Rot	LED aus	LED an
GPIO10	Grün	LED aus	LED an

Die Status-LEDs sind jeweils über einen 330 Ω Vorwiderstand mit den GPIO-Pins des ESP32-C3 verbunden:

- GPIO 8 $\rightarrow$ rote LED
- GPIO 10 $\rightarrow$ grüne LED

Jede LED ist in Reihe mit dem Widerstand geschaltet und anschließend gegen GND verbunden. Der Widerstand begrenzt den Strom und schützt die LED vor Überlastung. Die Ansteuerung erfolgt direkt über die digitalen Ausgänge des Mikrocontrollers.

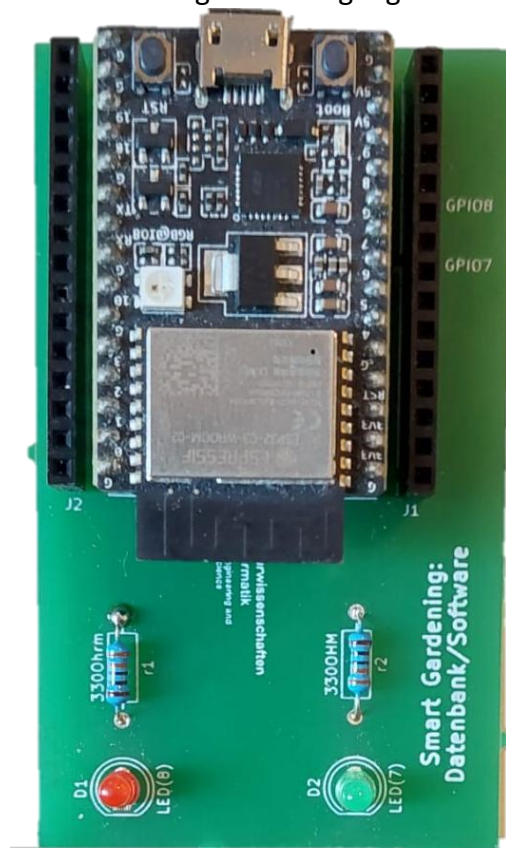


Abbildung 2: Schaltplan 1

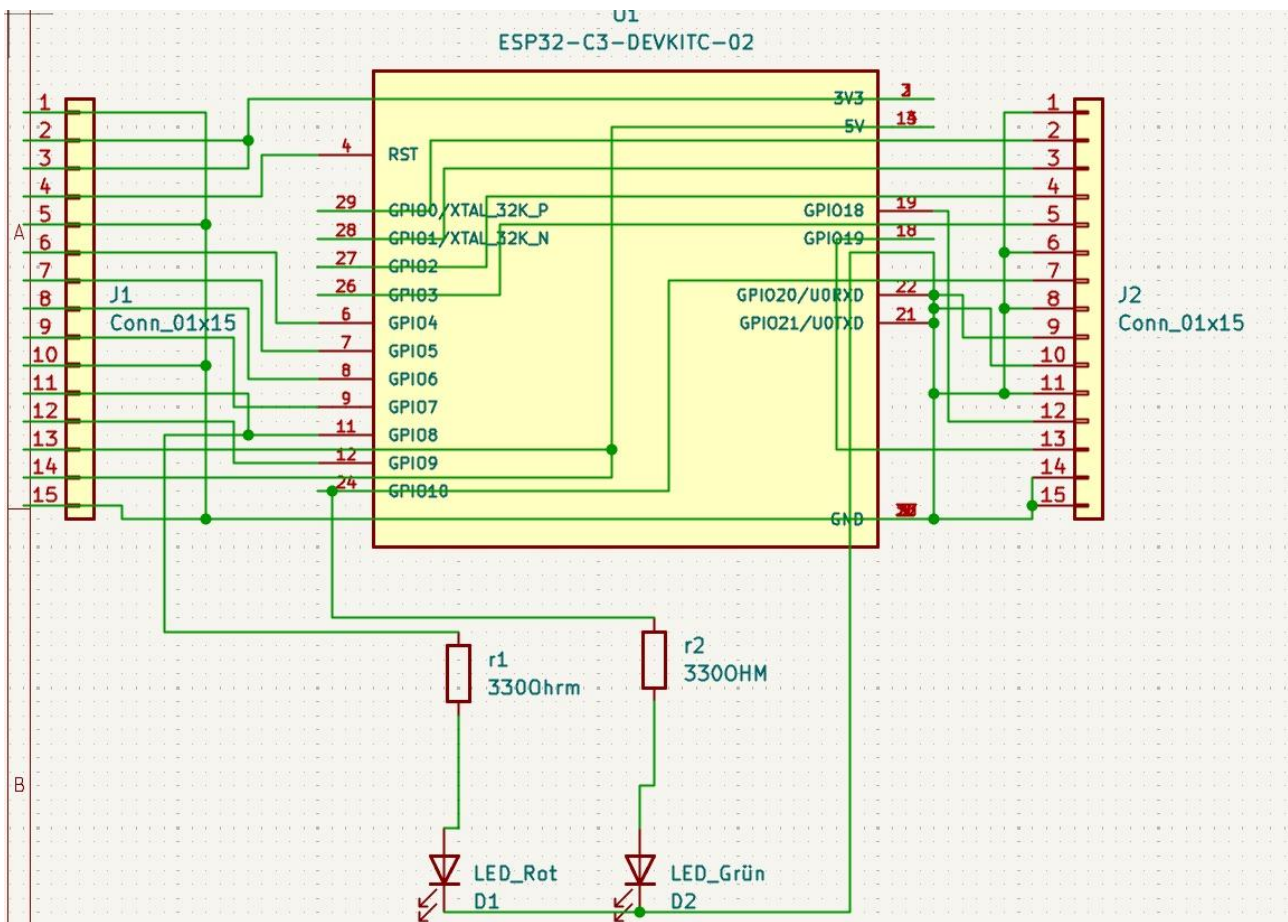


Abbildung 3:Schaltplan 2 $R1=R2=330\Omega$

Funktionsbeschreibung

1. Initialisierung und Zeit-Sync

Beim Start verbindet sich der ESP-Server mit dem Netzwerk, synchronisiert die Uhrzeit per NTP und fordert die aktuellen Konfigurationen vom Web-Server an. Parallel lädt der ESP-Sensor seine zuletzt gespeicherte Konfiguration aus dem Flash und sendet per ESP-NOW eine Konfigurationsanfrage an den Server, damit er die aktuelle Zeit und mögliche Änderungen erhält.

2. Konfigurationsverteilung

Der ESP-Server antwortet auf Anfragen zunächst mit der aktuellen Zeit und, falls nötig, mit einer Liste geänderter Konfigurationsnamen. Liegen Änderungen vor, überträgt der Server die vollständige Konfiguration an den Sensor. Der Sensor speichert empfangene Konfigurationen dauerhaft im Flash und passt Mess- und Sendeintervalle entsprechend an.

3. Messdatenfluss

Der ESP-Sensor führt Messungen durch und sendet die Messwerte per ESP-NOW an den ESP-Server. Der Server empfängt diese Daten und leitet sie per HTTP(S) an den Web-Server weiter, welcher die Messwerte darstellt. Der Web-Server kann bei Bedarf geänderte Konfigurationsnamen zurückliefern, die der Server dann an die Sensoren verteilt.

4. Pumpensteuerung

Wenn der Sensor aufgrund eines Schwellenwerts oder einer Regelung eine Pumpe aktivieren muss, erzeugt er einen Pumpenbefehl mit Angaben wie Pumpenname, Verzögerung, aktuellem Zeitstempel und Zeitpunkt der nächsten Anfrage und sendet diesen per ESP-NOW an die ESP-Pumpe. Die Pumpe führt den Steuerbefehl aus, kann optional eine Bestätigung zurücksenden und aktiviert zur visuellen Rückmeldung die Status-LEDs.

5. DeepSleep und Energieverwaltung

Alle Geräte regeln ihr Schlafverhalten nach den nächsten erwarteten Ereignissen: der Sensor schläft bis zur nächsten Messung, die Pumpe bis zur nächsten Anfrage, und der Server darf erst schlafen, nachdem er Initialkonfiguration und Zeit an die Sensoren übermittelt hat; er wacht spätestens eine Minute vor dem nächsten erwarteten Anfragezeitpunkt auf. Diese Regeln sorgen für Energieeffizienz und koordinierte Abläufe.

6. Fehlerfall und Datenpufferung

Fällt die Wi-Fi-Verbindung aus, puffert der ESP-Server Messdaten lokal und versucht später, sie an den Web-Server zu senden; bei vollem Speicher werden die ältesten Einträge gelöscht. Der ESP-Sensor nutzt im Fehlerfall seine im Flash gespeicherte Konfiguration weiter und arbeitet autonom nach den zuletzt bekannten Parametern.

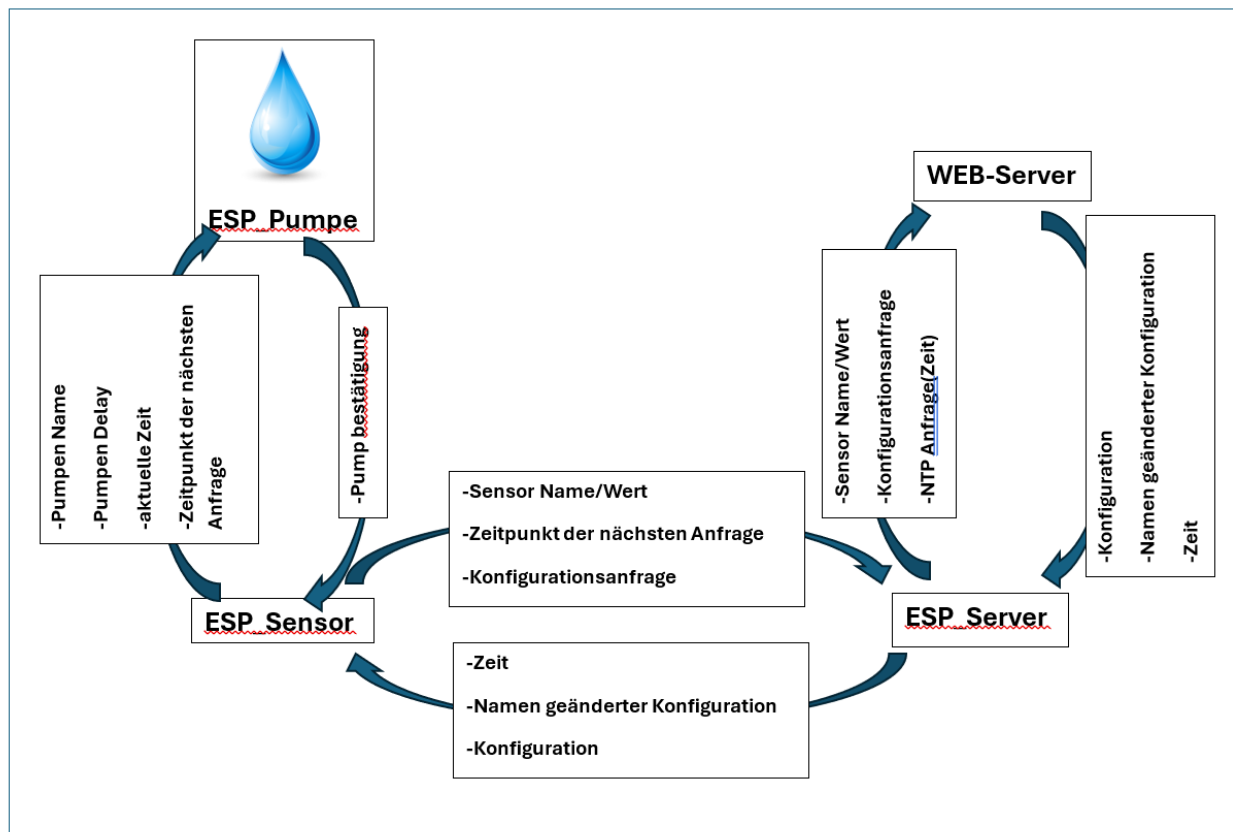


Abbildung 4; Visuelle Funktionsbeschreibung

Software

Server Software

Die Server-Software fungiert als Gateway zwischen dem lokalen ESP-Now-Netzwerk und dem entfernten Web-Server. Dieser empfängt Sensordaten von Sensor-ESP, puffert diese lokal, bündelt sie und sendet sie über HTTPS an die zentrale Datenbank; gleichzeitig beantwortet sie Zeit- und Konfigurationsanfragen der Clients und leitet neue Konfigurationsnamen an den Sensor-ESPs weiter. Wichtige Funktionen sind NTP-/SNTP-Zeitabgleich, HTTP-Client-Kommunikation und eine Sleep-Planung, die das Gerät energieeffizient in DeepSleep versetzt, bis die nächste Verbindung nötig ist. Die Implementierung ist modular gehalten und enthält klare Stellen für Fehler- und Retry-Handling sowie für das Parsen/Erzeugen der Nachrichtenformate.

Pumpen-Software

„Die Pumpen-Software steuert die Pumpen und sorgt für ein zuverlässiges, vorhersehbares Schalten. Sie empfängt per ESP-Now Start- und Stopp-Signale, bestätigt Empfang mit einfachen ACK-Nachrichten und setzt GPIOs über einen kleinen Assembler-Routine, um Latenz und Timing zu minimieren. Die Logik umfasst Zustandsübergänge (Start, Running, Finished), Zeitstempel für Laufzeiten, einfache Sicherheitschecks. Die Kombination aus C++-Steuerung und gezieltem Assemblerzugriff ist bewusst gewählt, um zuverlässige Hardwaresteuerung mit überschaubarem Code-Footprint zu verbinden.

Sensor-Software

Die Sensor-Software misst Umgebungsdaten, verwaltet lokale Konfigurationen und steuert die Kommunikation mit Pumpen und Server. Sensoren fragen initial Konfigurationen an, speichern diese dauerhaft in NVS/Flash und nutzen RTC-Attribute für kurzlebige Zustände über DeepSleep-Zyklen hinweg; Messintervalle werden entweder zeitbasiert oder nach konfigurierten Intervallen geplant. Beim Senden folgt die Komponente einer festen Reihenfolge: Zeit-Sync, Senden an Pumpen (mit ACK/Retry), dann Senden an den Server; bei Verbindungsproblemen puffert sie Daten lokal und versucht später erneut. Die Software ist robust gegenüber WLAN-Ausfällen, nutzt NVS-Blobs für Persistenz.

Disclaimer zur Dokumentation

Für tiefergehende Details siehe die kommentierten Stellen und Funktionen im Repository, insbesondere `Server.cpp`, `WiFiManager.cpp`, `ClientSensor.cpp`, `ClientPump.cpp`, `globals.h` / `globals.cpp` und die README-/Sprint-Pläne. Nutze diese Quellen als primäre Referenz; die obigen Texte dienen als zusammenfassende Orientierung.

Fortschritte

Sprint 1: Erforschen der Bluetooth-Möglichkeiten

Sprint 2: Verbinden zweier ESP32 über Bluetooth-Now; Gestaltung der Platine und Bestückung (Platinendesign / Assembler)

Sprint 3: ESP32 → Web-Server Connection inklusive Web-Page & Datenbank

Sprint 4: ESP32 → Web-Server mit Verbindung von Bluetooth-Now

Sprint 5: Implementierung der Konfigurationsübertragung

Sprint 6: Cleaning Code; Zusammenbau (Hardware-Integration)

Sprint 7: DeepSleep implementieren

Sprint 8: Anbindung der ESP32-Pumpe

Sprint 9: Eventuell: ESP-Server Code für ESP-IDF umbauen

Sprint 10: Cleaning Code; Dokumentation

Sprint 11: Planen und Einbinden der Stromdaten

Sprint 12: Bugfixing (selten wird die Pumpe nicht erreicht)

Sprint 13: Webpage verbessern (eine Seite zur Konfiguration); Video

Sprint 14: Software auf finale ESP schreiben und Fake-Daten durch echte Werte ersetzen; Dokumentation; Video

Sprint 15: Kalibrierung für Sensoren einarbeiten; Dokumentation; Video

Sprint 16: Cleaning Code; Dokumentation; Video

Sprint 17: Dokumentation schreiben (umfangreich); Video

Sprint 18: Webpage stylen; Dokumentation; Video

Reflexion

Im Green Gardening Projekt konzentrierte sich die Arbeit auf die Software für drei Firmware-Programme: Sensor, Pumpe und Server. Die Hardware besteht im Wesentlichen aus einer Platine, für visuelles Feedback; alle anderen Funktionen laufen rein softwareseitig auf den ESP-Modulen. Die folgende Reflexion fasst positive Aspekte, Probleme und konkrete Verbesserungen zusammen, damit die Software reproduzierbar, wartbar und robuster wird.

Was hat gut funktioniert?

- **Hardware-Zusammenbau:** Der Zusammenbau der Demonstrations-Platine
- **Teamarbeit:** Die anfänglichen Meetings der Green-Gardening Gruppe

Was hat nicht so gut funktioniert?

- **Integration ins WLAN:** Durch die verschiedenen WLAN-Netzwerke war es schwierig alle 3 Mikrocontroller ins gleiche Netzwerk zu bringen.
- **Kommunikation in und unter den Gruppen:** Durch teilweise fehlende Kommunikation und nicht zustande gekommene Gruppentreffen wurde die Zusammenarbeit erschwert.
- **Platinen Entwicklung:** Beim Entwickeln der Platine war ein GPIO-Pin falsch verbunden, wodurch die Schaltung nicht wie geplant funktioniert hat.

Wie könnte man es beim nächsten Mal besser machen?

- **Kommunikation:** Für zukünftige Projekte wäre ein einheitliches Kommunikationsmedium sowie eine höhere Regelmäßigkeit der Treffen hilfreich, um Abstimmungen effizienter zu gestalten.

Insgesamt sind wir stolz auf das erreichte Ergebnis und die Funktionsfähigkeit unseres Systems. Die gewonnenen Erkenntnisse werden uns in zukünftigen Projekten helfen, effizienter und strukturierter zu arbeiten.

Zusammenfassung

Im Rahmen des Green-Gardening-Projekts haben wir als Gruppe die Software für ein kleines, aber klar strukturiertes IoT-System entwickelt, das aus drei voneinander getrennten Firmwaren-Programmen besteht: Sensor, Pumpe und Server. Die Hardware wurde bewusst minimal gehalten; lediglich die Pumpe verfügt über eine kleine Platine, die visuelles Feedback über LEDs ermöglicht. Der Schwerpunkt des Projekts lag daher eindeutig auf der Softwarearchitektur und der sauberen Trennung der Verantwortlichkeiten zwischen den Modulen. Der Sensor übernimmt die Messung der Bodenfeuchte und enthält zusätzlich die Entscheidungslogik, die bestimmt, ob ein Bewässerungsbedarf besteht. Die Pumpe fungiert als reiner Aktuator: Sie führt die vom Sensor ausgelösten Befehle aus und liefert über ihre LED-Platine Rückmeldungen zum aktuellen Zustand, etwa ob ein Pumpvorgang läuft oder ein Fehler vorliegt. Der Server bildet die zentrale Vermittlungsinstanz im System. Er empfängt die Sensordaten, synchronisiert die Zeit, puffert Nachrichten bei temporären Wi-Fi-Ausfällen und leitet alle relevanten Informationen an das Web-Backend weiter, sodass eine Visualisierung und Steuerung über das Webinterface möglich ist. Aus strategischen Gründen haben wir diese Rollen strikt voneinander getrennt. Der Sensor trifft Entscheidungen autonom, die Pumpe konzentriert sich ausschließlich auf die Ausführung und das Feedback, und der Server sorgt für Kommunikation, Stabilität und Anbindung an das Web. Diese klare Aufgabenteilung erhöht die Robustheit des Gesamtsystems, erleichtert die Fehlersuche und ermöglicht es, jedes Modul unabhängig weiterzuentwickeln oder auszutauschen, ohne die Gesamtarchitektur zu gefährden.