2025

Jerrit Schnaible

BZTG Oldenburg

6.5.2025

IoT-Projekt BloomBuddy

**Ein Bild, das Text, Schrift, Logo, Design enthält.

KI-generierte Inhalte können fehlerhaft sein.**

Inhaltsverzeichnis

[1. Projektbeschreibung 3](#_Toc197434377)

[Kurzbeschreibung: 3](#_Toc197434378)

[Link zum GitHub-Repository: 3](#_Toc197434379)

[https://github.com/jerrit151/300\_Projekt\_BloomBuddy 3](#_Toc197434380)

[2. Anforderungen und Funktionalitäten 3](#_Toc197434381)

[✅ Sensorik 3](#_Toc197434382)

[✅ Aktoren-Steuerung 3](#_Toc197434383)

[✅ Webinterface & Benutzerinteraktion 4](#_Toc197434384)

[3. Benötigte Komponenten 4](#_Toc197434385)

[3.1 Hardware 4](#_Toc197434386)

[3.2 Software & Datenbank 5](#_Toc197434387)

[3.3 Verwendete Software im Detail 5](#_Toc197434388)

[4. Systemarchitektur 5](#_Toc197434389)

[5. Zeitplanung (Meilensteine) 6](#_Toc197434390)

[6. Offene Fragen & Herausforderungen 7](#_Toc197434391)

[Batterie- und Stromprobleme: 7](#_Toc197434392)

[Über- oder Unterbewässerung: 7](#_Toc197434393)

[Ungleichmäßige Wasserverteilung: 7](#_Toc197434394)

[Programmierung: 7](#_Toc197434395)

[7.Technische Besonderheiten 8](#_Toc197434396)

[8. Fazit & Zielsetzung 8](#_Toc197434397)

[Was sollte das Projekt am Ende können: 8](#_Toc197434398)

[Persönliches Fazit: 8](#_Toc197434399)

[9. Zusatzfunktionen 9](#_Toc197434400)

[Bedienungsanleitung – Automatische Pflanzenbewässerung "BloomBuddy" 10](#_Toc197434401)

[1. Systemübersicht 10](#_Toc197434402)

[Funktionen: 10](#_Toc197434403)

[2. Inbetriebnahme und Aufbau 10](#_Toc197434404)

[Benötigte Hardware: 10](#_Toc197434405)

[Benötigte Software: 11](#_Toc197434406)

[Benötigte Bibliotheken in Micropython: 11](#_Toc197434407)

[Vorbereitung: 11](#_Toc197434408)

[Inbetriebnahme: 11](#_Toc197434409)

[3. Funktionen im Detail 12](#_Toc197434410)

[3.1 Sensorik & Messungen 12](#_Toc197434411)

[3.2 Automatische Pumpensteuerung 12](#_Toc197434412)

[3.3 Datenverarbeitung & -übertragung 12](#_Toc197434413)

[3.4 Push-Benachrichtigung & Alarm 12](#_Toc197434414)

[4. Bedienung im Alltag 13](#_Toc197434415)

[Automatikbetrieb: 13](#_Toc197434416)

[Manuelle Steuerung über das Node-Red Dashboard 13](#_Toc197434417)

[Statuskontrolle: 13](#_Toc197434418)

[5. Wartung & Tipps 13](#_Toc197434419)

[6. Störungen & Fehlersuche 14](#_Toc197434420)

[7. Beschreibung des Node-Red-Flows 14](#_Toc197434421)

[Node-Red-Flow: 14](#_Toc197434422)

[Node-Red-Dashboard: 15](#_Toc197434423)

[a. MQTT-Empfang der Sensordaten 15](#_Toc197434424)

[b. Aufbereitung und Visualisierung der Einzelwerte 15](#_Toc197434425)

[c. Warnung bei niedrigem Füllstand 16](#_Toc197434426)

[d. Speicherung der Sensordaten in der Datenbank 16](#_Toc197434427)

[e. Manuelle Pumpensteuerung 16](#_Toc197434428)

[f. Debugging und Überwachung 16](#_Toc197434429)

[Zusammenfassung 16](#_Toc197434430)

[Installationsanleitung für das BloomBuddy Microcontroller-Projekt 17](#_Toc197434431)

[1. Übersicht der Komponenten 17](#_Toc197434432)

[1a. Benötigte Software für das Projekt 17](#_Toc197434433)

[1. Node-RED 17](#_Toc197434434)

[2. Thonny IDE 18](#_Toc197434435)

[3. HeidiSQL 18](#_Toc197434436)

[4. MQTT-Broker (z.B. Mosquitto) 18](#_Toc197434437)

[5. Python (inkl. MicroPython für ESP32) 18](#_Toc197434438)

[6. Treiber für den ESP32 18](#_Toc197434439)

[7. MariaDB 18](#_Toc197434440)

[2. Anschluss der Sensoren 20](#_Toc197434441)

[a) Sensoren (AHT21, BH1750, VL53L0X) 20](#_Toc197434442)

[b) Bodenfeuchtesensor (capacitive soil moisture sensor V2.0.0) 20](#_Toc197434443)

[c) Relaismodul (5V, 1-Kanal) 20](#_Toc197434444)

[d) Pumpe am Relaisausgang 20](#_Toc197434445)

[3. Hinweise zur Verkabelung 20](#_Toc197434446)

[4. Inbetriebnahme 21](#_Toc197434447)

[5. Funktionsweise 21](#_Toc197434448)

[6. Sicherheitshinweise 21](#_Toc197434449)

[7. Fehlerbehandlung 21](#_Toc197434450)

## 

## 1. Projektbeschreibung

Kurzbeschreibung:  
Mein Projekt ist eine Smarte-Pflanzenbewässerung, wobei die Temperatur, die Luftfeuchtigkeit, die Helligkeit am Standort der Pflanze, sowie die Bodenfeuchtigkeit der Erde gemessen und automatisch eine Bewässerung gesteuert wird. Zusätzlich wird in meinem Projekt der Füllstand des Wassertanks gemessen und eine entsprechende Push-Benachrichtigung ausgegeben, wenn der Füllstand des Tanks unter den 20% Wert fällt. Die Werte werden über Node-Red visualisiert. Alle Messdaten werden in einer Datenbank gespeichert, um später darauf zurückgreifen zu können.

## Link zum GitHub-Repository:

### <https://github.com/jerrit151/300_Projekt_BloomBuddy>

## 2. Anforderungen und Funktionalitäten

### ✅ Sensorik

**AHT21:** Dieser Sensor misst die Luftfeuchtigkeit und die Temperatur im Raum

**BH1750:** Dieser Sensor misst die Helligkeit am Standort der Pflanze

**VL53L0X TOF:** Dieser Sensor misst über TOF den aktuellen Füllstand des Wassertanks

**Kapazitiver Bodenfeuchtigkeitssensor:** Dieser Sensor misst die Bodenfeuchtigkeit in der Pflanzerde

### ✅ Aktoren-Steuerung

**AM325 Mini Pump:** Bewässerung der Pflanze, wenn die Pumpe angesteuert wird

### ✅ Webinterface & Benutzerinteraktion

Das Webinterface wird über Node-Red realisiert. Hier werden die aktuellen Messwerte angezeigt und für den Benutzer visualisiert. Der Benutzer kann also die aktuelle Helligkeit, die Luftfeuchtigkeit, die Temperatur, die Bodenfeuchtigkeit und den Füllstand des Wassertanks direkt ablesen. Außerdem bekommt der Benutzer eine Push-Benachrichtigung aufs Handy, wenn der Füllstand des Wassertanks unter 20% fällt.

## 3. Benötigte Komponenten

### 3.1 Hardware

| **Komponente** | **Modell/Typ** | **Funktion** |
| --- | --- | --- |
| **Mikrocontroller** | ESP32 S3 WROOM 1 | Sensoren auswerten und Pumpe ansteuern  Kommunikation mit Node-Red |
| **Sensor 1** | VL53L0X Time-of-Flight | Füllstand des Wassertanks überwachen |
| **Sensor 2** | AHT21 digitales Temperatur-Feuchtigkeitssensor-Messmodul mit I2C-Kommunikation | Messen der Temperatur und der Luftfeuchtigkeit im Raum |
| **Sensor 3** | Capacitive soil moisture sensor v2.0 kapazitiver Bodenfeuchtigkeitssensor | Messen der Bodenfeuchtigkeit in der Pflanzenerde |
| **Sensor 4** | BH1750 Lichtintensitätsbeleuchtungsmodul | Messen der Helligkeit am Standort der Pflanze |
| **Aktor 1** | AM325 Mini Pump  DC 3V-6V Peristaltische Dosierpumpe Selbstansaugend | Bewässerung der Pflanze |
| **Stromversorgung** | 5V 2A Steckernetzteil | Spannungsversorgung des ESPs, der Sensoren und der Pumpe |
| **Weitere Bauteile** | 5V 1 Kanal Relaismodul Low Level Trigger | Ansteuern der Pumpe über das Relais |
|  | Breadboard | Zum Aufbauen der Komponenten |
|  | Jumpercable in verschiedenen Längen | Zum Verbinden und anschließen der einzelnen Bauteile |
|  | PVC-Schlauch und Schlauchverbinder | Um das Wasser zur Pflanze zu führen |

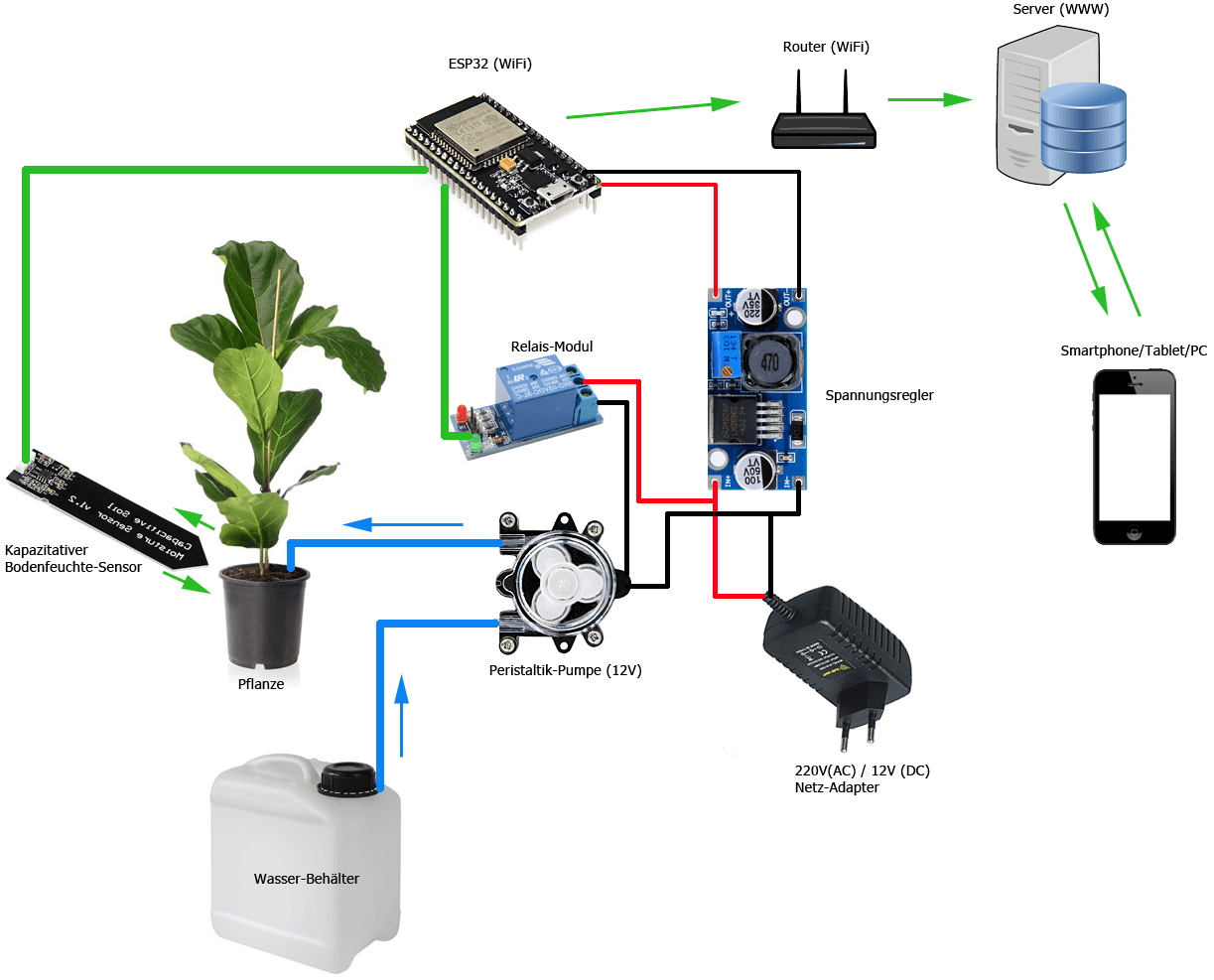
### 3.2 Software & Datenbank

| **Komponente** | **Technologie** | **Funktion** |
| --- | --- | --- |
| **Microcontroller-Code** | Micropython | Speziell für Systeme mit begrenzten Ressourcen  Benötigt nur wenig Speicherplatz  Für ARM-basierte Chips |
| **Webinterface** | Node-Red | Visualisierung der Messwerte |
| **Datenbank** | MariaDB | Speicherung der Daten |

### 3.3 Verwendete Software im Detail

* Node-RED (inkl. node-red-dashboard und node.js)
* Thonny IDE
* HeidiSQL
* Mosquitto MQTT-Broker
* Micropython für Thonny IDE
* MicroPython Firmware für den ESP32
* MariaDB
* HeidiSQL

## 4. Systemarchitektur



Die Abbildung weicht leicht von meinem Projekt ab, ist aber dennoch sehr vergleichbar. Die Änderungen, die ich hinzufügen würde, wäre das ich noch einen TOF-Sensor über I2C-Bus an den ESP32 anschließen würde, um den Füllstand des Wassertanks zu messen. Zusätzlich noch einen AHT21-Sensor zum Messen der Luftfeuchtigkeit und der Temperatur im Raum, dieser ist ebenfalls über den I2C-Bus mit dem ESP32 verbunden. Außerdem verbaue ich einen BH1750 zum Messen der Helligkeit am Standort der Pflanze, dieser ist auch an dem I2C-Bus angeschlossen.

## 5. Zeitplanung (Meilensteine)

| **Datum** | **Aufgabe** |
| --- | --- |
| KW 11 | Festlegung der Projektziele  Ermittlung der Anforderungen an das System (z.B. Sensoren für Feuchtigkeit, Temperatur)  Erstellung eines Projektplans mit Meilensteinen und Zeitrahmen |
| KW 12 | Definition der notwendigen Sensoren und Aktoren  Auswahl der geeigneten Hardwarekomponenten (z.B. Mikrocontroller, Pumpe)  Erstellung eines technischen Designs für das Bewässerungssystem  Bestellung der notwendigen Komponenten (z.B. Sensoren, Mikrocontroller, Pumpen) |
| KW 13-14 | Entwicklung der Software  Implementierung der Bewässerungslogik mit Micropython  Integration der Sensordaten zur Steuerung der Bewässerung  Durchführung von Tests zur Sicherstellung der korrekten Funktion  Implementierung der Daten in Node-Red, Visualisierung in Node-Red  Ablegen der gemessen Werte in einer Datenbank |
| KW 15 | Zusammenbau des Bewässerungssystems mit allen Komponenten  Durchführung von umfassenden Tests des gesamten Systems unter realen Bedingungen |
| KW 16 | Anpassung der Parameter zur Verbesserung der Effizienz und Zuverlässigkeit  Kalibrierung der Sensoren und Anpassung der Aktoren |
| KW 17 | Erstellung einer detaillierten Dokumentation des Systems, inklusive Schaltpläne und genauer Funktionsbeschreibung unter realen Bedingungen |
| KW 18 | Durchführung der finalen Tests und Abnahme des Systems  Installation des Systems am Einsatzort (an der Pflanze) |
| KW 19 | Reserve zum Ausgleich auftretender Probleme oder zur Verbesserung des Systems |

## 6. Offene Fragen & Herausforderungen

Batterie- und Stromprobleme:Unzureichende Batterieleistung oder Stromausfälle können das System lahmlegen, zum Beispiel bei der Versorgung über eine Powerbank

Über- oder Unterbewässerung:Falsche Einstellungen, fehlerhafte Sensoren oder unvollständige Daten können zu Über- oder Unterbewässerung führen, was der Pflanze schädigen kann

Ungleichmäßige Wasserverteilung:Ungleichmäßige Wasserverteilung kann die Pflanze unterschiedlich stark bewässern

Programmierung:Analogwertverarbeitung kann komplex sein, zudem kann die Darstellung der Daten in Node-Red eine Schwierigkeit sein. Außerdem ist die Überwachung des Füllstands vom Wassertank durch einen TOF-Sensor bisher nicht getestet wurden.

## 7.Technische Besonderheiten

Die technischen Besonderheiten an meinem Projekt sind zum einen die Push-Benachrichtigung, wenn der Füllstand des Wassertanks unter 20% (260mm) fällt. Die Push-Benachrichtigung ist in diesem Projekt eine E-Mail, die über Node-Red an den Empfänger verschickt wird. Eine weitere Besonderheit an diesem Projekt ist die Analogwertverarbeitung mit dem ESP32. Dazu wurde der Analogausgang des Capacitive Soil Moisture Sensors auf einen Analogeingang des ESP32 verdrahtet und die Auswertung des Sensors programmiert, um am Ende einen richtigen Wert zu erhalten. Der umgerechnete Wert wird dann im Programm weiterverarbeitet, um durch ein Relais dann eine Pumpe zur Bewässerung anzusteuern. Zudem kann die Pumpe jederzeit manuell über das Node-Red Dashboard ein und ausgeschaltet werden.

## 8. Zusatzfunktionen

Zusätzlich zum Automatikbetrieb des Systems wurde eine manuelle Steuerung der Pumpe implementiert. Die manuelle Steuerung erfolgt über einen Schalter auf dem Node-Red-Dashboard. Die manuelle Steuerung hat jederzeit Vorrang vor dem Automatikbetrieb. Die Pumpe kann zum Bespiel auch dann eingeschaltet werden, wenn die Bodenfeuchtigkeit 100% beträgt. Dies kann zum Beispiel dafür genutzt werden den Schlauch zu spülen oder ähnliches. Die manuelle Steuerung wird nicht automatisch ausgeschaltet, sondern geht auf OFF, wenn der Schalter auf dem Dashboard in OFF-Stellung ist. In diesem Fall geht das System sofort wieder in den Automatikbetrieb.

## 

## 9. Fazit & Zielsetzung

### Was sollte das Projekt am Ende können:

Das Projekt sollte am Ende automatisch die Bodenfeuchtigkeit in der Pflanzerde messen, diese auswerte und falls diese zu trocken ist, automatisch eine Pumpe ansteuern, welche die Pflanze dann bewässert. Außerdem werden Messwerte wie Helligkeit, Temperatur, Luftfeuchtigkeit und der Füllstand des angeschlossenen Wassertanks gemessen und über Node-Red verarbeitet. Zusätzlich sollten die Messwerte auf einem Node-Red Dashboard dargestellt werden und in einer Datenbank gespeichert werden. Eine Besonderheit ist an diesem Projekt das wenn der Wassertank unter 20% Füllstand fällt, automatisch eine Push-Benachrichtigung, in diesem Fall per E-Mail, ausgelöst wird und den Empfänger darüber informiert das der Füllstand des Wassertanks unter 260mm liegt, was ungefähr 20% des verwendeten Wassertanks entspricht. Damit er genug Zeit hat den Wassertank wieder ausreichend zu füllen. Zusätzlich wurde noch eine manuelle Steuerung der Pumpe über das Node-Red Dashboard implementiert worüber die Pumpe jederzeit manuell ein und ausgeschaltet werden kann. Die manuelle Steuerung hat zudem Vorrang vor dem Automatikbetrieb und wird nicht nach einer bestimmten Zeit abgeschaltet, sondern erst wenn der Schalter wieder auf „OFF“ steht. Alle genannten Punkte wurden entsprechend umgesetzt, um die Funktion zu gewährleisten. Auch die Kommunikation per MQTT funktioniert einwandfrei. Die Daten werden in Echtzeit auf dem Node-Red-Dashboard dargestellt und können jederzeit abgelesen werden. Die Messdaten, die in Node-Red dargestellt werden, reichen eine Woche zurück. Zusätzlich wurde programmiert das sich die Pumpe nicht mehr einschalten lässt wenn der Füllstand des Wassertanks < 280mm ist. Da der Füllstand von oben nach unten gemessen wird. Je kleiner der Messwert, desto höher ist der Füllstand.

### Persönliches Fazit:

Die Umsetzung des Projekts hat gut funktioniert. Anfangs hat die Analogwertverarbeitung nicht einwandfrei funktioniert, da die Messwerte stark geschwankt haben, auch hat die Qualität der Spannungsversorgung einen Einfluss auf die Analogwertverarbeitung. Als dann aber alle Werte richtig eingestellt und verarbeitet wurden hat diese gut funktioniert. Eine weitere Schwierigkeit, die ich hatte, war es eine Befestigung des TOF-Sensors am Wassertank zu finden, so dass er den Füllstand auch zuverlässig messen kann. Aber auch dieses Problem hat sich dann gelöst. Auch der Versand der Push-Benachrichtigung hat am Anfang nicht funktioniert, das Versenden der E-Mail ist immer wieder abgebrochen, dies lag an den falschen Einstellungen im E-Mail-Konto, konnte also ebenfalls behoben werden. Dann mussten noch die Messwerte des TOF-Sensors in Node-Red richtig verarbeitet werden, dass nur eine E-Mail versendet wird, wenn der Füllstand unter 260mm liegt. Auch diese Funktion hat nicht auf Anhieb funktioniert. Abgesehen von diesen kleineren Schwierigkeiten hat die Umsetzung wie anfangs geschrieben gut funktioniert und Spaß gemacht. Auch das man nun eine autonome funktionsfähige Pflanzenbewässerung für Zuhause hat ist gut. Zudem ist die Anlage sehr einfach auf weitere Pflanzen erweiterbar. Dafür sind nur kleinere Änderungen in der Software nötig und es werden weitere Capacitive Soil Moisture Sensoren benötigt.

# Bedienungsanleitung – Automatische Pflanzenbewässerung "BloomBuddy"

Willkommen zur Bedienungsanleitung für das Blombuddy-Bewässerungssystem! Dieses System dient zur automatisierten Versorgung Ihrer Pflanzen mit Wasser und informiert Sie zuverlässig über wichtige Umgebungsdaten.

Diese Anleitung führt Sie Schritt für Schritt durch die Funktionalitäten und den Ablauf der Anlage.

## 1. Systemübersicht

### Funktionen:

* Automatische Bewässerung der Pflanze anhand der Bodenfeuchtigkeit
* Messung und Überwachung von:
  + Bodenfeuchtigkeit
  + Temperatur und Luftfeuchtigkeit
  + Helligkeit
  + Wasserfüllstand (Entfernungssensor)
* Datenübertragung per MQTT an eine zentrale Steuerung/Visualisierung mit Node-Red
* Push-Benachrichtigungen bei niedrigem Tankfüllstand
* Speicherung aller Sensordaten in einer Datenbank
* Manuelle Steuerung der Pumpe über Node-Red-Dashboard

## 2. Inbetriebnahme und Aufbau

### Benötigte Hardware:

* ESP32 S3 Mikrocontroller
* Capacitive Soil Moisture Sensor V2.0.0 (Bodenfeuchte)
* AHT21 (Temperatur- & Luftfeuchte-Sensor)
* BH1750 (Helligkeitssensor)
* VL53L0X (Entfernungssensor, misst Wasserfüllstand)
* 5V Relais (Schaltet die Pumpe)
* 5V 2A Steckernetzteil
* AM325 Mini-Pumpe

### Benötigte Software:

* Node-RED (inkl. node-red-dashboard und node.js)
* Thonny IDE
* HeidiSQL
* Mosquitto MQTT-Broker
* Micropython für Thonny IDE
* MicroPython Firmware für den ESP32
* MariaDB
* MySQL

### Benötigte Bibliotheken in Micropython:

* aht.py
* bh1750.py
* VL53L0X.py

### Vorbereitung:

* Schließen Sie alle Sensoren und das Relais gemäß der Hardwareliste und des Schaltplans an die vorgesehenen Pins des ESP32 an.
* Stellen Sie sicher, dass der WLAN-Zugang (SSID & Passwort) korrekt im Code hinterlegt ist.

### Inbetriebnahme:

* Schalten Sie das System ein, indem Sie den ESP32 und die Pumpe mit Strom versorgen.
* Verlegen Sie die Schläuche der Pumpe so, dass ein Schlauch vom Wassertank zur Pumpe und der andere von der Pumpe zur Pflanze führt. Achten Sie dabei unbedingt auf die Flussrichtung der Pumpe, damit das Wasser korrekt gepumpt wird.
* Stecken Sie den Capacitive Soil Moisture Sensor bis zur Markierung auf dem Sensor in die Pflanzerde. Nur so kann eine zuverlässige Messung der Bodenfeuchte erfolgen.
* Platzieren Sie das Schlauchende der Pumpe mit ausreichend Abstand zum Sensor in der Erde, damit das Wasser nicht direkt auf den Sensor trifft. Dadurch werden Messfehler durch direkte Benetzung des Sensors vermieden.
* Und schon ist ihr BloomBuddy System erfolgreich installiert und nimmt seine Arbeit auf.

## 

## 3. Funktionen im Detail

### 3.1 Sensorik & Messungen

* Bodenfeuchtigkeit:  
  Der Sensor misst den Feuchtegrad im Boden. Die Werte werden zwischen "sehr trocken" (z. B. 3070) und "sehr nass" (z. B. 1700) kalibriert.
* Temperatur und Luftfeuchtigkeit:  
  Gemessen mit dem AHT21 und als Mittelwert aus mehreren Messungen ermittelt.
* Helligkeit:  
  Gemessen mit dem BH1750; Mittelwertbildung zur Genauigkeit.
* Wasserfüllstand:  
  Der VL53L0X misst die Entfernung zur Wasseroberfläche und erkennt so den Tankfüllstand.

### 3.2 Automatische Pumpensteuerung

* Ist der Boden zu trocken (≤ 40%), wird die Pumpe automatisch für 25 Sekunden eingeschaltet, um die Pflanze zu bewässern.
* Nach Ablauf der 25 Sekunden schaltet sich die Pumpe automatisch wieder ab.
* Ist die Bodenfeuchtigkeit hoch genug (> 40%), bleibt die Pumpe aus.
* Die Pumpe lässt sich nicht mehr einschalten wenn der Füllstand des Wassertank < 280 ist

### 3.3 Datenverarbeitung & -übertragung

* Jeder Messzyklus berechnet gemittelte Sensordaten und entfernt Ausreißer zur Verbesserung der Genauigkeit.
* Die Ergebnisse werden als JSON-Objekt verpackt und über MQTT an das Netzwerk (Node-Red) gesendet.
* In Node-Red werden die Daten empfangen, visualisiert und in eine Datenbank (MariaDB) gespeichert

### 3.4 Push-Benachrichtigung & Alarm

* Sinkt der Füllstand des Wassertanks unter 20% (260mm), wird eine Push-Benachrichtigung ausgelöst (eine E-Mail versendet).

## 4. Bedienung im Alltag

### Automatikbetrieb:

* Nach Anschluss startet das System automatisch.
* Die Sensoren messen regelmäßig alle Werte, regeln die Bewässerung und melden die Umgebungsinformationen an das Netzwerk.
* Sie müssen lediglich sicherstellen, dass der Wassertank regelmäßig nachgefüllt wird.

### Manuelle Steuerung über das Node-Red Dashboard

Die Pumpe kann jederzeit manuell über das Node-Red Dashboard gesteuert werden. Die manuelle Steuerung hat Vorrang vor dem Automatikbetrieb:

* Ist der Schalter im Dashboard auf „ON“ gestellt, läuft die Pumpe dauerhaft und der Automatikbetrieb ist deaktiviert. Dies ist beispielsweise nützlich, um den Schlauch zu spülen oder die Pumpe gezielt zu testen.
* Ist der Schalter auf „OFF“ gestellt, übernimmt das System wieder die automatische Steuerung der Bewässerung entsprechend der gemessenen Bodenfeuchtigkeit.
* Solange der Schalter auf „ON“ bleibt, bleibt die Anlage im Handbetrieb und geht nicht in den Automatikmodus über.

Die Steuerung erfolgt über einen Schalter im Node-Red Dashboard, der per MQTT mit dem System verbunden ist.

### Statuskontrolle:

* Die aktuelle Bodenfeuchte, Temperatur, Helligkeit und den Tankfüllstand können Sie über die angeschlossene Benutzeroberfläche in Node-Red jederzeit ablesen.
* Bei Problemen mit dem WLAN oder MQTT informiert das System im Konsolenausdruck.

## 5. Wartung & Tipps

* Sensor-Kalibrierung: Die Referenzwerte für „trocken“ und „nass“ sollten einmalig für Ihre Pflanze ermittelt und im Code angepasst werden.
* Wassertank: Halten Sie den Tank stets gefüllt, damit das System zuverlässig arbeiten kann.

## 6. Störungen & Fehlersuche

* Meldet das System „WLAN-Verbindung fehlgeschlagen!“ prüfen Sie Zugangsdaten und Router.
* Reagiert die Pumpe nicht:
  + Prüfen Sie das Relais oder die Verkabelung
* Falsche Sensordaten:
  + Sensoren auf korrekten Sitz und Verschmutzung prüfen.

## 7. Beschreibung des Node-Red-Flows

Übersicht der Funktionen im Node-RED-Flow für das BloomBuddy-Projekt

### Node-Red-Flow:

Ein Bild, das Text, Screenshot, Software, Zahl enthält.

KI-generierte Inhalte können fehlerhaft sein.

### Node-Red-Dashboard:

Ein Bild, das Text, Screenshot, Multimedia-Software, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

### a. MQTT-Empfang der Sensordaten

* **Node:** mqtt in („Sensorwerte“)
* Funktion: Empfängt die vom ESP32 gesendeten Sensordaten (Füllstand, Temperatur, Luftfeuchtigkeit, Helligkeit, Bodenfeuchtigkeit) über das Topic Zuhause/Wohnung/BloomBuddy.

### b. Aufbereitung und Visualisierung der Einzelwerte

* **Function-Nodes:** Extrahieren jeweils einen Wert (z.B. Füllstand, Temperatur, Luftfeuchtigkeit, Helligkeit, Bodenfeuchtigkeit) aus der empfangenen Nachricht.
* **UI-Gauges & Charts:** Jeder Wert wird auf dem Dashboard als Messinstrument (Gauge) und als Verlauf (Chart) angezeigt, z.B.:
  + Füllstand\_Tank (Gauge, Chart)
  + Temperatur (Gauge, Chart)
  + Luftfeuchtigkeit (Gauge, Chart)
  + Helligkeit (Gauge, Chart)
  + Bodenfeuchtigkeit (Gauge, Chart)

### c. Warnung bei niedrigem Füllstand

* **Switch-Node:** Prüft, ob der Füllstand unter 260 mm (ca. 20 %) liegt.
* **Function-Node:** Erstellt eine Warnmeldung, wenn der Wert unterschritten wird.
* **E-Mail-Node:** Versendet diese Warnung als E-Mail an eine festgelegte Adresse.
* **Delay-Node:** Begrenzung der E-Mail-Frequenz, damit nicht zu viele Warnungen gesendet werden.

### d. Speicherung der Sensordaten in der Datenbank

* **Function-Node:** Bereitet die Sensordaten und einen Zeitstempel für den SQL-Insert vor.
* **MySQL-Node:** Schreibt die Daten in die Tabelle sensorwerte\_wohnung der Datenbank.

### e. Manuelle Pumpensteuerung

* **UI-Switch:** Ermöglicht das manuelle Ein- und Ausschalten der Pumpe über das Dashboard.
* **MQTT-Out-Node:** Sendet den Schaltbefehl als MQTT-Nachricht an das ESP32-Programm, das daraufhin die Pumpe steuert.

### f. Debugging und Überwachung

* **Debug-Nodes:** Geben Nachrichten und Werte zur Kontrolle in der Node-RED-Seitenleiste aus.

## Zusammenfassung

Das BloomBuddy-System arbeitet weitgehend autark mit automatischer Steuerung und Übertragung aller relevanten Daten. Eingriffe Ihrerseits beschränken sich auf das Nachfüllen des Tanks und die Kontrolle der Sensorfunktion über Node-Red – den Rest erledigt die Anlage selbstständig. Zusätzlich können Sie die Pumpe jederzeit manuell über das Node-Red Dashboard steuern. Die manuelle Steuerung hat dabei Vorrang vor dem Automatikbetrieb: Ist der Schalter im Dashboard auf „ON“ gestellt, läuft die Pumpe dauerhaft und der Automatikmodus ist deaktiviert – dies kann beispielsweise zum Spülen des Schlauchs genutzt werden. Erst wenn der Schalter wieder auf „OFF“ steht, übernimmt das System wieder automatisch die Bewässerung entsprechend der gemessenen Bodenfeuchtigkeit.

# Installationsanleitung für das BloomBuddy Microcontroller-Projekt

Diese Anleitung erklärt Schritt für Schritt, wie man die Hardware korrekt anschließt und das System in Betrieb nimmst.

## 1. Übersicht der Komponenten

* ESP32 S3
* Capacitive Soil Moisture Sensor V2.0.0 (analog)
* AHT21 (Temperatur & Luftfeuchtigkeit, I2C)
* BH1750 (Helligkeit, I2C)
* VL53L0X (ToF Entfernungssensor, I2C)
* 5V Relaismodul
* AM325 Mini Pumpe
* 5V 2A Steckernetzteil
* Diverse Jumper-Kabel

### 1a. Benötigte Software für das Projekt

* Node-RED (inkl. node-red-dashboard und node.js)
* Thonny IDE
* HeidiSQL
* Mosquitto MQTT-Broker
* Micropython für Thonny IDE
* MicroPython Firmware für den ESP32
* MariaDB
* MySQL

### 1. Node-RED

* Zur Visualisierung, Steuerung und Automatisierung der Sensordaten und Pumpensteuerung.
* Node-RED wird als Flow-basierte Entwicklungsumgebung genutzt, um MQTT-Nachrichten zu empfangen, Daten zu verarbeiten, in eine Datenbank zu schreiben und Benachrichtigungen zu generieren.

### 2. Thonny IDE

* Zum Schreiben, Hochladen und Debuggen von MicroPython-Code auf dem ESP32.
* Die Thonny IDE ist besonders geeignet für Einsteiger und bietet eine einfache Möglichkeit, MicroPython-Code direkt auf das Board zu übertragen und auszuführen.

### 3. HeidiSQL

* Zur Verwaltung und Überwachung deiner MariaDB- oder MySQL-Datenbank, in die Node-RED die Sensordaten schreibt.
* HeidiSQL bietet eine grafische Oberfläche, um Tabellen zu erstellen, Abfragen auszuführen und die Datenbankstruktur zu verwalten.

### 4. MQTT-Broker (z.B. Mosquitto)

* Der MQTT-Broker ist das Bindeglied zwischen deinem ESP32 (der die Sensordaten publiziert) und Node-RED (das die Daten abonniert und weiterverarbeitet).
* Mosquitto ist ein weit verbreiteter, kostenloser MQTT-Broker, der sich einfach unter Windows installieren lässt.

### 5. Python (inkl. MicroPython für ESP32)

* Python wird auf deinem Laptop für die Thonny IDE benötigt.
* MicroPython muss auf dem ESP32 installiert werden, damit der bereitgestellte Code ausgeführt werden kann.

### 6. Treiber für den ESP32

* Damit dein Windows-Laptop den ESP32 erkennt, benötigst du die passenden USB-Treiber (meist CP210x oder CH340, je nach Board).

### 7. MariaDB

* Die eigentliche Datenbank, in die Node-RED die Sensordaten schreibt.

Anbei der Aufbau der Datenbank und einige Beispielmesswerte.

Ein Bild, das Text, Screenshot, Software, Display enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Display, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

## 2. Anschluss der Sensoren

### a) Sensoren (AHT21, BH1750, VL53L0X)

* VCC aller Sensoren an +5V (ESP32 5V-Pin (auf Breadboard))
* GND aller Sensoren an GND (ESP32 GND-Pin (auf Breadboard))
* SDA aller Sensoren an GPIO 5
* SCL aller Sensoren an GPIO 4
* Als Busabschluss wurden für SCL & SDA jeweils 4,7kΩ Widerstände auf+5V verdrahtet

### b) Bodenfeuchtesensor (capacitive soil moisture sensor V2.0.0)

* VCC an +5V
* GND an GND
* Signal an GPIO 15

### c) Relaismodul (5V, 1-Kanal)

* DC+ (VCC) an +5V (ESP32 5V-Pin)
* DC- (GND) an GND (ESP32 GND-Pin)
* IN1 an GPIO 8 (ESP32, wie im Code: Pin(8))
* Relais schaltet bei High-Signal, siehe Jumper

### d) Pumpe am Relaisausgang

* COM am Relais an +5V des externen Netzteils
* NO (Normally Open) am Relais an Plusleitung der Pumpe
* NC bleibt frei
* Minusleitung der Pumpe an GND des externen Netzteils

## 3. Hinweise zur Verkabelung

* **Alle GND-Leitungen** (ESP32, Sensoren, Relais, Pumpe) müssen verbunden sein.
* **Achtung:** Arbeite immer spannungsfrei, solange du verkabelst!
* **5V-Relais:** Der Steuerstrom für das Relais kommt vom ESP32. Die Last (Pumpe) wird über die Relaiskontakte geschaltet.

## 4. Inbetriebnahme

1. **Verkabelung kontrollieren** (siehe oben).
2. **Code auf den ESP32 flashen** (mit Thonny) .
3. **ESP32 mit Strom versorgen** (über USB Typ C per Powerbank oder ähnliches).
4. **Pumpe mit Strom versorgen** (über 5V 2A Steckernetzeil)
5. **MQTT-Server bereitstellen** (wie im Code, z.B. Mosquitto auf 192.168.33.79).
6. **Node-Red verwenden**, um Nachrichten zu senden:
   * {"Schalter1":"ON"} schaltet die Pumpe ein (Relais HIGH)
   * {"Schalter1":"OFF"} schaltet die Pumpe aus (Relais LOW)

Haben sie alle diese Punkte abgearbeitet sollte die Anlage sofort mit ihrer Arbeit beginnen. Die Steuerung über Node-Red ist optional. Die Anlage funktioniert auch ohne eine manuelle Steuerung der Pumpe.

## 5. Funktionsweise

* **Automatikbetrieb:** Die Pumpe schaltet automatisch ein, wenn die Bodenfeuchtigkeit ≤ 40 % beträgt, der Wassertank ausreichend gefüllt ist < 280mm und kein Handbetrieb aktiv ist. Sie läuft dann 25 Sekunden. Danach erfolgt eine erneute Messung
* **Manueller Betrieb:** Über MQTT kann die Pumpe jederzeit ein- und ausgeschaltet werden. Der Handbetrieb hat Vorrang.
* **Sensorwerte** werden regelmäßig gemessen und per MQTT publiziert.

## 6. Sicherheitshinweise

* **Kein Netzstrom:** Das System ist für 5V-Komponenten ausgelegt.
* **Relaismodul:** Achte darauf, dass der Steuerstrom des Relais den ESP32 nicht überlastet.
* **Wasser & Elektronik:** Sorge für Spritzwasserschutz und sichere Montage.

## 7. Fehlerbehandlung

* **Pumpe schaltet nicht:** Prüfe, ob das Relais beim Schalten klickt. Miss die Spannung am Relaisausgang.
* **Sensorwerte fehlen:** Prüfe I2C-Verbindungen (SDA/SCL) und Versorgungsspannung.
* **MQTT funktioniert nicht:** Prüfe WLAN, Broker-IP und MQTT-Topic.
* **ESP32 startet nicht:** Prüfe, ob die 5V-Versorgung stabil ist.