

Technische Spezifikation

Automatische Balkonbewässerung

Autor: Mohammad Abuosba
Letzte Änderung: 13. Juni 2025
Dateiname: Automatische_Balkonbewässerung_technische_Spezifikation
Version: 1

Inhaltsverzeichnis

1	Vorhandene Dokumente	5
2	Prozessüberblick.....	6
2.1	Fachlicher Workflow	6
2.2	Technischer Workflow	7
3	Technische Spezifikation SW	8
3.1	Überblick Komponenten.....	8
3.2	Beschreibung der Implementierung	9
3.2.1	Funktion 1: Zeitplan einstellen	9
3.2.2	Funktion 2: Verlauf anzeigen	12
3.3	System Infrastruktur	14
4	Technische Spezifikation Konstruktion	15
4.1	Baugruppen.....	15
5	Modul Abhängigkeiten	16

Abbildungsverzeichnis

Abbildung 1: Fachlicher Workflow.....	6
Abbildung 2: Komponentendiagramm.....	8
Abbildung 3: Klassendiagramm Zeitplan	11
Abbildung 4: Klassendiagramm Verlauf.....	13
Abbildung 5: Systeminfrastruktur	14
Abbildung 6: Pinout des μ Cs.....	15

Copyright

© Mohammad Abuosba

Die Weitergabe, Vervielfältigung oder anderweitige Nutzung dieses Dokumentes oder Teile davon ist unabhängig vom Zweck oder in welcher Form untersagt, es sei denn, die Rechteinhaber/In hat ihre ausdrückliche schriftliche Genehmigung erteilt.

1 Vorhandene Dokumente

Alle für die vorliegende Spezifikation ergänzenden Unterlagen müssen hier aufgeführt werden

Dokument	Autor	Datum
Last_Lastentheft.pdf	Dzaid, Johannes, Zul	25.04.2025
Automatische_Balkonbewässerung_Pflichtenheft.pdf	Dzaid, Johannes, Zul	23.05.2025

2 Prozessüberblick

2.1 Fachlicher Workflow

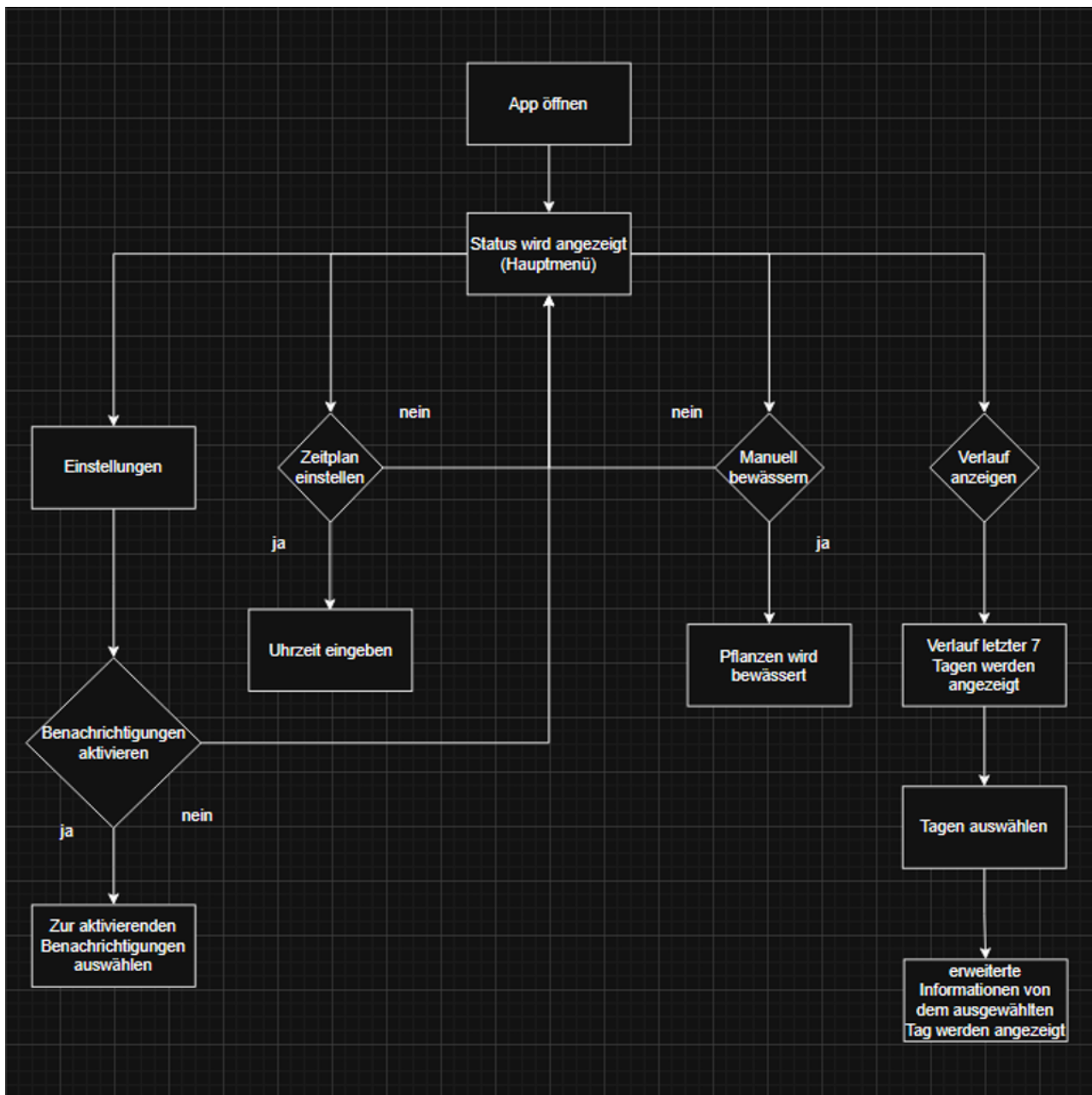


Abbildung 1: Fachlicher Workflow

Ablaufschema z.B.:

- Nutzer öffnet App → Dashboard → Navigation zu Verlauf/Zeitplan
- Manuelle/Automatische Bewässerung starten
- Ereignisse werden pro Wochentag gespeichert

2.2 Technischer Workflow

2.2.1 Sensor misst Spannung (analog) → ESP32

- Der Bodenfeuchtesensor liefert ein analoges Spannungssignal.
- Der ESP32 liest über einen ADC-Pin regelmäßig den Wert (z. B. alle 30 Sekunden).

2.2.2 ESP32 berechnet Feuchtigkeitswert

- Der ADC-Wert wird kalibriert und in einen Prozentwert (0–100 %) umgerechnet.

2.2.3 MQTT Publish

- Der ESP32 veröffentlicht den Feuchtigkeitswert an den Broker auf dem Topic:

2.2.4 App abonniert auf MQTT-Topic

- Die App ist mit demselben Broker verbunden und erhält die Sensordaten.
- Die UI zeigt den aktuellen Zustand

2.2.5 Automatische Entscheidung auf ESP

- Der ESP prüft lokal: Wenn value < threshold dann: GPIO High (Pumpe aktivieren)
- Alternativ: Prüfung gegen Zeitplan

2.2.6 Manuelle Steuerung über App

- Die App sendet MQTT-Befehl an
- "Jetzt Bewässern"

2.2.7 ESP abonniert auf Steuer-Topic

- Wenn Nachricht empfangen, GPIO High (Pumpe aktivieren wie eingestellt)

2.2.8 MQTT Logging

- Jede Aktion (Start, Stoppt, Fehler) wird an einem bestimmten Topic publiziert.

2.2.9 Fehlerhandling

- Wenn Sensor nicht lesbar (invalid), Fehlermeldung über MQTT publizieren
- App zeigt Fehlermeldung an

3 Technische Spezifikation SW

3.1 Überblick Komponenten

IT-Komponente	Funktion aus Pflichtenheft
Navigation	Drawer mit Seiten: Hauptmenü, Verlauf, Zeitplan
Verlauf	Anzeige pro Tag mit Icons und Dialog
Zeitplan	Zeitauswahl mit Schalter für automatische Bewässerung
Manuelle Steuerung	„Jetzt bewässern“-Button
UI-Komponenten	CustomScaffold, Popup-Dialoge

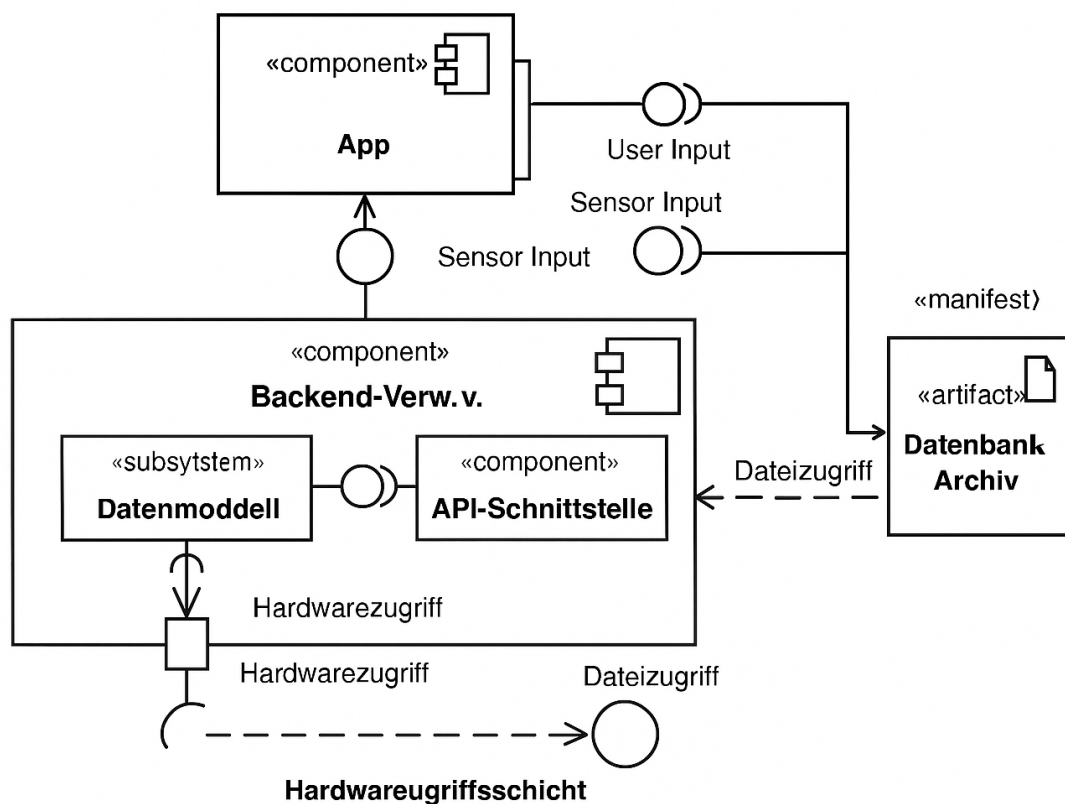


Abbildung 2: Komponentendiagramm

3.2 Beschreibung der Implementierung

3.2.1 Funktion 1: Zeitplan einstellen

#	Komponentendetail	Erforderliche Arbeiten
T1	GUI	Flutter ZeitplanScreen, Switch & TimePicker
T2	Logik	State Management (setState), Zeit speichern
T3	Unterfunktion	_selectTime() – öffnet Zeitwahl

T1: GUI

Was wird implementiert?

- Startbildschirm mit aktuellem Sensorstatus
- Navigation durch mehrere Sensoren via PageView
- „Jetzt bewässern“-Button

Elemente:

- Überschrift (Text)
- Menü-Button ☰ für Drawer
- Sensoranzeige in Slideshow
- Indikatoren (● ○ ○)
- Bewässerungsbutton (ElevatedButton)

Verwendete Widgets/Bibliotheken:

- PageView.builder
- ElevatedButton
- Text, Row, Column, Spacer
- Icons, Material3, AnimatedPositioned

Interaktionen:

- Swipe → Sensor wechseln
- Klick auf Menü → Drawer mit Seiten

Fehlerbehandlung:

- Sensorwerte = null → Anzeige: „Daten nicht verfügbar“
- Kein Sensor → leerer Zustand mit Hinweis

T2: Logik

Ablauflogik:

- Beim Öffnen der App werden die Sensordaten geladen (lokal oder via API).
- Bei Wechsel der PageView wird der aktuelle Index gespeichert.
- Daten werden als List<Map<String, String>> verwaltet.

Pseudocode:

```
PageView.builder(  
  itemCount: sensors.length,  
  onPageChanged: (i) => _currentIndex = i;  
)
```

State Management:

- StatefulWidget mit setState

Fehlerfälle:

- Ungültiger Wert → Farbliche Markierung (z. B. rot < 20 % Feuchtigkeit)

T3: Unterfunktion: “Jetzt bewässern”

GUI:

- Grüner Button, abgerundet
- Auf voller Breite, mittig

Funktionalität:

- Sendet manuellen Befehl zur Bewässerung (z. B. POST an API)
- Feedback via Snackbar oder Animation

Pseudocode:

```
onPressed: () {  
  sendWateringCommand();  
  showSnackbar("Wasser gesendet");  
}
```

Fehlerfälle:

Keine Verbindung → Snackbar mit „Fehler beim Senden“

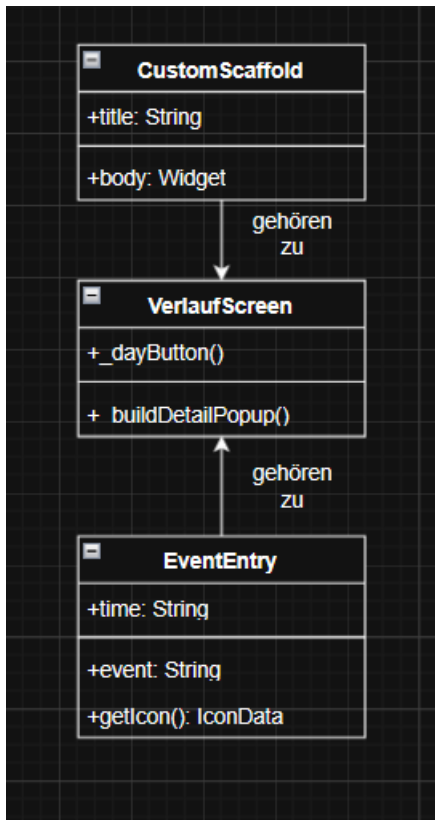


Abbildung 3: Klassendiagramm Zeitplan

3.2.2 Funktion 2: Verlauf anzeigen

#	Komponentendetail	Erforderliche Arbeiten
T4	GUI	VerlaufScreen mit Buttons je Wochentag
T5	Dialoge	Tagesbezogene Detaildialoge mit Icons
T6	Animation	Übergangsanimation zwischen Tagen
T7	Datenmodell	Map<String, List<Map<String, String>>> für Events

T4: GUI & Dialoge

- Verlauf-Bildschirm mit 7 Wochentags-Buttons
- Jeder Button öffnet Dialog mit Eventverlauf
- Icons je nach Ereignistyp

Elemente im Dialog:

- Titel: „Samstag – 29. April“
- Liste von Events mit Uhrzeit, Icon & Beschreibung
- Schließen-Button (oben links)

Widgets:

- Wrap, GestureDetector, Dialog
- ListView, Icon, Text, Row

T5: Animation

- showDialog nutzt Fade-In
- Neu: animierter Übergang zwischen Tagen (AnimatedSwitcher oder FadeTransition)

Fehlerbehandlung:

Kein Eintrag → „Keine Einträge für diesen Tag“

Formatfehler → Standard-Icon und grauer Text

T6: Datenmodell

Modell für Events:

```
{  
  "time": "11:00",  
  "event": "Automatisch bewässert (100 ml)"  
}
```

Tabelle:

Feld	Typ	Beschreibung
time	String	Uhrzeit im HH.mm-Format
event	String	Bschreibung mit Kontext
type	Enum	(intern): moisture, water, error...

Zuordnung Icon/Typ:

- Feuchtigkeit → 💧 Icons.opacity
- Wasserstand → 🌊 Icons.water_drop
- Fehler → ⚠️ Icons.error_outline

Logik zur Erkennung:

```
if (event.contains("bewässert")) => Icons.water  
else if (event.contains("Fehler")) => Icons.error_outline
```

Animation (zwischen Tagen im Dialog):

```
AnimatedSwitcher(  
  duration: Duration(milliseconds: 300),  
  child: EventListWidget(day),  
)
```

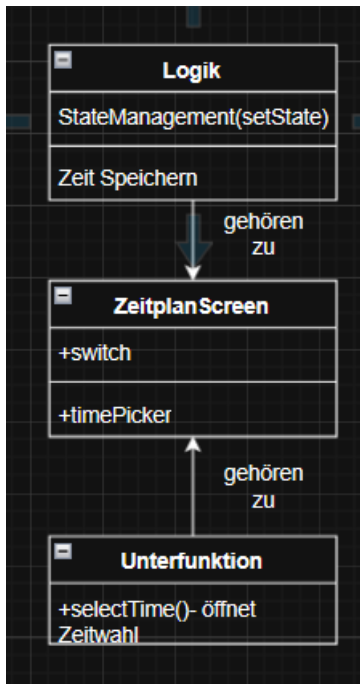


Abbildung 4: Klassendiagramm Verlauf

3.3 System Infrastruktur

- Plattform: Flutter (iOS, Android)
- Lokale Tests mit: iPhone Simulator, Android Emulator
- API-Kommunikation: (Wenn vorhanden, z. B. REST oder MQTT)
- App benötigt Netzwerkzugang
- Code verwaltet via GitLab Repository

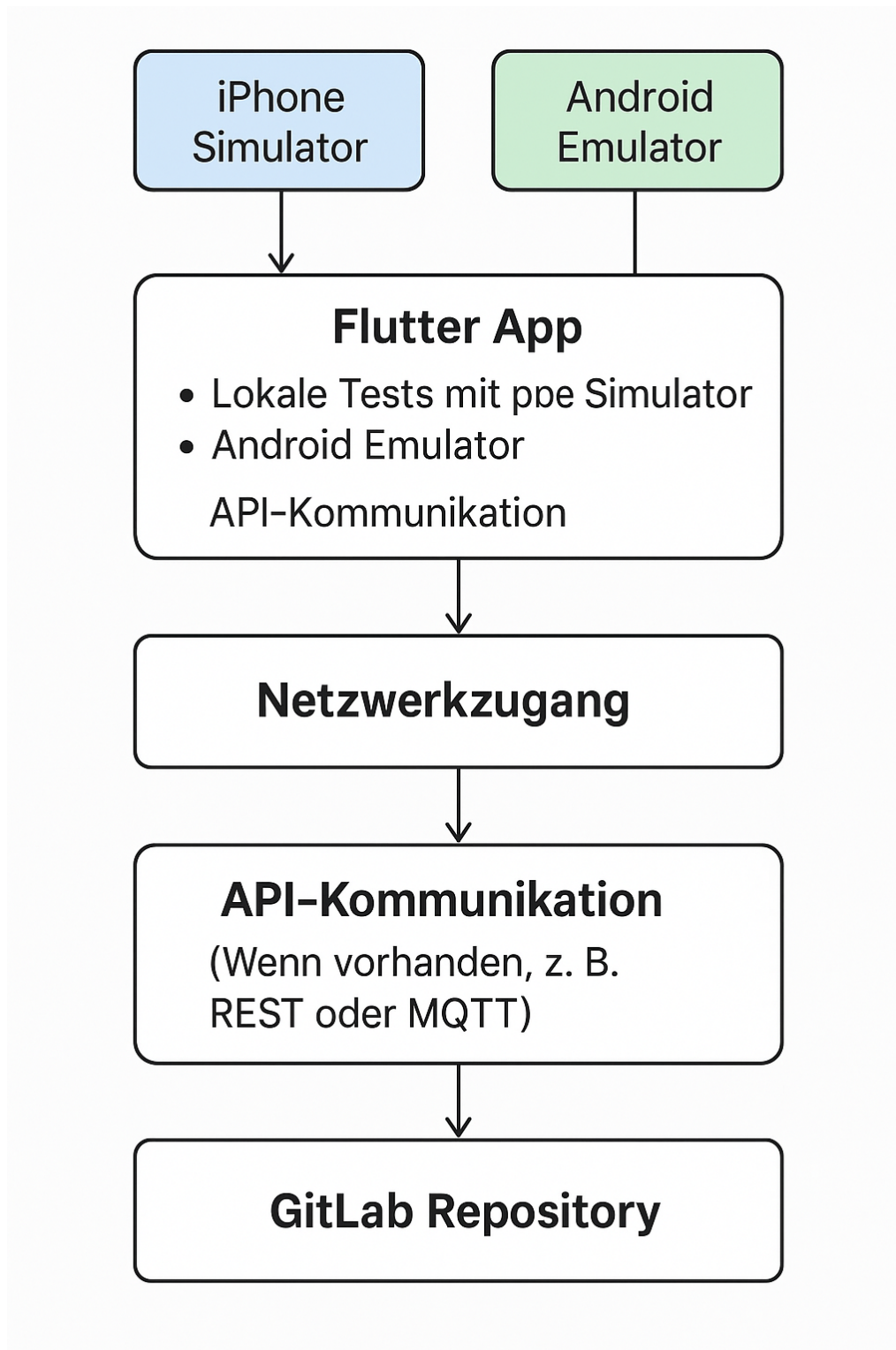


Abbildung 5: Systeminfrastruktur

4 Technische Spezifikation Konstruktion

4.1 Baugruppen

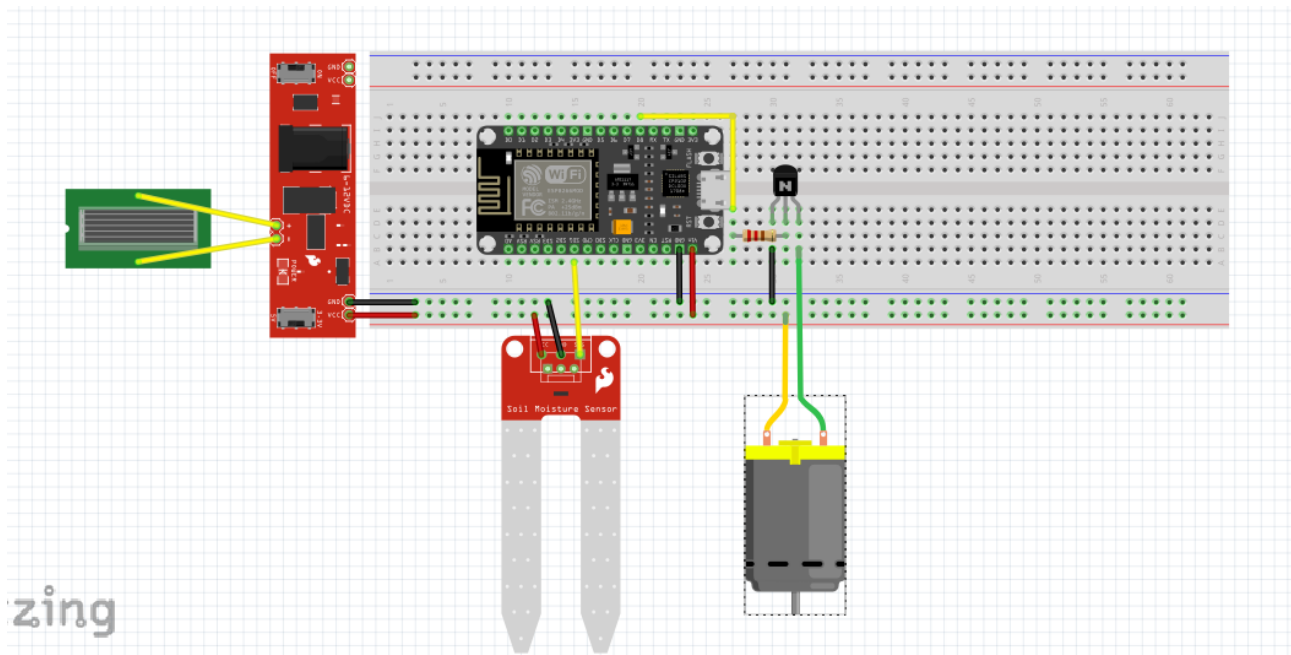


Abbildung 6: Pinout des μ Cs

Die Baugruppen sollen in diesem Abschnitt detailliert beschrieben werden.

- Eine Baugruppe erhält einen eigenen Namen
 - Bauteile, die in mehreren Ästen der Struktur vorkommen, werden mehrfach genannt
- Die Baugruppe beinhaltet
 - Einzelteile
 - Unterbaugruppen
- Bei der Strukturstückliste ist der gesamte Aufbau des Produkts erkenntlich
Bei der Baukastenstückliste werden Unterbaugruppen nicht weiter aufgegliedert, die Struktur ist nur einstufig

5 Modul Abhängigkeiten

- Flutter App ↔ Sensor-Hardware (ESP32)
- Flutter App ↔ Backend/Server (wenn vorhanden)
- Flutter App ↔ Plattformen (iOS, Android SDK)