# Projektarbeit Getränkemischanlage

mögliche Gliederung:

* Einleitung & Motivation
* Systemübersicht
* Hardwarebeschreibung
* Softwarearchitektur
  + Module, Klassenübersicht
* Kommunikation Raspberry Pi Pico
* GUI-Bedienung
* Versuch & Tests
* Fazit & Ausblick
* Anhang
  + Codeauszüge
  + Schaltpläne
  + GitHub-Link

## Einleitung und Motivation

**Ablauf (aus Sicht der GUI)**

1. Nutzer wählt ein Mischgetränk aus
2. GUI sendet Befehl an Pico
3. Pico aktiviert Pumpen nach Rezept
4. GUI zeigt aktuellen Füllstand & Temperatur an
5. Fehler oder Füllwarnung werden angezeigt

## Hardwarebeschreibung

### Wasserpumpen

* <https://www.conrad.de/de/p/offgridtec-tauchpumpe-12v-dc-10l-0-5-bar-max-5m-pumphoehe-884909107.html>
* Offgridtec Tauchpumpe 12V DC 10l 0,5 bar max. 5m Pumphöhe
* Technische Daten:
  + 10l/min Fördermenge
  + 12V Betriebsspannung
  + 3V Anlaufspannung
  + 0,5 bar Pumpendruck
  + 14,5W Nennleistung
  + 1,2 A DC Strom
  + 1,5m Kabellänge
  + Gewicht 0,2kg
  + Schutzart IP 68
  + Produktart Tauchpumpe
  + 6,53 € / Stück
  + nur für Geschäftskunden erhältlich
* Schutzart IP 68 Vorraussetzung, da die Pumpe im Flüssigkeitsbehälter verbaut ist
* Gewicht 0,2kg zum Vorteil damit Anlage leicht transportiert werden kann und Behälter schnell gewechselt werden können
* 1,5m Kabellänge notwendig aber ausreichend da Pumpen ind en Behältern verteilt sind und Kabel bis zur Steuerung reichen muss
* Eine relativ geringe Leistung von 14,5W ist vorteilhaft, da Steckboard des Raspberry und restliche Elektronik (z.B Transistor) nicht auf große Leistung ausgelegt werden muss, was Kosten spart und sicherer ist
* Eine Fördermenge von 10l/min ist völlig ausreichend. Bsp.-Rechnung: Gemischtes Getränk soll einen Inhalt von 400ml haben. Dafür bräuchte eine Pumpe 2,4s. Wenn dazu noch mehrere Pumpen gleichzeitig laufen (falls so viel Leistung zur Verfügung steht) ist das schnell genug.
* Es bietet sich die Ausführung als Tauchpumpe an, da dabei die Pumpe dauerhaft unterhalb des Wasserpegels steht und dadurch immer mit Wasser gefüllt ist. Dadurch läuft die Pumpe nie „leer“ (saugt Luft an) was dazu führen könnte, dass keine Flüssigkeit angesaugt wird da diese durch die Schwerkraft wieder zurück in den Behälter fliesen würde. Dadurch wird ein Betrieb bei zu hoher Leistung vermieden, der zu Überlast führen könnte und die Elektronik beschädigen könnte. Ein Sensor soll den Pegel des Flüssigkeitbehälters überwachen und somit sicherstellen, dass die Pumpe dauerhaft im Kontakt mit Wasser ist.

### Anschluss Pumpen

* Der Raspberry Pi selbst kann keine 12V-Pumpen direkt betreiben, da:
  + seine GPIOs nur 3,3V Signale liefern,
  + der Strom pro GPIO auf max. ~16 mA begrenzt ist,
  + und die Pi-eigene Stromversorgung nur 5V mit begrenzter Leistung bereitstellt.
* um die Wasserpumpen ansteuern zu können wird entweder ein Relais-Modul oder ein Transistor verwendet werden

|  |  |
| --- | --- |
| Komponente | Funktion |
| Externe 12V-Stromquelle | Versorgt die Pumpe mit ausreichend Leistung |
| MOSFET-Modul oder Relais-Modul | Ermöglicht das Schalten der Pumpe mit GPIO |
| Freilaufdiode (z. B. 1N4007) | Schützt vor Spannungsspitzen beim Ausschalten (besonders bei Relais) |

|  |  |  |
| --- | --- | --- |
| Variante | Vorteile | Nachteile |
| 1. N-Kanal MOSFET-Schaltung | - Sehr effizient (geringer Spannungsabfall)  - Schnell schaltbar (PWM möglich)- Kein mechanischer Verschleiß  - Leise im Betrieb  - Kompakt, günstig | - Erfordert korrektes Verständnis von Gate-Spannung (Logic-Level nötig)  - MOSFET muss geeignet gewählt sein (z. B. IRLZ44N statt IRF540N) |
| 2. Relaismodul (mechanisch) | - Einfach anzuschließen (auch für Anfänger)  - Galvanische Trennung (je nach Modul)  - Schaltet auch größere Lasten sicher | - Mechanischer Verschleiß (Begrenzte Lebensdauer)  - Klick-Geräusch beim Schalten- Langsamer als MOSFET (nicht für PWM geeignet)  - Höherer Stromverbrauch beim Anziehen |

wir wählen Variante 1: N-Kanal-MOSFET (logiklevel-tauglich).

* Vorteile:
  + Schneller, leiser, effizienter als Relais, Strom für Relais evtl zu gering. günstiger
  + Gut für PWM-Steuerung (z. B. Geschwindigkeit)
* IRLZ44N
* Infineon Technologies IRLZ44NPBF MOSFET 1 HEXFET 110 W TO-220

|  |  |
| --- | --- |
| Kriterium | Eigenschaft |
| Logic-Level-MOSFET | Ja – schaltet bereits ab ca. 2 V Gate-Spannung sicher durch (ideal für 3,3 V GPIO) |
| Max. Drain-Strom (I<sub>D</sub>) | 47 A – weit über dem benötigten Strom der Pumpe (1,2 A), bietet hohe Reserve |
| Durchlasswiderstand R<sub>DS(on)</sub> | nur 0,022 Ω – sorgt für kaum Wärmeentwicklung und geringen Spannungsverlust |
| Spannungsfestigkeit | 55 V Drain-Source-Spannung – ausreichend für 12 V Systeme |
| Gehäuse | TO-220 – leicht zu montieren |
| Verfügbarkeit | Direkt bei Conrad erhältlich, zuverlässig und dokumentiert |
| Schaltgeschwindigkeit | Schnell – geeignet für PWM-Ansteuerung (z. B. Drehzahlregelung, Dosierung) |
| Zuverlässigkeit | Industriell erprobt, robust und weit verbreitet |

Schaltung

1. **MOSFET-Modul anschließen:**
   * **Gate (Steuereingang):** Verbinde den Gate-Pin des MOSFET-Moduls über einen 220 Ω Widerstand mit einem GPIO-Pin des Raspberry Pi.
   * **Drain (Ausgang):** Verbinde den Drain-Pin mit dem Minuspol der Pumpe.
   * **Source (Eingang):** Verbinde den Source-Pin mit dem gemeinsamen Massepunkt (GND).
2. **Pumpe anschließen:**
   * **Pluspol:** Verbinde den Pluspol der Pumpe mit dem Pluspol der 12V-Stromversorgung.
   * **Minuspol:** Verbinde den Minuspol der Pumpe mit dem Drain-Pin des MOSFET-Moduls.
3. **Freilaufdiode installieren:**
   * Schalte die Freilaufdiode parallel zur Pumpe, wobei die Kathode (Markierung) mit dem Pluspol der Pumpe verbunden wird.
4. **Gemeinsame Masse:**
   * Verbinde den Masseanschluss der 12V-Stromversorgung mit dem GND des Raspberry Pi, um einen gemeinsamen Massepunkt sicherzustellen

#### Testen der Schaltung

1. **GPIO-Pin konfigurieren:**
   * Setze den verwendeten GPIO-Pin des Raspberry Pi als Ausgang.
2. **Pumpe ein- und ausschalten:**
   * Setze den GPIO-Pin auf HIGH, um die Pumpe einzuschalten.
   * Setze den GPIO-Pin auf LOW, um die Pumpe auszuschalten.
3. **PWM-Steuerung (optional):**
   * Für eine Geschwindigkeitsregelung der Pumpe kannst du PWM-Signale verwenden. Stelle sicher, dass das MOSFET-Modul PWM-kompatibel ist.

### Raspberry Pi Pico

Als Steuereinheit wird der Raspberry Pi Pico gewählt, da er ein kostengünstiges Mikrocontroller-Board ist das sich besonders für einfache bis fortgeschrittene Elektronik- und Steuerungsprojekte eignet. Er basiert auf dem selbst entwickelten RP2040-Chip und ist für seine Flexibilität, Energieeffizienz und Benutzerfreundlichkeit bekannt.

Vorteile des Raspberry Pi Pico

* Günstiger Preis
* Eigener RP2040-Mikrocontroller ausreichend für viele Embedded-Anwendungen.
* Vielfältige Schnittstellen
* USB-Programmierung (Drag & Drop)
* Niedriger Energieverbrauch
* Große Community & gute Dokumentation
* Kompakte Bauform

### Sensoren

#### Temperatursensoren

Um einen **Temperatursensor an den Raspberry Pi Pico** anzuschließen, hängt die genaue Vorgehensweise vom verwendeten Sensor ab. Zwei sehr gängige Sensoren sind:

1. **Analog**
2. **Digital**

Da wir mehrere Temperatursensoren anschließen wollen wählen wir einen digitalen Sensor.

**DS18B20 (1-Wire)**

Dieser Sensor braucht eine spezielle Bibliothek, funktioniert aber sehr genau.

**Benötigt:**

* DS18B20
* 4.7kΩ Widerstand (zwischen DATA und VCC)
* 3 Jumperkabel

**Anschluss:**

| **DS18B20 Pin** | **Verbinden mit** |
| --- | --- |
| GND | GND |
| VDD | 3.3V |
| DATA | z. B. GPIO15 (Pin 21) + 4.7kΩ Pull-up zu VDD |

#### Füllstandsensoren

Es gibt hauptsächlich 3 verschiedene Arten von Füllstandsensoren die an den Raspberry Pi Pico angeschlossen werden können

1. kapazitiver Füllstandsensor
2. Schwimmerschalter
3. Ultraschall-Füllstandsensor

Da Schwimmerschalter relativ günstig und zuverlässig sind, werden diese gewählt

Magnetischer Schwimmer-Sensor

TRU COMPONENTS FS-3101 Schwimmerschalter 200 V/AC 0.5 A 1 Schließer, 1 Öffner IP64 1 St.

<https://www.conrad.de/de/p/tru-components-fs-3101-schwimmerschalter-200-v-ac-0-5-a-1-schliesser-1-oeffner-ip64-1-st-1570021.html?utm_source=chatgpt.com#productTechData>

* technische Daten:
  + Kontaktart: 1 Schließer, 1 Öffner
  + Material: Polypropylen
  + Anschluss: Kabel, offenes Ende
  + Kabellänge: 30.00 cm
  + Schaltspannung: 200 V/AC (das ist die Maximalbelastung des Kontakts, der eigentliche Schalter ist ein mechanischer Kontakt, kein aktiver Sensor)
  + Schaltstrom (max.): 0.5 A
  + Leistung: 10 W
  + Min. Temperatur: -20 °C
  + Max. Temperatur: +80 °C
  + Schutzart: IP64
  + Typ: FS-3101

Der Schwimmerschalter ist im Prinzip ein **mechanischer Schalter** mit einem **trockenen Kontakt** (ohne eigene Spannung), der je nach Wasserstand öffnet oder schließt.

Anschluss:

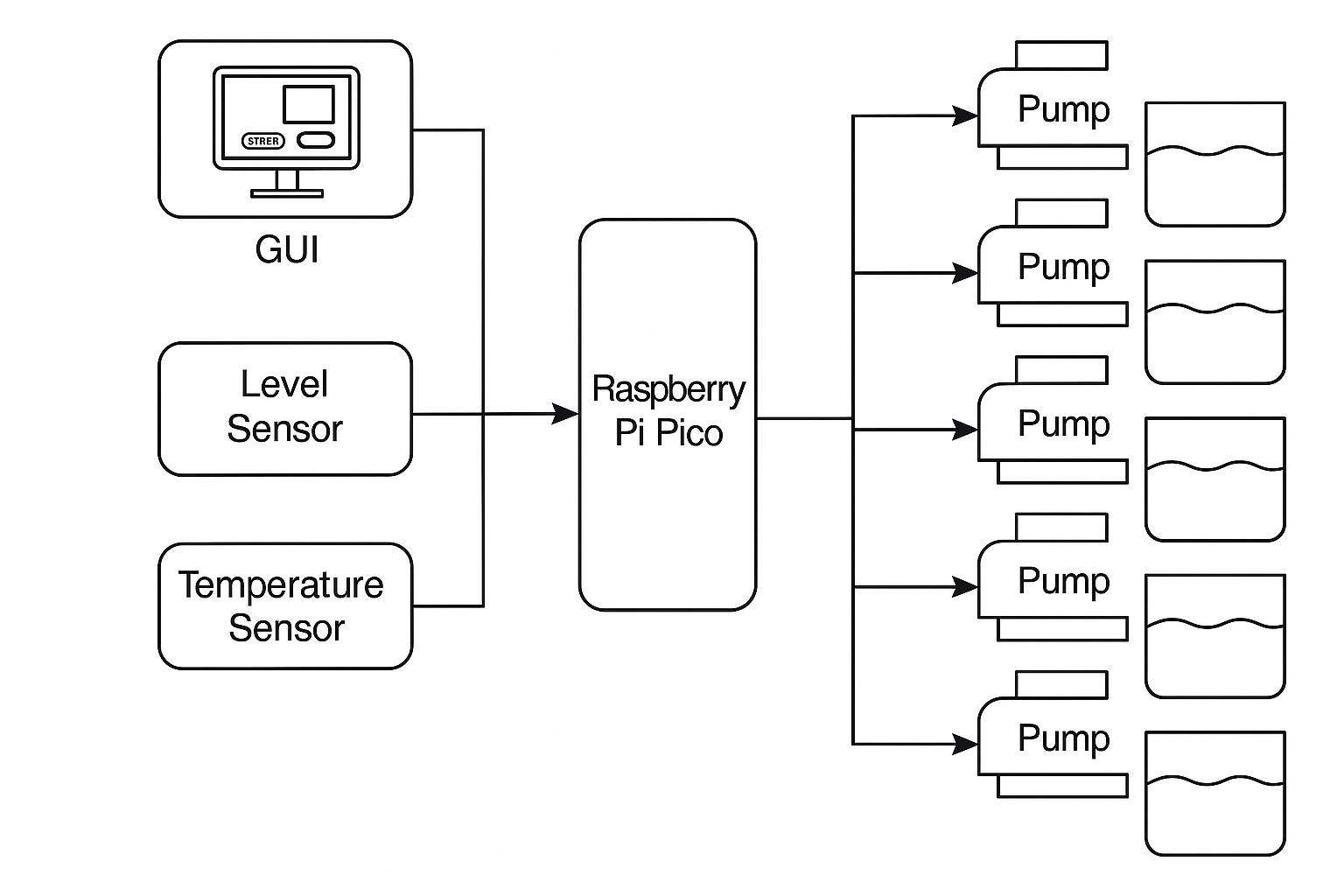
* Verbinde einen GPIO-Pin mit einem Ende des Schwimmerschalters.
* Das andere Ende des Schwimmerschalters kommt auf GND (Masse).
* Der Pull-up-Widerstand (10kΩ) wird zwischen 3,3 V und GPIO-Pin geschaltet, so dass der Pin im Normalfall HIGH (3,3 V) ist.
* Wenn der Schwimmerschalter schließt, zieht er den Pin auf GND (LOW).
* Wenn Schwimmerschalter offen: GPIO-Pin ist durch den Pull-up auf HIGH (3,3 V)
* Wenn Schwimmerschalter geschlossen: GPIO-Pin ist mit GND verbunden → LOW

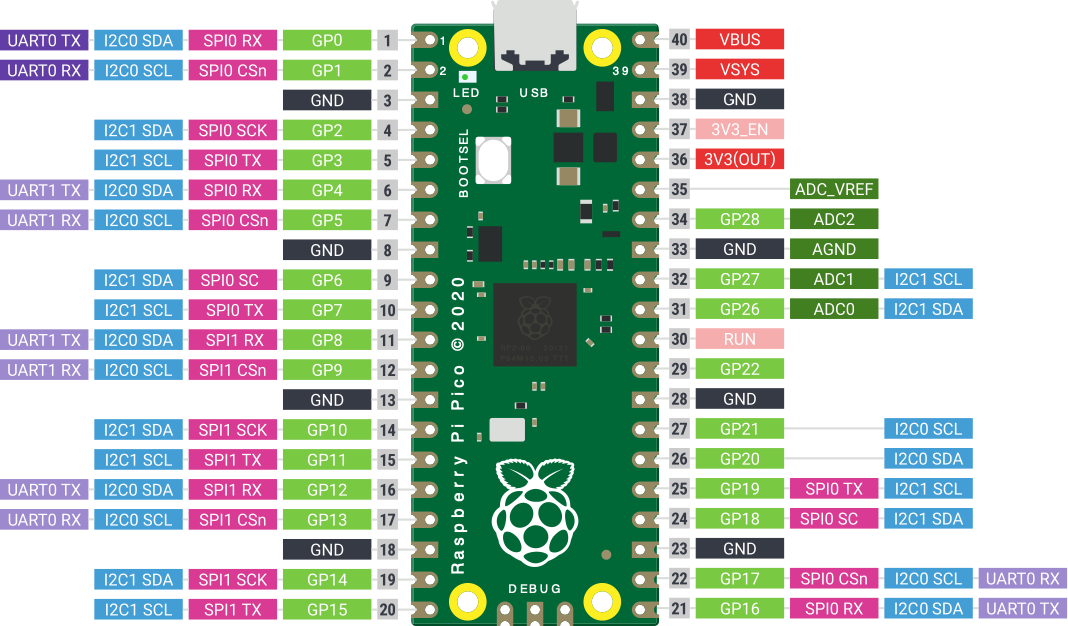
### Schnittstelle

### mechanischer Aufbau

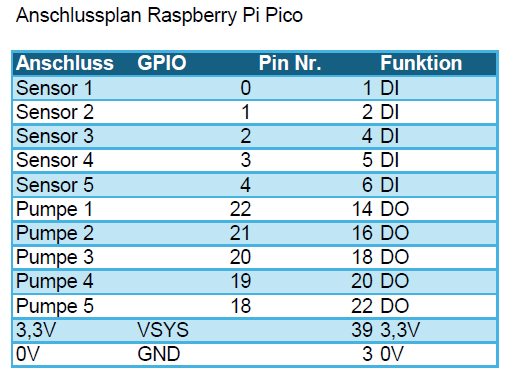
## Schaltplan

Blockschaltbild





Anschlussplan



## Arbeitsweise

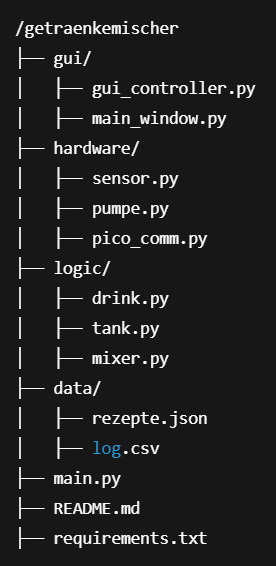
gemeinsame ToDo Liste

wöchentliche Meetings

* Analyse Projektziele und -ergebnisse
* Fortschrittsüberwachung
* Besprechung von Fehlern/Problemen
* Meilensteine/Deadlines setzen

#### GitHub

mögliche Ordnerstruktur:



#### Zeitplan

Woche 1:

* mechanischen Aufbau planen
* welche Bauteile werden benötigt?
* elektrischen Aufbau (Blockschaltbild erstellen)
* Bauteile bestellen

#### Arbeitsaufteilung

* Laurin
  + Bauteilauswahl
  + Schaltplan
  + Algorithmen:
    - Rezept
* Yannik
  + Ansteuerung Raspberry
  + Algorithmen:
    - d
  + Klassen:
* Dean

## Softwarearchitektur

* Kommunikation mit Pico: Seriell via USB (z. B. mit pyserial)
* GUI: Mit tkinter

### Klassen

### Module/Klassen:

### Sensor: liest Werte vom Pico

### Pumpe: steuert Pumpen (Ein/Aus)

### Drink: verwaltet Rezepte

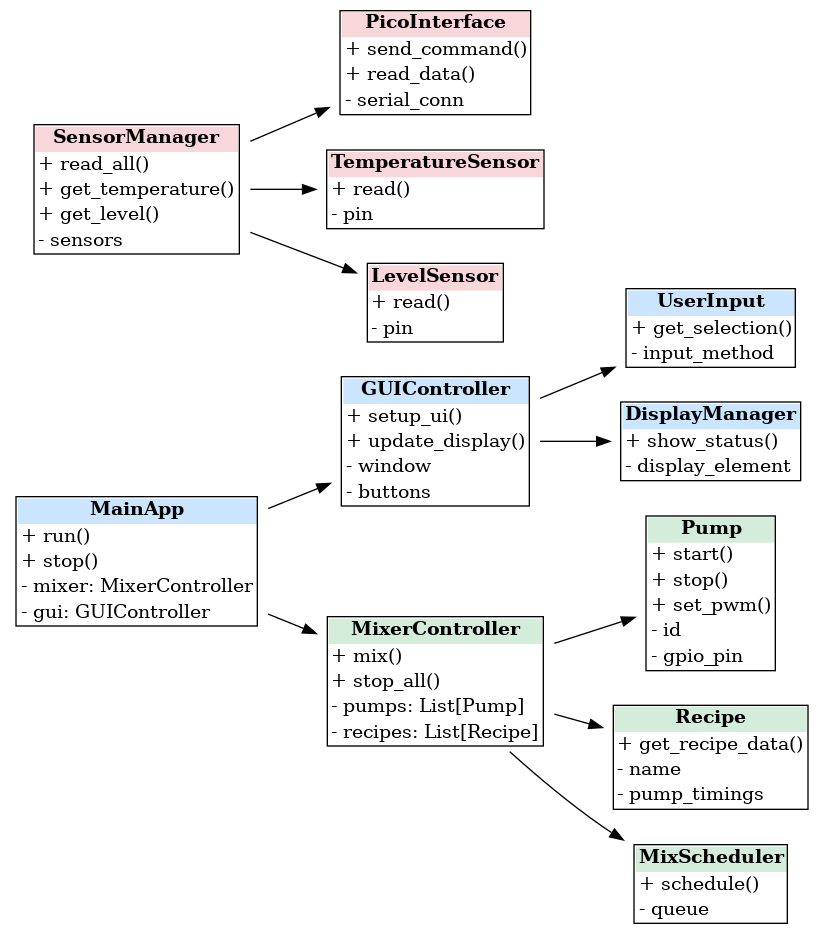
### Tank: speichert Daten zu Füllstand und Temperatur

### MixerController: steuert den Mischvorgang

### GUIController: verbindet GUI mit Logik

### Logger: speichert Temperatur- und Füllstände

#### Klassendiagramm



### Algorithmen

**1. Steuerungs- und Ablaufalgorithmen**

Diese regeln den Mischvorgang:

**🔹 Mischsteuerung (Rezept-Algorithmus)**

* Steuerung der **Reihenfolge und Menge** der Pumpen pro Getränk.
* Z. B. bei Cocktail „Cola-Rum“: 150 ml Cola + 50 ml Rum.

🧠 **Ansatz**:

rezept = {"Cola": 150, "Rum": 50}

for zutatenname, ml in rezept.items():

einschaltzeit = ml / ml\_pro\_sekunde[zutatenname]

pumpe[zutatenname].ein()

sleep(einschaltzeit)

pumpe[zutatenname].aus()

**2. Sensor-Auswertung**

Zur Sicherstellung der Funktion und Sicherheit:

**🔹 Füllstand-Überwachung mit Hysterese**

* Vermeide Flackern bei schwankendem Signal.
* Hysterese = Signalwechsel nur bei stabilem Wert über Zeit.

**🔹 Fehlersensor-Erkennung**

* Wenn z. B. ein Füllstandssensor **zu lange denselben Wert** liefert oder **unerwartete Werte** (z. B. „voll“, obwohl leer), → **Warnung oder Abschaltung**.

**3. Temperaturregelung**

Wenn du Temperatur misst, kannst du:

**🔹 Alarmalgorithmus**

* Bei Über-/Unterschreitung (z. B. >30 °C) gibt es einen **Alarm**.

**🔹 Temperatur-Mittelwertfilter**

* Glättet Sensorwerte über Zeit:

python

KopierenBearbeiten

temperaturwerte = []

def update\_temp(neuer\_wert):

temperaturwerte.append(neuer\_wert)

if len(temperaturwerte) > 10:

temperaturwerte.pop(0)

return sum(temperaturwerte) / len(temperaturwerte)

**4. GUI-Logik**

**🔹 Menüführung / Zustandsmaschine**

* Auswahl von Getränken → Übersicht → Mischvorgang starten
* Beispielzustände:
  + Start, Getränkeauswahl, Mischen, Fehler, Bereit

**🔹 Ereignisbasierte Steuerung**

* „Wenn Button gedrückt → Rezept ausführen“

**5. Sicherheitsalgorithmen**

**🔹 Notabschaltung**

* Wenn Sensor „leer“ meldet, aber Pumpe läuft → sofort Stopp
* Oder: Wenn Temperatur zu hoch

**🔹 Timeout-Steuerung**

* Pumpe läuft nie länger als z. B. 10 Sekunden

**6. Erweiterte Idee (optional)**

Falls du dich tiefer reinarbeiten willst:

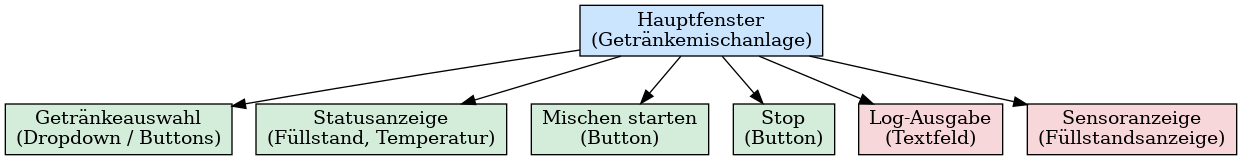
**🔹 Kalibrierungsalgorithmus**

* Bestimme, wie lange eine Pumpe für 100 ml braucht
* Daraus Pumpleistung berechnen → präzisere Steuerung

**🔹 Wartungs-Zyklus**

* Zähle Pumpzyklen → Hinweis: „Behälter auffüllen“ oder „System spülen“

### GUI



## Tests