Keg Washer System Documentation

README: Keg Washer System (Version 5.1.7)

Overview
The Keg Washer System automates the cleaning and sanitizing of beer kegs,
the system is DIY built integrating hardware mostly bought on ali-express and software components.
It features precise cycle control, real-time parameter updates via Firebase, and a user-friendly web interface for monitoring and customization.

Motivations
This system was built buy a brewer who learned to write code and loves DIY, not a machine engineer who learned how to brew… 😊
therefore the machine may not be fully perfect, things could be done better! Please let me know if you have suggestions on Yisrael@atlow.co.il

Features
1. Pre boot Test Cycle: Verifies the functionality of sensors, valves, and relays (~2 minutes).
2. Full Cycle:
   - Air Purge: Clears residual beer using compressed air until the green button is released.
   - Water Rinse: Rinses the kegs with water (~30 seconds).
   - Air Purge: Purges the system post-rinse (~20 seconds).
   - Caustic Rinse: Cleans with caustic solution (~60 seconds). The caustic solution could be replaced with PBW or other cleaning aid of your choice.
   - after Caustic rinse the system is purged with air and rinsed with water (the default is 3 times) (~80 seconds)
   - Sanitization: Uses peracetic acid (~90 seconds). PAA can be replaced with Starsan or sanitizer of your choice.
   -CO2 Purge- Purges the system post-sanitati (~60 seconds).
   - CO2 Pressurization: Builds keg pressure (~45 seconds).
3. Sanitation Cycle: Focuses solely on peracetic acid sanitization (~3 minutes).
4. Purge Cycle: Uses compressed air to clear Full kegs (~2 minutes).

Hardware Requirements
1. Raspberry pi
2. Power Supply:
   - 220VAC input.

- 220VAC to 24VDC converter.
- Power supply for Raspberry Pi.

3. Actuators:
   - 1 pump (220VAC relay-controlled).
   - 16 24V relays for solenoid valves and lights.
   - 2 220vac relays
   - 1 220vac relay with adjustable timer
   - 1 electric switch (to turn off pressure system if wanted)
   - 3 signal lamps
   - 1 on/off switch
4. Sensors:
   - 2 pressure sensors connected via ADS1115.
   - ADC converter
   - 1 water level- float sensor
5. Buttons:
   - Green, red, emergency stop, fill water, and fill caustic tank switches.
6. Heating Element:
   - Controlled via a 220VAC relay.

7. Pneumatic Components
   - 10 pneumatic solenoids
   - 10 pneumatic valves
8. Miscellaneous
   - electric project box
   - 2 keg couplers
   - metal framing for electric box, kegs, pump caustic and PAA tanks

Setup Instructions

Hardware

1. Refer to the wiring diagram (included) for detailed connections between GPIO pins, relays, and components.
2. GPIO assignment:
   - Green BTN = 25
   - Rred BTN = 27
   - Emergancy BTN = 24
   - Water_Level sensor= 10
   - pressure_cancel=18
   - Shutdown Btn=22
   - Fill Caustic switch=23

- Fill Paa Switc=14
- Standby light = 11
- In Proces light = 9
- Eror light =6
- Pump = 13
- Heat = 5
- Main_Drain = 19
- Water_In = 26
- Air_In = 0
- Caustic_In = 15
- Caustic_Out = 7
- Paa_In = 8
- Paa_Out =16
- Co2_In = 21
- keg1= 12
- keg2= 20

2. Ensure all power supplies are stable and securely connected.

Troubleshooting

1. Relay Fails to Trigger: Inspect relay and GPIO pin connections.
2. Button Unresponsive: Confirm GPIO pins and button wiring.
3. Status Lights Not Working: Verify GPIO outputs and light connections.
4. Firebase Connection Issues: Confirm internet connectivity and credentials.
5. kegs don't empty: switch to stronger compressor or use extended purge times.

Software

Python Script

Install dependencies:

```bash
sudo apt-get install python3-pip
pip3 install RPi.GPIO firebase-admin Adafruit-circuitpython-ads1x15 pytz
```

Create a service file to ensure that the keg washer program runs on boot of the raspberry pi:

If you don't want your raspberry pi to run the keg washer program automatically on boot skip the following stages.

~~~
```bash

sudo nano /etc/systemd/system/kegwasher.service

```
~~~

Fill the service file with these details, make sure to mention your own .env with firebase credentials (check firbase setup):

```
[Unit]

Description=My Awesome Service

Wants=network-online.target

After=network-online.target nss-lookup.target


[Service]

EnvironmentFile=/home/raspberry/Desktop/keg-washer-firebase-
adminsdk-7ww1i-aa79>

ExecStart=/usr/bin/python3 /home/raspberry/Desktop/KwV5.py

WorkingDirectory=/home/raspberry/Desktop

StandardOutput=inherit

StandardError=inherit

Restart=on-failure

RestartSec=10s

User=raspberry

Environment="PATH=/usr/bin:/usr/local/bin"

[Install]

WantedBy=multi-user.target
```

After creating the service file, reload systemd to read the new service configuration:

~~~
 ```bash

sudo systemctl daemon-reload

 ```
~~~

Enable the service to start on boot, and then start the service immediately:

~~~
 ```bash

  sudo systemctl enable kegwasher.service sudo systemctl start
myapp.service

 ```
~~~

*****

If you skipped creating the service file- on each time you want to run the keg washer-

Run the script:

~~~
 ```bash

 python3 KwV5.1.7.py

 ```
~~~

pyhton code explained

The Python script for the keg washer system is highly modular, with each function playing a critical role in the automated cleaning, sanitizing, and error-handling process for beer kegs. Below, I've elaborated on each function to give you a more descriptive understanding of their purpose and functionality, organized into logical groups based on their role in the system.

The functions are grouped and ordered as follows:

1. **Cycle Management Functions**: Functions that handle the overall cleaning, sanitizing, and pressurizing cycles, providing an overview of what each cycle entails and detailing the flow of action functions.

2. **Action Functions**: Functions called within cycles to perform specific cleaning or sanitizing actions.

3. **Supporting Functions**: Functions that provide support during cycles, such as pressure checks or conversions.

4. **Background Functions**: Functions that run continuously to support system safety and operation.

5. **Utility Functions**: Functions that manage the system setup, error handling, and overall workflow.

Cycle Management Functions

1. **Cycle():** The `Cycle()` function initiates a complete cleaning, sanitizing, and pressurizing cycle. It sequentially manages all the stages of cleaning, including air purging, water rinsing, caustic rinsing, and final pressurization. This function ensures that the entire cleaning process runs smoothly and thoroughly, leaving the kegs clean, sanitized, and pressurized for immediate use.

Flow of Actions:
- AirPurge(): Begins with purging the keg with compressed air to remove any initial residues.
- WaterSquirt(): Performs a thorough rinse with water to clean out the inside of the keg.
- AirPurge(): Another air purge follows to ensure all water is expelled.
- causticrinse(): Introduces caustic solution to break down organic matter inside the keg.
- AirPurge(): Ensures any remaining caustic is expelled.
- WaterSquirt(): Rinses the keg to remove the caustic residue.
- paasanitize(): Applies peracetic acid (PAA) for final sanitization of the keg.
- Co2purge(): Uses CO2 to expel any remaining PAA and prepare the keg for pressurization.
- kegprssurize(): Pressurizes the keg with CO2 to ensure it is ready for use.

2. **ShortCycle():** The `ShortCycle()` function runs a shortened version of the complete cycle, focusing primarily on quick sanitization. This cycle skips the caustic rinse, making it ideal for situations where a fast turnaround is required, such as when kegs are already mostly clean and only require final sanitization.

Flow of Actions:
- paasanitize(): Applies PAA to sanitize the keg.
- Co2purge(): Uses CO2 to remove the PAA and prepare the keg for use.
- kegprssurize(): Pressurizes the keg to ensure it is ready for immediate use.

3. **purgecycle():** The `purgecycle()` function is designed to clear any residual contents from the kegs without running a complete cleaning cycle. This function is used to empty kegs and prepare them for maintenance or storage by purging with air to ensure all contents are removed.

Flow of Actions:
- AirPurge(): Repeatedly purges the keg with air to ensure all residual contents are removed, preparing the system for the next cycle or maintenance.

Action Functions

1. **AirPurge():**Recure, purgetimeon, purgetimeoff): The `AirPurge()` function is used to purge the kegs with compressed air. It repeats the purge operation a specified number of times (`Recure`), with controlled durations for the purge (`purgetimeon`) and rest (`purgetimeoff`). This step is essential for removing any residual liquid or contaminants from the kegs, ensuring they are empty before proceeding to the next stage of the cleaning cycle. The function also includes pressure checks to verify that the system is operating within safe parameters and the keg actually empties out.

2. **WaterSquirt():** The `WaterSquirt()` function manages the water rinsing process, where the keg is flushed with water to remove any remaining residue. It controls the timing and sequence of water flow, allowing for multiple squirt types—including medium, short, and long squirts—based on predefined settings. This flexibility ensures that the kegs are thoroughly cleaned before caustic rinse, or to neutralize caustic pre sanitization. The function also checks pressure to confirm that the water is being dispensed correctly and safely.

3. **causticrinse():** This function is responsible for the caustic rinse stage of the cleaning process. It manages the flow of caustic solution. It also handles pressure checks to make sure the rinse is proceeding as intended. This stage is critical for breaking down organic matter inside the keg, and the function ensures that all surfaces are in contact with the caustic for an adequate amount of time to achieve effective cleaning.

4. **PumpSquirt(Purge):** The `PumpSquirt()` function controls the pump to squirt either caustic or PAA into the keg for cleaning or sanitation. Depending on the selected chemical it determines the purge gas (air or CO2), it determines the appropriate number of squirts and duration for each squirt. This helps ensure that the kegs are properly cleaned and sanitized with the correct chemical and that the internal surfaces are exposed to the cleaning agents for the optimal amount of time.

5. **paasanitize():** The `paasanitize()` function handles the sanitization process using PAA (peracetic acid). It controls various outputs to pump PAA into the keg, maintaining adequate pressure throughout the process. After sanitizing, it uses CO2 to purge the PAA, ensuring that the keg is ready for use. Peracetic acid is a powerful oxidizing agent used as a sanitizer due to its effectiveness in killing bacteria, viruses, and other pathogens without leaving harmful residues. This function

plays a key role in the final sanitization, ensuring that no harmful bacteria or residues are left inside the keg.

6. **Co2purge():** The `Co2purge()` function uses CO2 gas to purge the kegs after the PAA sanitization step. The function alternates CO2 between the kegs, maintaining proper timing for opening and closing the CO2 valves. This process ensures that all excess liquid or remaining PAA is expelled from the keg, leaving it free from any contaminants or oxygen. The function also includes checks to make sure that sufficient pressure is being maintained throughout the purge.

7. **kegprssurize():** This function pressurizes the keg with CO2 until the desired pressure is reached. It continuously monitors the input and output pressure values to verify that the keg is properly pressurized. If the system detects any anomalies, such as the pressure taking too long to build, it will adjust accordingly to wait a sufficient amount of time or raise an error. This step is crucial for preparing the keg for immediate use in the dispensing process, ensuring that it is properly pressurized to maintain product quality.

Supporting Functions

1**. convert_pressure():** This function reads the raw analog values from the pressure sensors connected to the ADS1115 ADC, converts them into meaningful pressure readings in PSI, and updates global variables for input and output pressure. The conversion takes into account calibration offsets (VLin and VLout) and scales the raw voltage accordingly. Accurate pressure readings are critical for the proper functioning of the cleaning and sanitizing cycles, as they help ensure that the system maintains the necessary pressure during different stages of keg processing.

2. **checkpruessurecanceled():** This function monitors if the pressure check process has been manually canceled by the mechanical switch. It uses a button input to determine whether the pressure metering should be turned off. If canceled, the pressure values are set to extremely high or zero to indicate that the process has been interrupted. This allows operators to cancel pressure checks and pivot the keg washer to work by set times and not by real-time pressure checks.

Background Functions

1. **protectheat():** The purpose of this function is to safeguard the heating element by monitoring the water level. The heating element is switched off when the water level sensor indicates a low level, and it is turned on when the water level is adequate. This ensures that the heating element is never exposed to conditions that could lead to overheating, such as operating without sufficient water. The function runs continuously to provide real-time protection, preventing damage to the equipment and ensuring operator safety.

2. **get_current_settings():** This function is responsible for continuously fetching the latest configuration parameters from the web interface. It retrieves data from the "parameters" collection in Firestore database and updates a shared list of settings used throughout the program. This allows real-time synchronization between the web interface and the physical keg washer system. It ensures that any parameter changes made by users, such as adjusting the cleaning cycle time or changing pressure thresholds, are reflected in the operation without requiring a restart.

3**. filltanks():** The `filltanks()` function is responsible for managing the filling caustic and peracetic acid (PAA) tanks, based on the inputs from GPIO pins. These GPIO pins are connected to switches that can be manually activated to fill the tanks with water, but this feature is only enabled when a cycle is not running. The function is only a semi automatic aid to help fill the tanks and will not shut itsef when tanks are full! As long as the switch is on, the tank will fill!

4. **shutdown():** This function monitors the shutdown button. The shut down button is “pressed” as long as the machine on/off switch is on and power is supplied . Once the on/off switch is turned off- the Raspberry pi still gets power for a minute, but the power button is no longer pressed which powers off the system when it has still has power. It provides a controlled shutdown mechanism to prevent abrupt power losses, which could damage the system. By ensuring that the shutdown process is deliberate, verified, and protects the hardware from any un-wanted damage.

5. **checkbtn():** The `checkbtn()` function monitors the status of the control buttons, such as the green (start), red (stop), and emergency buttons, and updates the global button status variable accordingly. This function is crucial for user interaction, allowing the operator to control the keg washer, pause or stop cycles, or initiate emergency stops if necessary. It provides a direct link between physical button presses and the corresponding system behavior.

6. **Pauseindicator():** This function visually signals that the keg washer system is paused. It turns on the green standby light and repeatedly blinks the orange in-process light until the system resumes operation. This visual cue helps operators understand the system status at a glance and ensures that the system's paused state is clearly communicated, minimizing confusion during manual interventions or troubleshooting.

Utility Functions

1. **Err(ErrNmbr):** This error-handling function is designed to manage and signal different types of errors in the system. Depending on the error number (`ErrNmbr`), it triggers specific outputs, such as activating lights or controlling solenoid valves. Each error corresponds to a specific issue, such as:

- Err #1: Air input failure.
- Err #2: Water input failure.

- Err #3: Caustic input failure.
- Err #4: CO2 input failure.
- Err #5: Emergency button pressed.
- Err #6: Timeout while building pressure.
- Err #7: PAA input failure.

By using visual indicators (red error light) and controlling relevant outputs, the function provides clear feedback to the operator, helping to identify and resolve problems efficiently.

2. **fetch_settings_once():** This function works similarly to `get_current_settings()` but runs only once. It is used to initialize key parameters before the system begins a cycle. By fetching the settings only once, it ensures that the keg washer starts with the correct initial configuration. This setup is essential for configuring operational parameters, such as timing for each cycle phase or pressure limits, providing a stable starting point before entering the main operation loop.

3. **Stdby():** The `Stdby()` function resets the keg washer system to standby mode. It clears all outputs and sets up the GPIO pins for input and output, preparing the system for a new cycle. It also turns on the green standby light, signaling that the system is ready and waiting for user input to begin a cleaning cycle. This function is crucial for ensuring that all components are in a known state before initiating a new operation, reducing the risk of unexpected behavior.

4. **boot():** The `boot()` function initializes critical services and prepares the system for normal operation. It starts necessary safety processes like heat element protection and shutdown monitoring, then runs a series of initial checks to ensure all components are functioning properly. If successful, it calls the `main()` function to start the regular operational flow. This boot process is crucial for ensuring the system is in a stable and ready state before initiating any keg cleaning operations.

5. **main():** The `main()` function acts as the primary operational loop for the keg washer system. It continuously checks the button inputs, handles process flow, manages different cleaning cycles, and controls how the system responds to user inputs. It also manages the transition between different states, such as pausing, resuming, or stopping cycles, and starts appropriate processes based on button interactions. This function is essential for orchestrating the various subprocesses and ensuring the entire system operates as intended.

In summary, the keg washer Python script is meticulously designed to manage the complete automation of keg cleaning and sanitization, ensuring safety, efficiency, and proper functionality at each stage. Each function plays a specific role, from monitoring inputs and handling errors to controlling valves, pumps, and indicators, all while maintaining real-time synchronization with the Firebase database for dynamic operation control.

Frontend Description

This web application serves as the interface for managing and controlling the automated keg washer system. The platform integrates user authentication, real-time parameter updates, real-time logs, and detailed control over the keg washing cycle stages, facilitating operational transparency and control.

- Deploy the web interface locally or on a server using:

~~~
```bash

python3 -m http.server

```
~~~

using github pages is a good option as well.
make sure to change the port address to your servers port (se Backend description)

Backend Description

The backend, written in `index.js`, is a Node.js application using the Express framework. It handles Firebase authentication and database interactions. Key features include user login, parameter updates, and error handling. The backend ensures seamless communication between the frontend and the keg washer hardware via Firebase.

a good option for running the Backend server is Renedr io, that could build and deploy a node server.

Make sure to create a cornjob on cornjob.com to ping the server and keep it "awake" every 15 minutes.

Firebase Setup Guide

1. Go to Firebase Console and create a new project.
2. Add your app to the Firebase project and generate configuration files.
3. Enable Firestore and set up collections like `parameters` and `buttons` as required.
4. Enable email/password authentication in the Firebase Console.
5. Generate a private key JSON file for Firebase Admin SDK and update the backend and Python scripts with their location and environment secrets.

Wiring Diagram

Refer to the provided wiring diagram for setup and GPIO pin connections.

For more details, refer to the User Manual.

User Manual: Keg Washer System

Chapter 1: Introduction

The Keg Washer System simplifies keg cleaning and sanitization through automated cycles and user-friendly controls.

Chapter 2: System Overview

1. Power System: 220VAC input with a 12V converter.
2. Control System: Raspberry Pi managing GPIO inputs and outputs.
3. Frontend: A web interface for parameter updates and monitoring.

Chapter 3: Hardware Setup

Refer to the wiring diagram for detailed connections between GPIO pins, relays, and components.

Chapter 4: Button Functions, and In-Cycle Behavior, Signal Lamps

- **Green Button**: Starts the full cycle from standby. During air purge, releasing it halts the air purge step.
- **Red Button**: Starts the sanitation cycle from standby. Has no effect during an active cycle.
- **Both Buttons**: Starts the purge cycle from standby. During an active cycle, has no effect.
- **Emergency Stop**: Stops all operations immediately, regardless of the current cycle.
- **Fill Water Switch**: Activates water tank filling during standby or active cycle, provided it doesn't interrupt safety.
- **Fill Caustic Switch**: Activates caustic tank filling during standby or active cycle, provided it doesn't interrupt safety.

Button	Function During Cycles
Green Button	Starts the full cycle from standby. During air purge, releasing it halts the air purge step.
Red Button	Starts the sanitation cycle from standby. Has no effect during an active cycle.
Both Buttons	Starts the purge cycle from standby. During an active cycle, has no effect.
Emergency Stop	Stops all operations immediately, regardless of the current cycle.
Fill Water Switch	Activates water tank filling during standby or active cycle, provided it doesn't interrupt safety.
Fill Caustic Switch	Activates caustic tank filling during standby or active cycle, provided it doesn't interrupt safety.

Cycle

Stages and Button Behavior

1. **At Power On**:
   - **Emergency Stop Button**: Emergency stop fault.
   - **Green Button**: Start cycle test.
2. **Waiting to Start Test Cycle**:
   - **Emergency Stop Button**: Emergency stop fault.
   - **Green Button**: Start cycle test.
3. **During Test Cycle**: All buttons are inactive.
4. **Standby**:
   - **Emergency Stop Button**: Emergency stop fault.
   - **Green Button (Long Press)**: Initiates full cycle.
   - **Red Button (Long Press)**: Initiates keg drain.
   - **Both Buttons (Long Press)**: Initiates short cycle (sanitization, co2 purge).

Stage	Emergency Stop BTN	Green BTN	Red BTN	Green+Red
At power on	Emergency stop fault	Start cycle test	-	-
Waiting to start test cycle	Emergency stop fault	Start cycle test	-	-
During test cycle	-	-	-	-
Standby	Emergency stop fault	Long press - initiates full cycle	Long press - initiates keg drain (1 min)	Long press - initiates short cycle (sanitization, rinsing)
During full cycle	Emergency stop - Emergency stop fault	Pause	Stop - return to standby	-
During pause	Emergency stop - Emergency stop fault	Long press - resumes full cycle	Stop - return to standby	-
During drain cycle	Emergency stop - Emergency stop fault	Stop - return to standby	-	-
During short cycle	Emergency stop - Emergency stop fault	Pause	Stop - return to standby	-
After pressure fault	Emergency stop fault	Return to standby	-	-
After emergency stop fault	Emergency stop - Emergency stop fault	Return to standby	-	-

Signal Lamps

- **Green Light**: System is in standby mode.
- **Orange Light**: System is in a cycle (including short cycle and purge cycle).
- **Red Light**: Error detected, or emergency stop.

Signal lamp/Mening		green	orange	red
Fetching data from server	Red blinking + Orange stable + Green off		🟧	blinking
Wating for green buton to start test cycle	Endles carusel- Each lamp is on for a second and 2 other are off	Blinking 1 at a time	Blinking 1 at a time	Blinking 1 at a time
During Test cycle	Green on + Orange on + Red off	🟩	🟧	
test cycle sucsess	All lights blink 3 times	blinking	blinking	blinking
Standby	Green light on	🟩		
During Cycle (including short cycle & purge cycle)	Orange light on		🟧	
Paused Cycle	Green Light blinking	blinking		
Error	Red light on/blinking			🟥

Chapter 5: Cleaning Cycles

1. Preboot Test Cycle (~1.5 minutes): Tests all connected sensors and valves.
2. Full Cycle (~10 minutes): Air purge, water rinse, air purge, caustic rinse, sanitization, CO2 pressurization.
3. Sanitation Cycle (~3 minutes): Uses peracetic acid to sanitize kegs.
4. Purge Cycle (~2 minutes): Clears kegs with air.

*for more detail and depth, check REDME "pyhton code explained"

Chapter 6: Maintenance

1. Tighten hose bands to prevent leaks.
2. Clean pump surroundings regularly.
3. Inspect pneumatic valves and solenoids for blockages or wear.
4. Check all sensors (pressure, water level) for correct readings.
5. Regularly drain and clean caustic and peracetic acid tanks.
6. Inspect electrical connections and tighten loose wiring.

Chapter 7: web fronted interface

Overview

This web application serves as the interface for managing and controlling the automated keg washer system. The platform integrates user authentication, real-time parameter updates, real-time logs, and detailed control over the keg washing cycle stages, facilitating operational transparency and control.

Key Features

1. **User Authentication:**
   - Secure login using email and password ensures only verified users can change parameters.
   - User session management to ensure data security and privacy.
2. **Parameter Management:**
   - Update and save operational parameters for the keg washing process on real-time.
   - Access and reset to default settings as needed.
   - View historical parameter updates for traceability and preferred set of parameters.
   - The application divides the keg washing process into multiple stages, such as air purging, water rinsing, caustic cleaning, and sanitization.
   - Each stage's parameters (e.g., duration, repetitions, flow rates) can be adjusted to suit specific operational needs.
3. **Real-Time Logs and Monitoring:**
   - Integrated live logs that provide updates from the keg washer in real-time as a console to the raspberry pi.
   - View system logs directly from the interface for troubleshooting and performance monitoring.
4. **Language Support:**
   - Multi-language interface for enhanced accessibility, with a toggle for switching between Hebrew and English.

Usage Guide

1. **Getting Started:**
    - Log in with your credentials. If you're a new user, ensure your account is registered with the system administrator. You can check if your account is registered by trying reset your password.
    - Once logged in, review the operational parameters displayed on the dashboard.
2. **Adjusting Parameters:**
    - Navigate through the different stages of the keg washing process.
    - Modify values for each parameter, such as duration and repetitions, as needed.
    - Save the settings to ensure the changes are applied.
3. **Resetting to Defaults:**
    - Use the "Reset to Defaults" option to quickly revert parameters to their default values if needed
4. **Logs, previously saved values $ live Console:**
    - Access the "previously saved values" section to view parameters saved by you or other users previously, click on preview to see the parameter set, or "load" to load the parameters to the keg washer.
    - Access the "keg Washer Logs" to view each keg washer session (date, turned on, amount of kegs rinsed/purged/only sanitized, turned off)
    - Access the "Washer console" section to view live updates from the keg washer.

5. **Logout:**
6. Always log out after completing your session to secure your data and settings.

Chapter 8: FAQs

1. Why isn't the system starting? Check power supply and ensure the script is running.
2. What do the status lights mean? Checkout table in chapter 4.
3. How can I update cycle parameters? Use the frontend to adjust values in Firebase.
4. What if the buttons don't work? Inspect GPIO connections and replace faulty switches.
5. How do I reset parameters? Use the "Reset to Defaults" option in the frontend.
6. Why is the error light on? Check logs for specific error messages.

7. How can I view the cycle history? Use the Firebase interface to access logs.
8. Can I add more cycles?   Modify the script and update the frontend accordingly.

Chapter 9: Safety Protocols

General Guidelines

1. Always wear appropriate PPE (gloves, goggles, protective clothing) when handling caustic chemicals and peracetic acid.

2. Ensure the emergency stop button is functional and accessible before starting the machine.

3. Regularly inspect hoses and connections for leaks or damage.

Emergency Stop

- The emergency stop button is connected to GPIO 24. When pressed, all operations cease immediately.

- Ensure it is tested weekly to guarantee proper functionality.

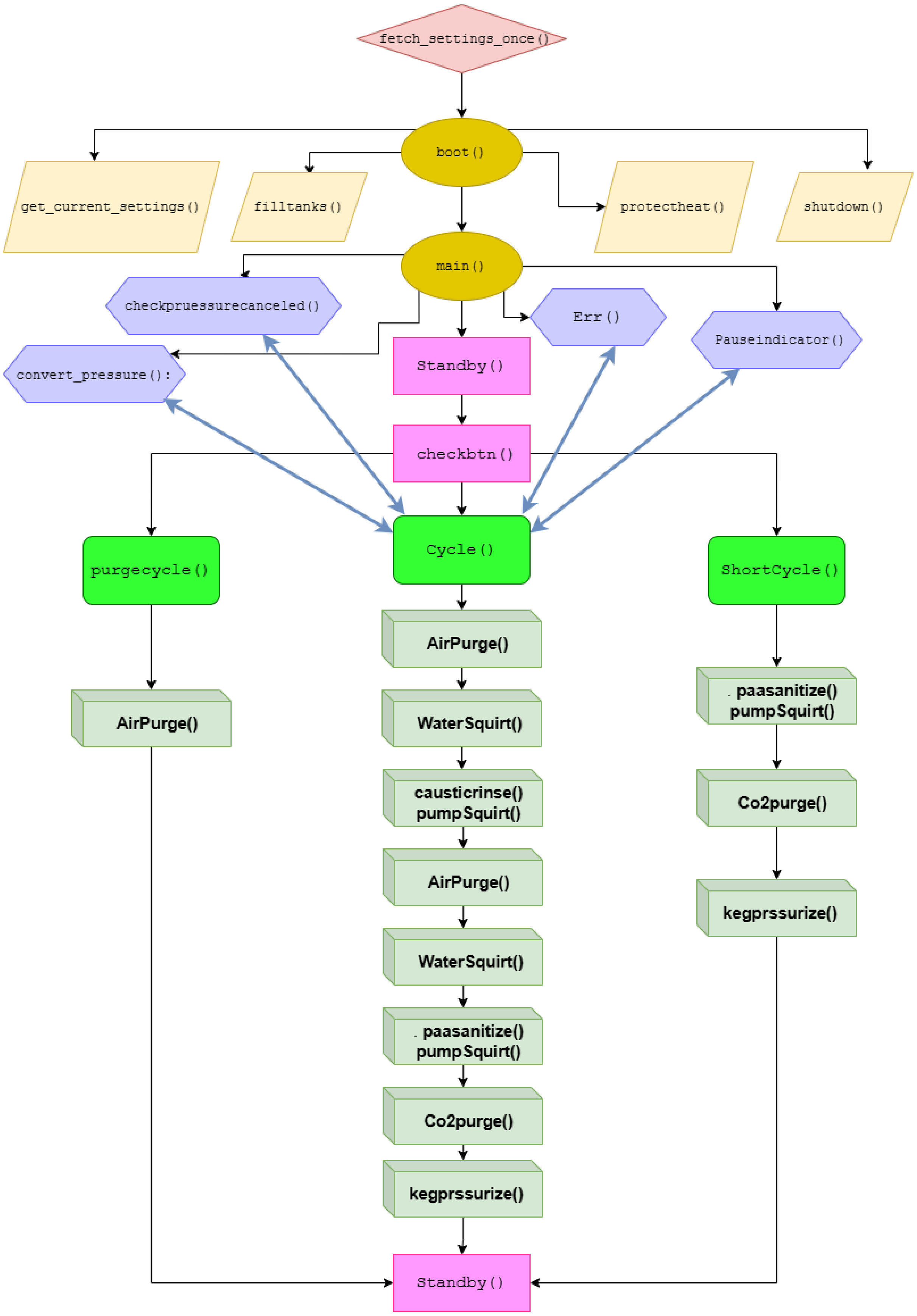
Handling Chemicals

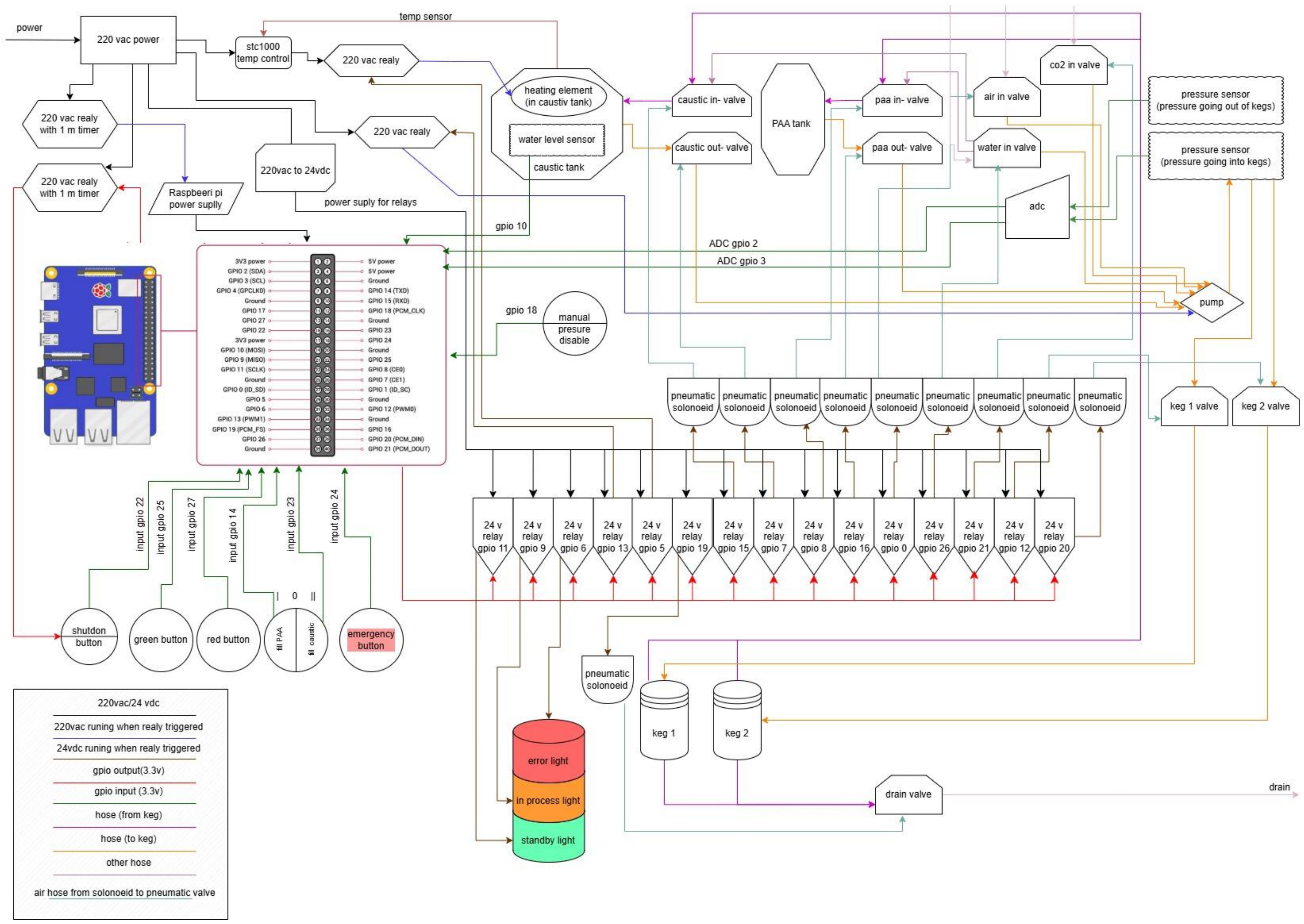
1. Follow MSDS guidelines for caustic soda and peracetic acid.

2. Store chemicals in designated, ventilated areas, away from heat sources.

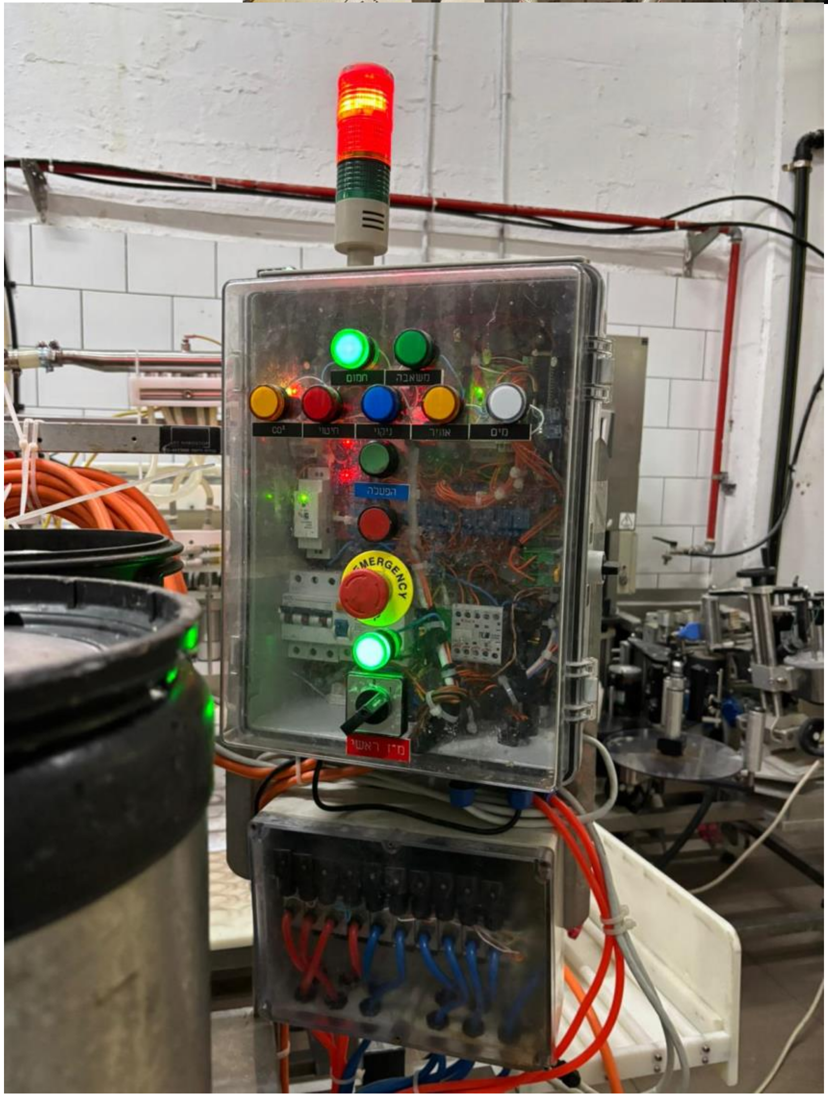
3. Clean spills immediately with appropriate neutralizing agents.

Chapter 10: flow charts, diagrams and pictures




power
220 vac power
220 vac realy with 1 m timer
220 vac realy with 1 m timer
stc1000 temp control
220 vac realy
220 vac realy
220vac to 24vdc
Raspbeeri pi power suplly
power suply for relays
temp sensor
heating element (in caustiv tank)
water level sensor
caustic tank
gpio 10
caustic in- valve
caustic out- valve
PAA tank
paa in- valve
paa out- valve
air in valve
water in valve
co2 in valve
pressure sensor (pressure going out of kegs)
pressure sensor (pressure going into kegs)
adc
ADC gpio 2
ADC gpio 3
pump
gpio 18
manual presure disable
3V3 power
GPIO 2 (SDA)
GPIO 3 (SCL)
GPIO 4 (GPCLK0)
Ground
GPIO 17
GPIO 27
GPIO 22
3V3 power
GPIO 10 (MOSI)
GPIO 9 (MISO)
GPIO 11 (SCLK)
Ground
GPIO 0 (ID_SD)
GPIO 5
GPIO 6
GPIO 13 (PWM1)
GPIO 19 (PCM_FS)
GPIO 26
Ground
5V power
5V power
Ground
GPIO 14 (TXD)
GPIO 15 (RXD)
GPIO 18 (PCM_CLK)
Ground
GPIO 23
GPIO 24
Ground
GPIO 25
GPIO 8 (CE0)
GPIO 7 (CE1)
GPIO 1 (ID_SC)
Ground
GPIO 12 (PWM0)
Ground
GPIO 16
GPIO 20 (PCM_DIN)
GPIO 21 (PCM_DOUT)
input gpio 22
input gpio 25
input gpio 27
input gpio 14
input gpio 23
input gpio 24
shutdon button
green button
red button
fill PAA
fill caustic
| 0 ||
emergency button
pneumatic solonoeid
24 v relay gpio 11
24 v relay gpio 9
24 v relay gpio 6
24 v relay gpio 13
24 v relay gpio 5
24 v relay gpio 19
24 v relay gpio 15
24 v relay gpio 7
24 v relay gpio 8
24 v relay gpio 16
24 v relay gpio 0
24 v relay gpio 26
24 v relay gpio 21
24 v relay gpio 12
24 v relay gpio 20
keg 1 valve
keg 2 valve
error light
in process light
standby light
keg 1
keg 2
drain valve
drain
220vac/24 vdc
220vac runing when realy triggered
24vdc runing when realy triggered
gpio output(3.3v)
gpio input (3.3v)
hose (from keg)
hose (to keg)
other hose
air hose from solonoeid to pneumatic valve

EMERGENCY


חמום
משאבה
CO²
חיטוי
ניקוי
אוויר
מים
הפעלה
EMERGENCY
מ"ז ראשי