



POLITÉCNICO
DE PORTALEGRE

Escola Superior
de Tecnologia
e Gestão

Destilador Inteligente

Sérgio Manuel Figueira do Carmo

TRABALHO DE FINAL DE CURSO

Curso: Engenharia Informática

Orientador: Sérgio Correia

Ano Letivo: 2023 | 2024

Setembro | 2024

Agradecimentos

A elaboração e realização deste projeto contou com o apoio de pessoas extremamente importantes, sem elas, nada seria possível de ser concretizado, pelo que desejo, desde já, exprimir os meus sinceros agradecimentos a todos os que me ajudaram pois permitiram que este trabalho se tornasse realidade.

Em primeiro lugar, quero agradecer aos meus pais e ao meu irmão, que me apoiaram desde o primeiro dia, sem eles, certamente não estaria aqui, um muito obrigado!

Igualmente, pela sua dedicação, disponibilidade, simpatia, amizade, conselhos, orientação, tudo, esteve sempre presente, disponível em tudo, tornou possível que nos encontrássemos presencialmente praticamente diariamente nos últimos meses antecedentes á apresentação do projeto, um obrigado muito especial ao Sr. Professor Sérgio Duarte Correia.

Um agradecimento especial ao meu primo e Eng. João Rodrigues, foi graças a ele e a sua motivação que me inscrevi no presente curso de Engenharia Informática.

Não posso também deixar de parte, todos os meus amigos, que também me apoiaram desde o primeiro dia.

Quero também agradecer ao FabLab que cedeu o espaço para as reuniões presenciais, montagem da componente elétrica do destilador e uso das suas ferramentas.

Um agradecimento também ao Sr. Eng. Pedro Pinto pelo seu tempo e dedicação na criação e impressão das porcas em 3D para os sensores de nível de água.

Resumo

Este projeto, tem como finalidade modernizar e automatizar um destilador eletromecânico inicialmente avariado e com um funcionamento completamente analógico. A modernização da máquina consistiu em remover todos os componentes antigos e substituídos por componentes digitais, tal como LEDs (*light emitting diode*), relé e um microprocessador chamado de ESP32, com exceção dos principais atuadores.

Perante um levantamento de necessidades inicial, foi selecionado um módulo processador *embedded*, assim como todos os componentes digitais a inserir na plataforma física. O trabalho realizado envolveu todo projeto dos componentes físicos, a sua instalação, cablagem e teste.

Foi desenvolvido o firmware necessário à gestão e operação do equipamento em C/C++, nomeadamente com a programação de páginas web para a interface de utilizador.

Todo o funcionamento do equipamento foi testado, tanto do ponto de vista dos seus componentes físicos, como a nível do firmware e interface web, envolvendo os *stakeholders* e futuros utilizadores do equipamento, o laboratório de Química da ESTGD/IPP.

Palavras-Chave: AJAX, Arduino, C/C++, Computação *Embedded*, Destilação, ESP32, HTML, JS

Abstract

The main goal of this project is to modernize and automate an electromechanical distiller that was initially broken and operated entirely analogically. The modernization of the machine consisted of removing all old components and replacing them with digital components, such as LEDs (light emitting diodes), relays and a microprocessor called ESP32, except for the main actuators.

Based on an initial needs assessment, an embedded processor module was selected, as well as all digital components to be inserted into the physical platform. The work carried out involved the entire design of the physical components, their installation, wiring and testing.

The firmware necessary for the management and operation of the equipment was developed in C/C++, namely with the programming of web pages for the user interface.

The entire operation of the equipment was tested, both from the point of view of its physical components and at the level of the firmware and web interface, involving the stakeholders and future users of the equipment, the Chemistry Laboratory of ESTGD/IPP.

Keywords: AJAX, Arduino, C/C++, Distillation, Embedded Computation, ESP32, HTML, JS

Lista de Abreviaturas, Siglas e Símbolos

A	Amps,
AC	Alternating current,
AJAX	Asynchronous JavaScript and XML,
AP	Access Point
CSS	Cascading Style Sheets,
DC	Direct current,
GPIO	General-Purpose Input Output,
HTML	Hypertext Markup Language,
IDE	Integrated Development Environment,
JS	JavaScript,
JSON	JavaScript Object Notation,
LED	Light-emitting diode,
NC	Normally Closed,
NO	Normally Open,
SBC	Single Board Computer
SoC	System-on-Chip,
SPDT	Single Pole, Double Throw,
SPI	Serial Peripheral Interface,
SRAM	Static Random Access Memory,
SSID	Service Set Identifier
V	Volts,
XML	Extensible Markup Language,

ÍNDICE GERAL

1.	INTRODUÇÃO, Enquadramento, e Justificação do Tema	2
2.	Metodologia e Meios Utilizados	3
3.	Limitações da Pesquisa/Trabalho	3
4.	Plataforma Física.....	5
4.1.	Arquitetura Física da Plataforma	5
4.2.	Controlador Digital.....	7
4.3.	Componente Eletromecânica	11
4.4.	Sensores	12
4.4.1.	Filtragem de Ruído das Entradas	12
4.4.2.	Depósito e Sensores de Nível	14
4.4.3.	Botões Frontais.....	15
4.5.	Atuadores	16
5.	Software	20
5.1.	Análise de Requisitos.....	20
5.1.1.	Requisitos Funcionais.....	21
5.1.2.	Requisitos Não-Funcionais	22
5.2.	Arquitetura Geral	23
5.2.1.	Diagrama de Casos de Uso	23
5.2.2.	Diagrama de Atividade do Ciclo de Programa.....	24
5.2.3.	Modelo de Camadas do Firmware.....	27
5.3.	- Modelo de Software.....	29
5.3.1.	Modelo de ficheiros	30
5.3.2.	Dependências	30
5.4.	Ligar a Uma Rede Wi-Fi Externa Nova.....	31
5.5.	Ligar ao AP do Destilador (<i>SoftAP</i>)	33
5.6.	Página web.....	34
5.7.	Modelo de Funcionamento	36
6.	Testes	40
6.1.	Prototipagem em Placa de Ensaio.....	40
7.	CONCLUSÃO.....	41
	REFERÊNCIAS BIBLIOGRÁFICAS.....	42
	ANEXOS	44
	Anexo 1 – Instruções de Uso Geral	45

Anexo 2 – Instruções de Uso do Wi-Fi	47
Anexo 3 – Esquema Elétrico	49

ÍNDICE DE FIGURAS

Figura 1 - Destilador	5
Figura 2 – Arquitetura Geral do Destilador	6
Figura 3 - Relógio Analógico	7
Figura 4 - Contactor	7
Figura 5 - Controlador de Nível Analógico	7
Figura 6 – Layout dos pins do ESP32	10
Figura 7 - Fonte 5VDC e proteção DC.....	12
Figura 8 - Ponteiras Isoladoras e Alicate de Cravar	12
Figura 9 - Barramento Central	12
Figura 10 - Barramento Lateral.....	12
Figura 11 - Esquema Elétrico do Filtro Passa-Baixo Passivo.....	13
Figura 12 - Esquema Elétrico do ESP32 com o Filtro Passa-Baixo Implementado.....	14
Figura 13 - Funcionamento dos Sensores de Nível [8].....	14
Figura 14 - Porcas M14 3D	15
Figura 15 - Reservatório de Água.....	15
Figura 16 - Digital Inputs	15
Figura 17 - Botões do Painel Frontal	16
Figura 18 - Esquema Elétrico das Bobinas dos Relés	17
Figura 19 – Módulos de Relês.....	17
Figura 20 - Esquema Elétrico dos Contactos dos Relés.....	17
Figura 21 - Relé do Modo Manual	18
Figura 22 - Esquema Elétrico do Relé do Modo Manual	18
Figura 23 - Bomba de Água	18
Figura 24 – Válvula.....	19
Figura 25 - Esquema Elétrico Indicadores	20
Figura 26 - Indicadores LED	20
Figura 27 - Diagrama de Casos de Uso.....	24
Figura 28 - Diagrama de Atividade do Ciclo de Programa.....	25
Figura 29 - Esquema Representativo das Camadas do Firmware	27
Figura 30 – Ligação à Rede Externa	32
Figura 31 - Botão de Reset ao Wi-Fi.....	33
Figura 32 - Página Web com Imagens.....	35
Figura 33 - Página Web com Botões	35
Figura 34 – Diagrama de Sequência do funcionamento de Utilizador	37
Figura 35 - Uso em Modo Manual	38
Figura 36 - Uso em Modo Automático	39
Figura 37 - ESP32 na Placa de Ensaio	40
Figura 38 - Esquema de Ligação do ESP32 na Placa de Ensaio	40

ÍNDICE DE TABELAS

Tabela 1 - ESP32 GPIO's.....	9
Tabela 2 - GPIO INPUTS.....	10
Tabela 3 - GPIO OUTPUTS	11
Tabela 4 - Componentes Usados Placa de Ensaio	40

1. INTRODUÇÃO, ENQUADRAMENTO, E JUSTIFICAÇÃO DO TEMA

O projeto "Destilador Inteligente" insere-se no contexto da modernização e automação de equipamentos laboratoriais. A crescente procura por soluções automatizadas nos laboratórios reflete a necessidade de otimizar operações, reduzir falhas humanas e garantir a repetibilidade dos resultados, especialmente em áreas como a química e a engenharia.

Este trabalho justifica-se pela necessidade de restaurar e atualizar um destilador eletromecânico obsoleto, inicialmente avariado e com funcionamento completamente analógico. A modernização proposta não só visa recuperar o equipamento, mas também introduzir funcionalidades avançadas que ampliem a sua utilidade e segurança. A substituição dos componentes analógicos por digitais, incluindo LEDs, relés, e um microprocessador ESP32, permite a automatização de tarefas que anteriormente dependiam exclusivamente de intervenção manual, reduzindo erros e aumentando a eficiência do processo.

A escolha do ESP32 como unidade central do sistema baseia-se nas suas características técnicas, como a capacidade de processamento, conectividade Wi-Fi, e compatibilidade com a programação em C/C++. Estas características possibilitam o desenvolvimento de uma interface web para o controlo e monitorização do equipamento, facilitando o uso e a interação por parte dos operadores. Assim, o projeto atende à crescente necessidade de atualização tecnológica dos laboratórios, proporcionando uma solução moderna, acessível e de fácil implementação.

Este relatório aborda a metodologia utilizada para a modernização do destilador, abrangendo desde a seleção de componentes e o desenvolvimento do software até a realização de testes que validaram o funcionamento da solução proposta. O destilador modernizado não só resgata a funcionalidade de um equipamento essencial, mas também o eleva a um novo patamar de desempenho, alinhado com as exigências tecnológicas dos dias de hoje.

O documento está organizado em 7 capítulos. Após capítulos introdutórios ao nível das metodologias e limitações, o Capítulo 4 apresenta uma descrição detalhada da componente física do destilador. O Capítulo 5, aborda o desenvolvimento do *firmware* necessário para o funcionamento do destilador tal como a página web. De seguida, o Capítulo 6 aborda os testes efetuados no projeto e por fim no Capítulo 7, será apresentado uma conclusão sobre todo o trabalho desenvolvido.

2. METODOLOGIA E MEIOS UTILIZADOS

A multidisciplinaridade do presente projeto implicou a utilização de um conjunto de ferramentas e metodologias bastante plurais, nomeadamente:

- i. O *firmware* do destilador é programado em linguagem C, inclui os ficheiros de inclusão (ficheiros *de cabeçalho* com a extensão .h). A página *WEB* é programada em HTML com os respetivos ficheiros de CSS para adicionar estilos e os ficheiros JS, para executar JavaScript.
- ii. Todo o projeto foi elaborado no IDE *Visual Studio Code v1.92.2* com as extensões *PlatformIO IDE v3.3.3*, *GitHub Copilot v1.223.0* e *Doxygen v1.0.0*.
- iii. Todo o projeto à exceção do relatório foi também elaborado no sistema operativo *Fedora 40 (Linux)*, em que o relatório foi elaborado no *Microsoft Office* tal como as tabelas.
- iv. O esquema elétrico foi elaborado no *QElectroTech v5.15.7*.
- v. Os diagramas foram desenhados no *software Draw.io v24.7.5*.

Todo o projeto, está também disponível na página no GitHub do autor:
<https://github.com/primisc/IntelligentDestiller>

3. LIMITAÇÕES DA PESQUISA/TRABALHO

Não foram encontradas muitas dificuldades/limitações na pesquisa acerca do trabalho. A comunidade Arduino é muito grande e por consequência, existe muita documentação

disponível. Talvez por haver tamanha documentação, tenha havido uma certa dificuldade em filtrar o que realmente se adequasse às necessidades de concretização do projeto.

4. PLATAFORMA FÍSICA

4.1. Arquitetura Física da Plataforma

O ponto de partida do presente projeto foi um equipamento existente conforme será descrito posteriormente, totalmente analógico, embora não se encontrando em funcionamento (ver Figura 1). Tendo em conta o facto de a maioria dos equipamentos internos serem obsoletos e não ser possível a sua substituição, optou-se por remover os componentes analógicos, mantendo-se os atuadores e interruptores. Todos os restantes componentes foram substituídos, nomeadamente interfaces (contactores) e indicadores luminosos, e adicionados novos componentes para alimentação, proteção, controlo digital.



Figura 1 - Destilador

A arquitetura geral do novo destilador, do ponto de vista do equipamento físico, é representada pelo esquema de processos da Figura 2. Este é constituído por um depósito com água destilada e alimentado através de uma bomba (BMB), e controlado com três sensores de nível (SMIN, SMAX e ALARM). Dentro do depósito existe uma resistência de aquecimento (RAQ) que vai atuar sobre a água destilada presente no

depósito até ao seu ponto de ebulação. O vapor é controlado por uma válvula elétrica (V2) que o direciona para o depósito com a amostra objeto da destilação. Uma segunda válvula elétrica (V1) é acionada permitindo a entrada de água fria no condensador. Esta vai arrefecer o vapor vindo da amostra provocando a sua condensação. Existe também uma terceira válvula (V3), manual, servindo para retirar a água destilada existente no depósito, permitindo a sua manutenção.

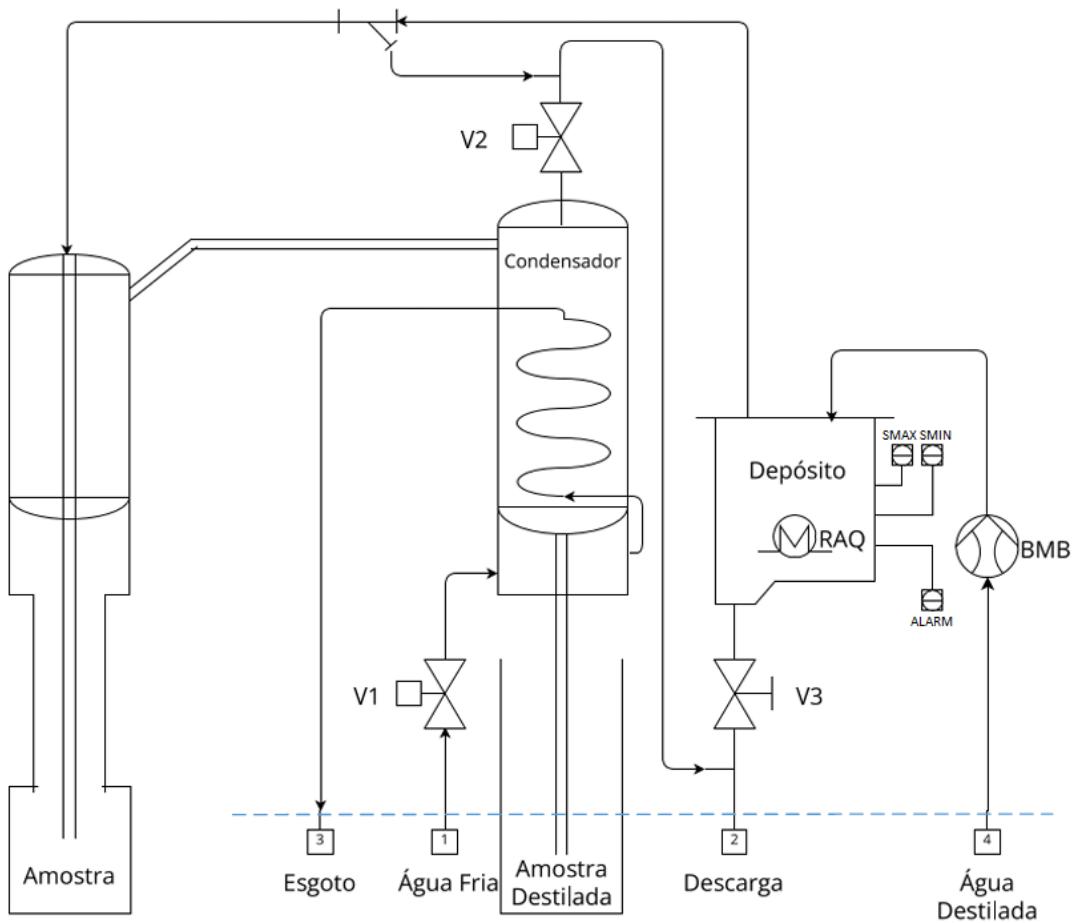


Figura 2 – Arquitetura Geral do Destilador

O equipamento era inicialmente constituído por um temporizador (Figura 3), um contactor (Figura 4), dois controladores de nível analógicos (Figura 5), indicadores luminosos, e diversos botões. Tanto os atuadores como os sensores tinham uma tensão de operação de 230V/50Hz, tendo sido removidos alguns destes componentes como o temporizador analógico, contactor e cabos. Estes foram trocados por componentes

digitais, tal como relés, indicadores LED, um módulo digital ESP32 WROVER-E, e uma fonte de alimentação de 230VAC/50Hz com saída a 5VDC.



Figura 5 - Controlador de Nível Analógico



Figura 3 - Relógio Analógico

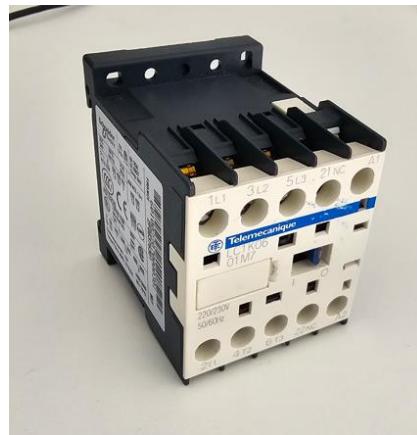


Figura 4 - Contactor

4.2. Controlador Digital

O módulo de processamento usado no projeto é um *ESP32 WROVER-E*, representado pela Figura 6. Este é um SoC (*System on a Chip*) desenvolvido pela empresa *Espressif Systems*, sendo um microcontrolador de baixo custo e baixo consumo de energia [1].

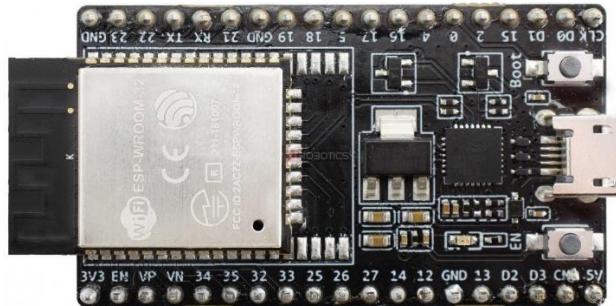


Figura 6 - ESP32-WROVER-E

O microcontrolador ESP32 é compatível com a *framework* de programação Arduino (linguagem de programação C/C++), por isso, pode ser programado pelos mesmos IDE's. No caso deste mesmo projeto, foi programado na plataforma *Visual Studio Code* [2] com o plug-in *PlatformIO* [3] em linguagem de programação C/C++.

O ESP32 tem um processador com 2 núcleos que podem ser controlados individualmente e a frequência pode ser ajustada entre 80MHz e 240MHz [4, p. 7], 520KB de SRAM (*Static Random Acess Memory*) e 4MB de memória FLASH. O ESP32 tem também incorporado um módulo de Wi-Fi e Bluetooth com antena incorporada, sendo o Wi-Fi capaz de suportar os protocolos 802.11 b/g/n (802.11n até 150Mbps). A memória SRAM é um tipo de memória volátil usada em sistemas computacionais, sendo que esta memória usa circuitos *flip-flop* que assegura que os dados fiquem presentes enquanto seja alimentada com energia elétrica. Este tipo de memória tem a vantagem de oferecer alta velocidade de acesso e ser energeticamente muito eficiente. No caso da memória FLASH, esta é uma memória não volátil que pode ser apagada e reprogramada, que serve para armazenar o *firmware* desenvolvido. Esta memória, pode ser dividida em partições e estas serem redimensionadas às necessidades do programador [5]. No caso do presente projeto, foi repartido da seguinte forma:

- Partição para APP – 3145728 bytes,
- Partição para SPIFFS – 917504 bytes.

A partição APP é onde o *firmware* vai ser armazenado e a partição SPIFFS é onde podem ser armazenados ficheiros diversos [6]. No caso deste projeto são armazenados todos os ficheiros relacionados com a página WEB (*.html, *.css, *.js, *.png).

Em relação à gestão de entradas e saídas (ou GPIO – *General Purpose Input/Output*), cada pino pode ser configurado como *inputs* para ler dados de por exemplo sensores, botões, etc., ou podem ser definidos como *output* para atuar sobre o estado de um componente, como por exemplo um motor, válvula, ou indicador luminoso (ver Figura 6).

O módulo usado tem 32 GPIO's, dos quais 19 estão disponíveis para serem usados como é constatado pela Tabela 1, onde tons de cor verde representam GPIO's que podem ser usados tanto para *inputs* como para *outputs*. Os GPIO's 0 e 1 não podem ser usados como *inputs* e os GPIO's 3, 34, 34-36 e 39 não podem ser usados como *outputs* (tons de cor amarela na Tabela 1). Os GPIO's de 6 até 11 são reservados para interação com a memória FLASH, não podendo ser usados como I/O (tons de cor vermelha na Tabela 1). Esta organização limita a capacidade de input/output do equipamento e será uma restrição ao mapeamento a ser efetuado.

Tabela 1 - ESP32 GPIO's

GPIO	Input	Output	Notes
0	pulled up	OK	outputs PWM signal at boot, must be LOW to enter flashing mode
1	TX pin	OK	debug output at boot
2	OK	OK	connected to on-board LED, must be left floating or LOW to enter flashing mode
3	OK	RX pin	HIGH at boot
4	OK	OK	
5	OK	OK	outputs PWM signal at boot, strapping pin
6	x	x	connected to the integrated SPI flash
7	x	x	connected to the integrated SPI flash
8	x	x	connected to the integrated SPI flash
9	x	x	connected to the integrated SPI flash
10	x	x	connected to the integrated SPI flash
11	x	x	connected to the integrated SPI flash
12	OK	OK	boot fails if pulled high, strapping pin
13	OK	OK	
14	OK	OK	outputs PWM signal at boot
15	OK	OK	outputs PWM signal at boot, strapping pin
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		input only
35	OK		input only
36	OK		input only
39	OK		input only

ESP32-DevKitC

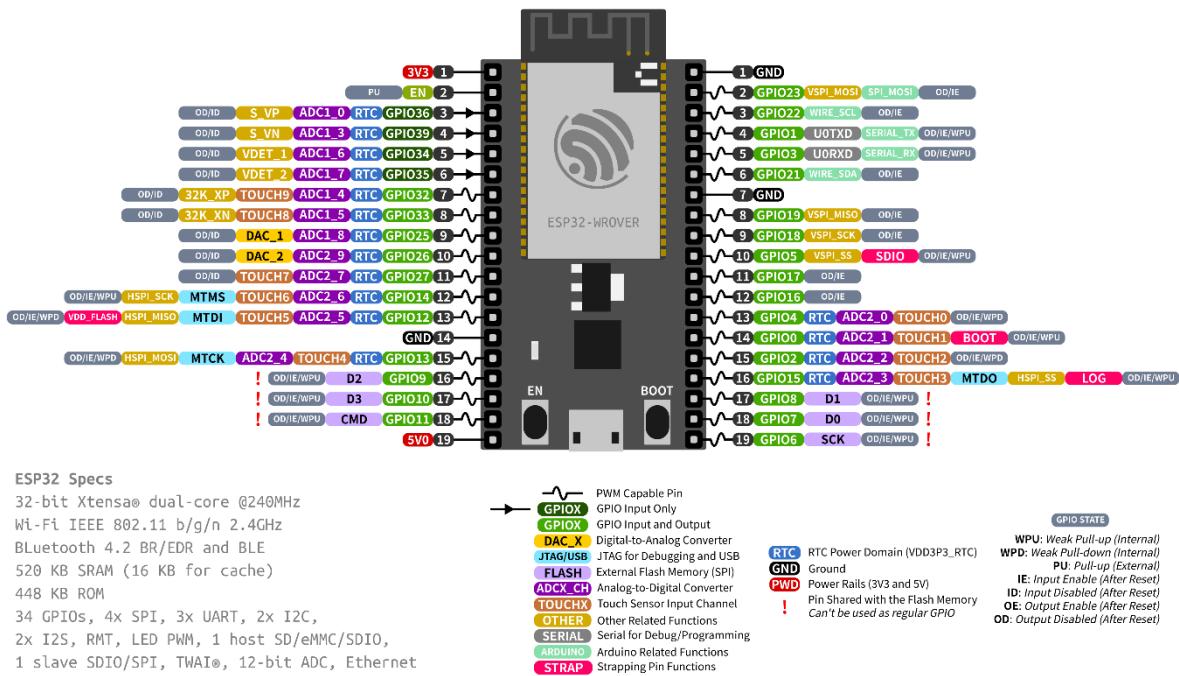


Figura 6 – Layout dos pins do ESP32

Para controlo e gestão dos periféricos necessário ao destilador, foram usados cinco *inputs* (ver Tabela 2) e nove *outputs* (ver Tabela 3), de um total de dezanove GPIO's.

Tabela 2 - GPIO INPUTS

GPIO	Símbolo	Descrição
14	PIN_SMAX	Sensor de nível água
25	PIN_SALARM	Sensor de nível de alarme
26	PIN_SW_MAN	Interruptor de modo manual
27	PIN_SMIN	Sensor de nível água mínimo
32	PIN_SW_AUTO	Botão de modo auto

Tabela 3 - GPIO OUTPUTS

GPIO	Símbolo	Descrição
4	PIN_IND_MAX	Indicador luminoso de nível água máximo
5	PIN_IND_MIN	Indicador luminoso de nível água mínimo
15	PIN_IND_AUTO	Indicador luminoso do modo auto
18	PIN_IND_ALARM	Indicador luminoso de nível de água de alarme
19	PIN_VALV_WATER_IN	Válvula de entrada de água fria
21	PIN_VALV_WATER_OUT	Válvula de descarga de vapor
22	PIN_BMB	Bomba de água
23	PIN_RAQ	Resistência de aquecimento
33	PIN_IND_MAN	Indicador luminoso do modo manual

4.3. Componente Eletromecânica

A componente eletromecânica tem um papel muito relevante no destilador. É ela que vai ser complementada com a componente de *software*. Como já foi referido no capítulo 4.1 (Arquitetura), alguns componentes foram removidos, nomeadamente um relógio temporizador, um contactor, dois controladores de nível analógicos, e toda a cablagem. Estes componentes, estavam danificados ou em muito mau estado de conservação. Foram mantidos outros componentes como os botões, a resistência de calor (RAQ), a bomba de água (BMB), e as válvulas (V1 e V2), por se apresentarem com um bom estado de funcionamento. Como foram removidos componentes, estes tiveram de ser substituídos por novos elementos com o mesmo tipo de funcionalidade. Foi adicionado um ESP32 para substituir o relógio temporizador, relés para substituir o contactor, e sensores de nível digitais para substituir os controladores analógicos. Os cabos foram completamente substituídos e usadas ponteiras isoladoras (Figura 8) em todos os cabos. Foi adicionado um barramento lateral (Figura 10) dedicado somente para a alta e baixa tensão, albergando também a fonte de alimentação, a proteção para baixa tensão através de fusível de 1A e um disjuntor bipolar de 16A (Figura 7). Foi ainda adicionado outro barramento central para ligação dos sensores de nível de água, o relé do botão do modo manual, e um barramento de terra (Figura 9).



Figura 10 - Barramento Lateral



Figura 9 - Barramento Central



Figura 8 - Ponteiras Isoladoras e Alicate de Cravar



Figura 7 - Fonte 5VDC e proteção DC

4.4. Sensores

4.4.1. Filtragem de Ruído das Entradas

Como se está a trabalhar com alta tensão alternada (240V AC) junto com baixa tensão (5V DC) e o microprocessador é muito sensível a ruído nos sinais de *input*, foi detetado nos testes em que ao ligar/desligar o botão de modo manual, os *outputs* aleatoriamente trocavam de estado, sem que fosse comandado. Depois de vários testes, foi detetado com recurso a um osciloscópio que ao ser acionado botão de modo manual, este

causava ruído de alta frequência nos I/Os, já que os fios agem como antenas aos campos eletromagnéticos gerados pela alta tensão nos fios, e pelas bobinas dos relés [7].

Com o diagnóstico foi realizada uma separação de todos os cabos internos da melhor maneira possível, separando os fios de alta e baixa tensão, o que diminuiu o problema, mas não o solucionou. Realizando-se uma pesquisa na literatura, o problema foi resolvido com a implementação de um filtro passa-baixo passivo (ver Figura 11), que visa suprimir as altas frequências geradas pelos componentes e fios que operam em alta tensão. Este problema surge quando são ligados componentes que geram campos magnéticos por consumirem muita corrente que neste caso, correspondia a ligar as bobinas dos relés. Esta ação gerava um pico de corrente que por sua vez, gerava ruído nos fios de baixa tensão.

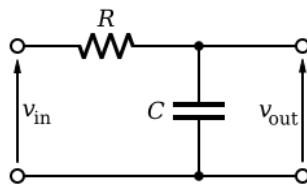


Figura 11 - Esquema Elétrico do Filtro Passa-Baixo Passivo

Foi então aplicada a Expressão (1) e como foi usado um condensador de tântalo de 220nF e uma resistência de 470OHMs, conforme Figura 12, o resultado da equação é aproximadamente $\sim 15392\text{Hz}$, sendo esta a frequência máxima de corte do filtro. O filtro dimensionado foi adicionado um filtro para cada *input* individual

$$f_c = \frac{1}{2\pi RC} \quad (1)$$

Ligados diretamente na saída da fonte de alimentação foi adicionado aos terminais de 5V DC um condensador eletrolítico de 1000uF/16V, e um condensador de tântalo de 220nF, para filtragem adicional de ruído na linha de alimentação.

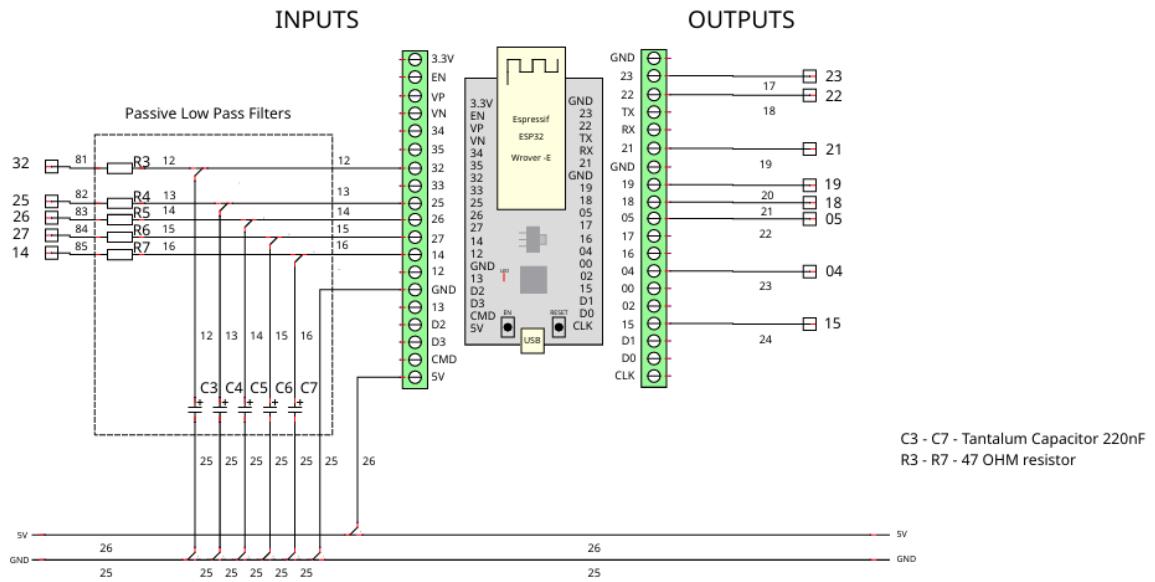


Figura 12 - Esquema Elétrico do ESP32 com o Filtro Passa-Baixo Implementado

4.4.2. Depósito e Sensores de Nível

O vapor de água destilada é gerado dentro de um depósito em inox. Este depósito incorpora uma resistência de aquecimento alimentada a 230VAC/50Hz, e três sensores de nível.

Os sensores de nível são sensores de reflecção, usam um emissor infravermelho para deteção de líquido conforme Figura 13, são eletricamente alimentados com 5VDC e quando ativos, respondem com um sinal de 5VDC, por isso podem ser ligados diretamente aos pinos do ESP32 para serem processados (ver Figura 16).

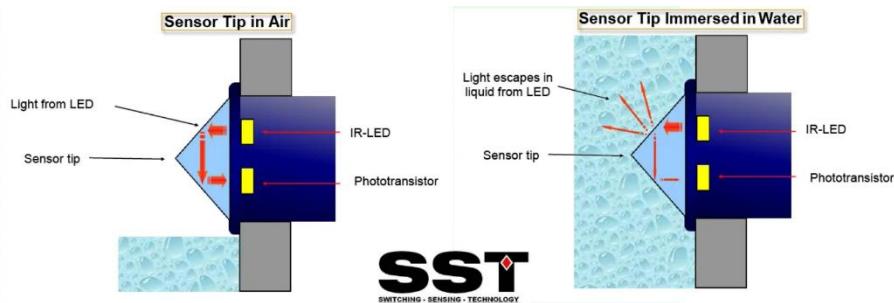


Figura 13 - Funcionamento dos Sensores de Nível [8]

Os sensores de nível, estão localizados na lateral do depósito através de furações feitas na parede do depósito (ver Figura 15) e estão seguros com porcas M14 (Figura 14) impressas no laboratório FabLab com uma impressora 3D de resina.



Figura 14 - Porcas M14 3D



Figura 15 - Reservatório de Água

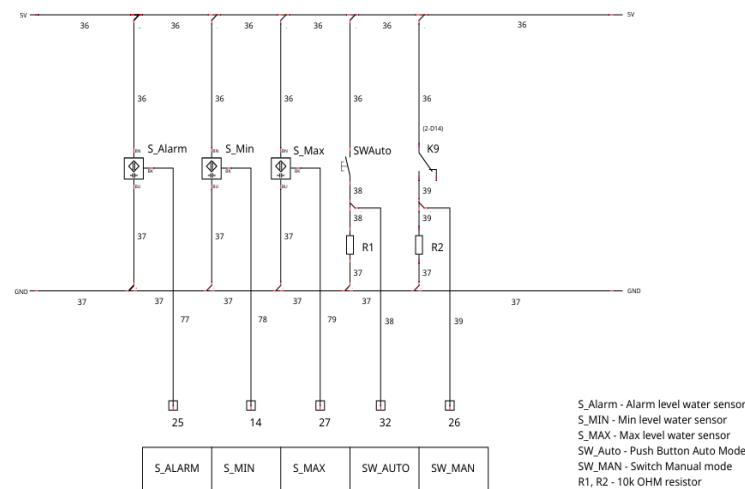


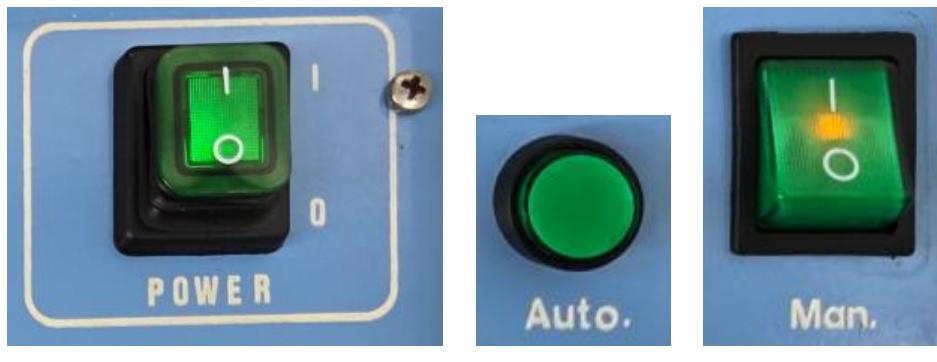
Figura 16 - Digital Inputs

4.4.3. Botões Frontais

O interruptor de “POWER” (Figura 17.a) liga/desliga todo o equipamento. Este é um interruptor bipolar de dois estados, isto é, interrompe a passagem de corrente elétrica na fase e neutro, tem um indicador incorporado de néon de 250VAC.

O botão “Auto” (Figura 17.b) é um botão de pressão (*push-button*), e, portanto, só passa energia quando pressionado. Ao ser pressionado, ativa/desativa o modo automático.

O interruptor “Man.” (Figura 17.c) é idêntico ao interruptor de “POWER” referido anteriormente, e tem também um indicador néon incorporado, pelo que, é atuado na linha de 230AC.



a) Interruptor de “POWER”

b) Botão “Auto.”

c) Interruptor “Man.”

Figura 17 - Botões do Painel Frontal

4.5. Atuadores

Para controlar os componentes de saída (ou atuadores), foram adicionados dois módulos de relés, cada um com 4 relés individuais, com saídas SPDT (*Single Pole, Double Throw*), conforme a Figura 19. O módulo da esquerda está dedicado a controlar os componentes comandados a 250VAC, como a bomba de água (BMB), as duas válvulas (V1 e V2) e a resistência de aquecimento (RAQ). O módulo da direita, controla os indicadores LED frontais (indicadores de nível de água máximo, mínimo, alarme e modo automático), sendo estes atuadores comandados a 5VDC. Foi necessário adicionar relés aos indicadores frontais LED, mesmo sendo estes comandados a 5VDC, para não sobrecarregar o ESP32, já que a corrente de saída recomendada de cada pino é de 20mA.



Figura 19 – Módulos de Relês

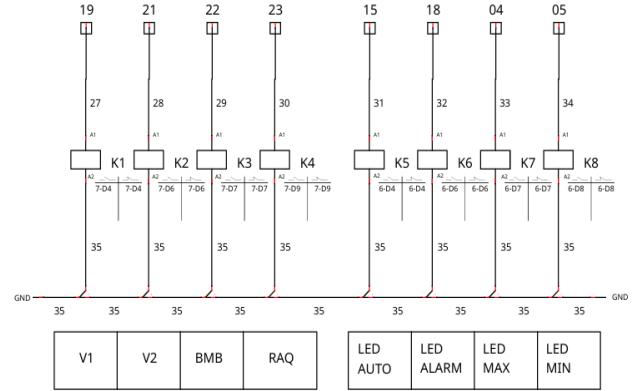


Figura 18 - Esquema Elétrico das Bobinas dos Relés

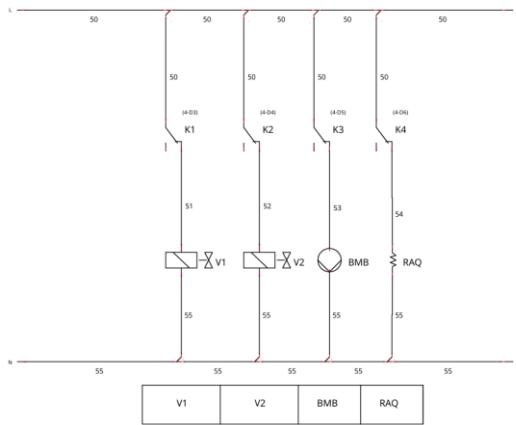
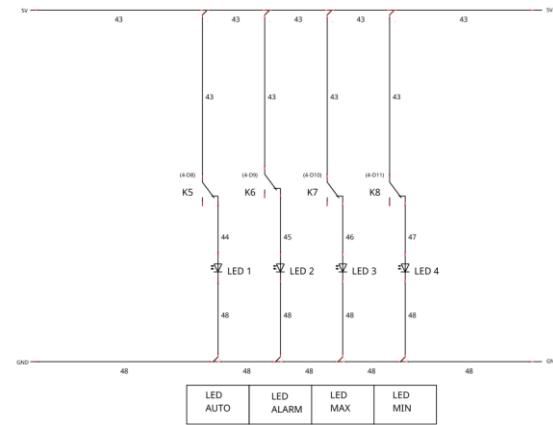


Figura 20 - Esquema Elétrico dos Contactos dos Relés



Para aproveitar o botão do modo manual, que está em perfeito estado de funcionamento, e porque tem um indicador embutido de néon de 250VAC/50Hz, foi adicionado um relé (Figura 21) dedicado somente ao botão de modo manual, em que ao ser ligado, aciona o relé para o microprocessador poder receber essa informação (ver Figura 20), e tendo em conta que o microcontrolador não pode receber mais que 5VDC diretamente nos pinos. A ligação ao microcontrolador tem um filtro passa-baixo passivo, como já foi referido em “Filtragem de Ruído das Entradas” para remover qualquer ruído causado pelo circuito de alta tensão ou até mesmo por alguma interferência eletromagnética externa.

Para realizar a sua ligação ao I/O do ESP32 é necessário usar um relé de interface conforme esquema da Figura 22.

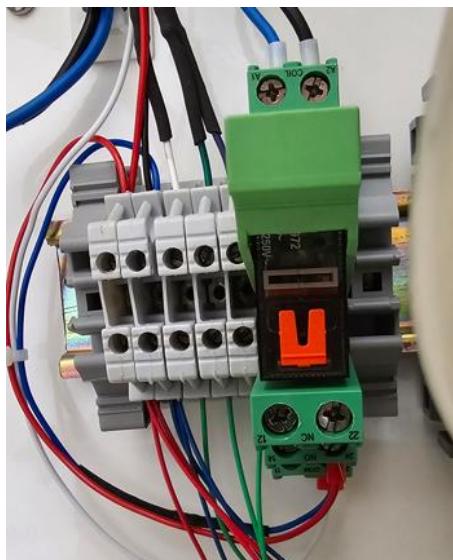


Figura 21 - Relé do Modo Manual

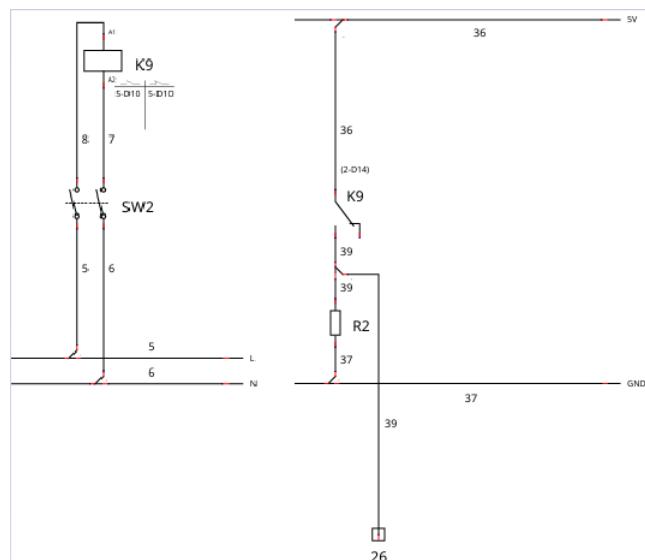


Figura 22 - Esquema Elétrico do Relé do Modo Manual

A água é bombeada por uma bomba do tipo oscilante que opera a 230VAC/50Hz e consome 0.17A representada na Figura 23. Tendo em conta que a bomba opera a 230VAC, foi necessária sua ligação por meio de um relé que atua como interface ao ESP32.



Figura 23 - Bomba de Água

Existem duas válvulas solenoide (ver Figura 24) com tamanho 1/8" acionadas a 240VAC/50Hz que são NC (*normally closed*). A válvula "V1" controla a entrada de água para arrefecer a serpentina do condensador de vapor, e a válvula "V2" controla a saída de vapor.



Figura 24 – Válvula

Existem quatro indicadores LED (ver Figura 26) no painel frontal da máquina, são todos acionados com 5VDC, e são todos operados pelo segundo módulo de relés (ver Figura 25). São usados dois indicadores de cor laranja, outro de cor verde, e 1 de cor vermelha. Enquanto o modo automático usa um indicador de cor laranja, o indicador verde está responsável pelo nível máximo de água no reservatório. Por sua vez, o nível mínimo tem associada a cor laranja, e o nível de alarme tem associada a cor vermelha, acendendo de modo intermitente para destacar a prioridade de atenção.



Figura 26 - Indicadores LED

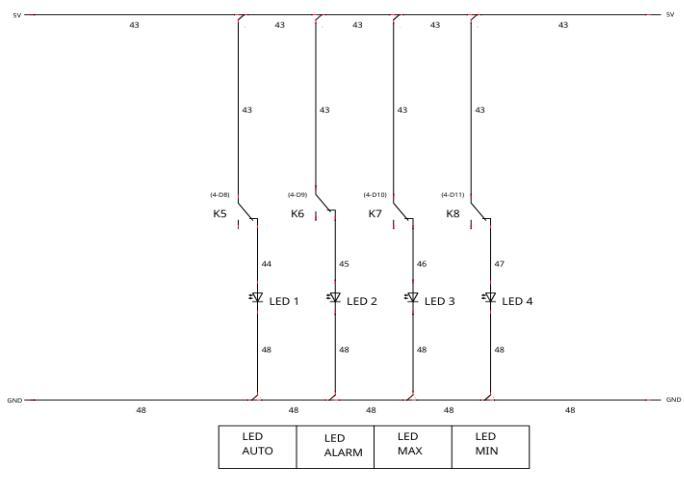


Figura 25 - Esquema Elétrico Indicadores

5. SOFTWARE

5.1. Análise de Requisitos

Os requisitos funcionais e não funcionais do projeto destacam a utilização da placa ESP32 WROVER-E DevKitC V4 como principal escolha, devido às suas características técnicas. Esta contém 26 pinos de I/O, conectividade Wi-Fi integrada de até 150 Mbps, e uma CPU dual-core que permite uma separação eficiente entre o *firmware* que controla o processo físico, e o servidor web. Apesar dessas vantagens, há limitações significativas como a memória de programa (4MB Flash), ou o número de pinos que impede o uso de módulos adicionais como um cartão SD. A restrição ao nível da memória limita a implementação da página WEB a tecnologias leves como HTML, CSS e JavaScript. Alternativas como o Raspberry Pi e Arduino Mega foram consideradas, mas apresentam desvantagens, como número insuficiente de pinos de I/O no caso do *Raspberry* ou menor desempenho de CPU no caso do *Arduino Mega*.

A tarefa de levantamento de requisitos iniciou-se com reuniões com os futuros utilizadores do laboratório de química, onde se procurou definir os objetivos do projeto e os requisitos funcionais.

5.1.1. Requisitos Funcionais

- 1) O destilador deverá funcionar num “Modo Automático”. Isto significa que:
 - a) Será possível ligar/desligar um temporizador na página web, que ser ativo:
 - i) Liga a resistência de aquecimento (RAQ) para formar vapor se o nível de água estiver nos parâmetros ótimos,
 - ii) Liga a bomba de água (BMB) se o nível de água descer abaixo do nível mínimo,
 - iii) Abre as válvulas (V1 e V2) para arrefecer o condensador e sair o vapor para a amostra, respetivamente.
 - b) Efetue a gestão automática do nível de água no depósito que vai funcionar da seguinte forma:
 - i) A resistência de calor (RAQ) liga se o nível de água estiver entre o nível máximo e o nível mínimo.
 - ii) A bomba de água (BMB) liga quando o nível de água estiver abaixo do nível mínimo e desliga quando chegar ao nível máximo,
- 2) Também deverá ser possível operar o equipamento em “*Modo Manual*”. Isto significa poder controlar todos os atuadores do equipamento individualmente através da página web com as seguintes restrições.
 - a) Não é possível ligar a bomba de água (BMB) se o nível de água estiver no nível máximo,
 - b) Não é possível ligar a resistência de calor (RAQ) se o nível de água estiver num nível de alarme.
- 3) Controlo do destilador via página WEB
 - a) Pretende-se criar maior usuabilidade através do controlo do equipamento usando a página web, onde será possível controlar:

- i) Relógio temporizador para que o destilador desligue ao fim de um tempo específico,
- ii) O modo automático é permitido ser ativado e desativado através da página web, o que se este modo for desativado, interrompe o processo de destilação se este estiver ativo.
- iii) Será possível ativar ou desativar os atuadores individualmente se o modo manual estiver ativo, embora com as restrições acima descritas.

5.1.2. Requisitos Não-Funcionais

- 1) O projeto vai usar um microcontrolador *embedded* com a designação de “*ESP32 WROVER-E DevKitC V4 board*” pelo que foram medidas as seguintes vantagens e desvantagens:
 - a) Vantagens:
 - i) O ESP32 conta com 26 I/O pins usáveis,
 - ii) WIFI integrado - 802.11n (2.4 GHz), até 150 Mbps,
 - iii) Conta com um microprocessador com 2 cores (core 0 corre *firmware* da máquina e core 1 o servidor WEB),
 - iv) O microprocessador trabalha de 40MHz até 240MHz,
 - v) A *Firmware* é desenvolvida em C.
 - vi) A arquitetura do CPU é *open source*, o que é possível programar ao nível do CPU se necessário, ao contrário se fosse usado, por exemplo, um raspberry pi
 - b) Desvantagens:
 - i) A memória para programar o *firmware* em conjunto com a página WEB disponível é muito limitada (4MB flash),
 - ii) Por limitação de memória disponível no ESP32, velocidade de processamento e I/O disponíveis, apenas é possível fazer a página WEB em

HTML usando CSS, e JavaScript, não havendo mais pins I/O disponíveis para ligar um módulo para cartão SD.

- iii) Limitação de velocidade do microprocessador (40MHz até 240MHz)
- c) Foram analisadas outras alternativas ao microprocessador *ESP32 WROVER-E DevKitC V4* que são as seguintes:
 - i) Minicomputador *Raspberry pi*
 - (1) Uma opção boa seria usar um *raspberry pi*, é um SBC (*Single Board Computer*) o que significa ser muito compacto, todo o *raspberry pi* está em um só módulo, teria muito mais memória e poder de processamento, mas os I/O não seriam suficientes (14 pins usáveis) para o projeto [9].
 - ii) Placa de desenvolvimento Arduino Mega
 - (1) O Arduino Mega tem muitos I/O (54 I/O), mas não tem WIFI incorporado e tem um CPU muito inferior (16MHz e um só core).
- 2) Página WEB
 - a) Por restrição de processamento e memória disponível, a página WEB vai ser desenvolvida em HTML, CSS, JavaScript, AJAX e *WebSockets*.
 - b) É importante que a página web atualize os dados apenas quando houver alterações nos estados dos I/O's do destilador.
- 3) Wi-Fi
 - a) É pretendido que seja criado um AP no ESP32 para acesso direto à página web
 - b) É também pretendido que o ESP32 se possa ligar a uma rede Wi-Fi externa

5.2. Arquitetura Geral

5.2.1. Diagrama de Casos de Uso

O diagrama de casos de uso representado na Figura 27 representa as relações das ações decorridas no sistema. O utilizador controla o modo manual (*Manual Mode*)

fisicamente e o modo automático (*Auto Mode*) da máquina, tanto fisicamente, como por ação na página web. O nível de água (*Water Lvl Management*) é controlado automaticamente no modo automático pelos sensores de nível máximo (*Water Max Sensor*), e nível mínimo (*Water Min Sensor*) e pode ser parcialmente administrado no modo manual. Como o sistema é um sistema *embedded*, este é também manipulado pelos sensores, onde estes têm um papel importante na administração do nível de água e do aquecimento da mesma. O modo de alarme (*Alarm*), é controlado pelo sensor de alarme (*Water Alarm Sensor*) o que implica restrições em todos os modos. A página web (Web Server) inclui o uso de um temporizador (*Timer*).

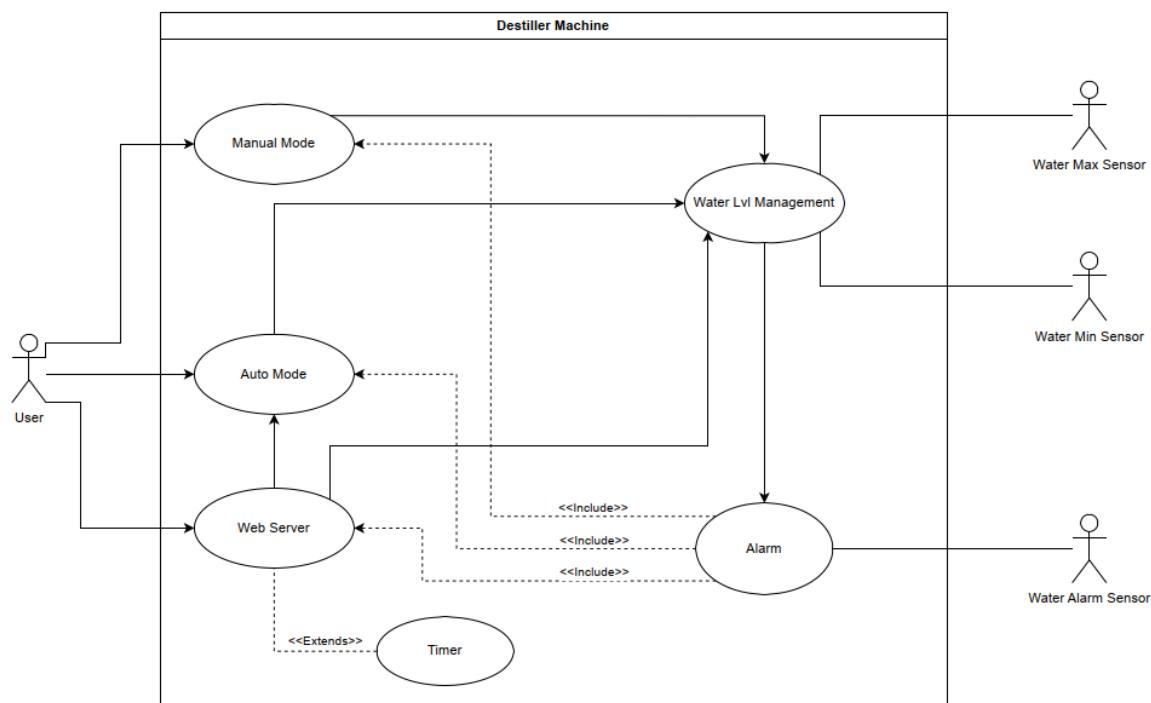


Figura 27 - Diagrama de Casos de Uso

5.2.2. Diagrama de Atividade do Ciclo de Programa

O Diagrama de Atividade do Ciclo de Programa representado na Figura 28, exibe os dois ciclos presentes no *firmware*, em que o primeiro (Loop 1) é responsável por executar o *firmware de controlo da componente física* do destilador (gerir indicadores, modos e a máquina de estados), e o segundo, para correr o servidor web, e o temporizador. Estes surgem por se trabalhar com um processador *multi-core*. A grande vantagem de usar dois *cores* é que podemos executar duas tarefas simultaneamente. A

gestão de ambos os *cores* do ESP32 é feita internamente pelo sistema operativo “*FreeRTOS*” em que só é necessário iniciar o processo (“*xPortGetCoreID()*”) e ser indicado qual é o segundo *loop*. O “*FreeRTOS*” é especialmente desenvolvido para microcontroladores, porque estes têm recursos muito limitados. “*FreeRTOS*” é um *kernel opensource*, isto significa que pode ser adquirida gratuitamente [10].

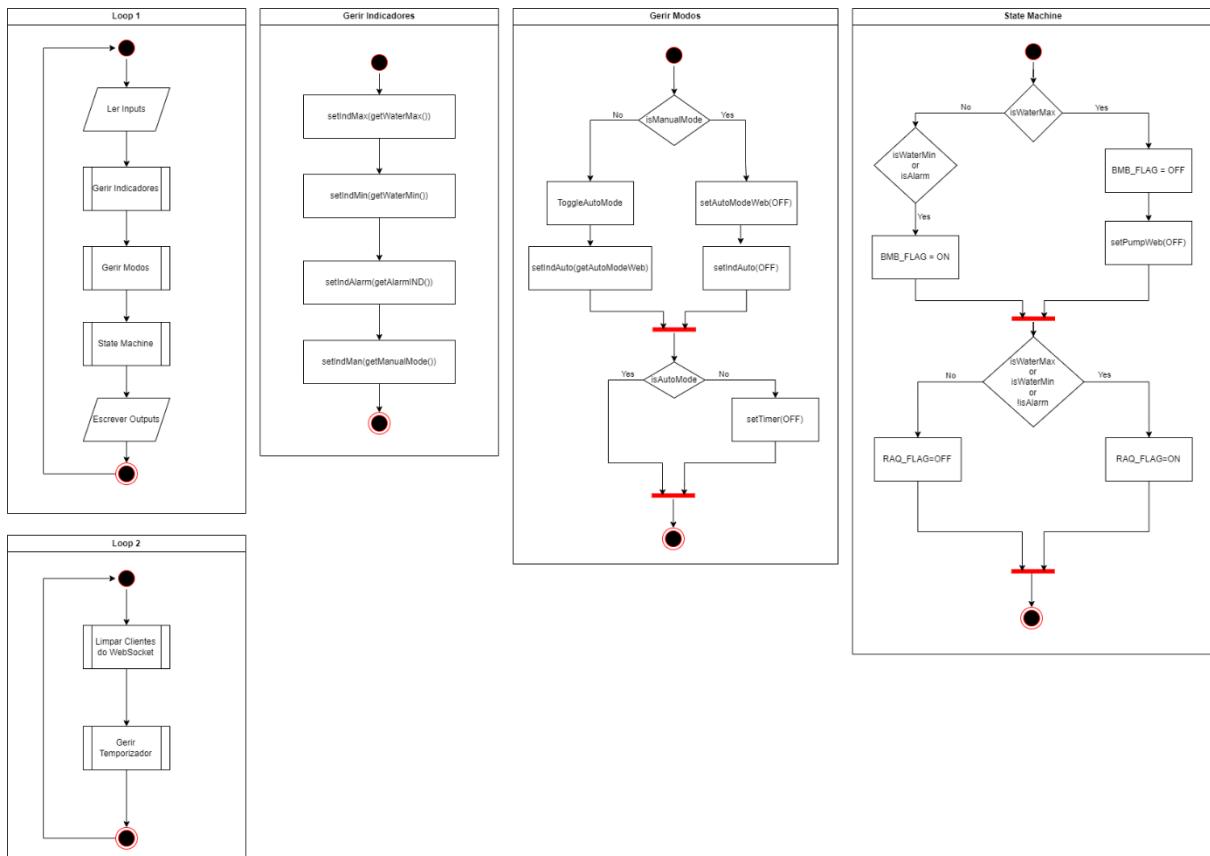


Figura 28 - Diagrama de Atividade do Ciclo de Programa

O primeiro ciclo de programa (*loop 1*) implementa parte do sistema de controlo do destilador e é responsável pela gestão dos indicadores, modos de operação, e controlo dos principais atuadores da máquina. Este é descrito da seguinte forma:

1. Gestão de Indicadores e Modos de Operação:

- As funções *indicatorsManagement()* e *modeManagement()* são chamadas para gerir, respetivamente, os indicadores da máquina e os modos de operação (Automático e Manual).

2. Máquina de estados

A máquina de estados está dividida nos três seguintes blocos

- **Bloco de Trabalho Principal:**

Este bloco, controla a bomba de água e a resistência de aquecimento com base nos níveis de água e no estado de alarme:

- **Bomba de Água (BMB_FLAG):** Desliga-se quando o nível da água está no máximo e liga-se quando o nível está no mínimo ao nível de alarme.
- **Resistência de Aquecimento (RAQ_FLAG):** Liga-se quando o nível da água está dentro dos limites de água.

- **Lógica de Controlo dos Atuadores:**

Este bloco é implementado na forma de máquina de estados, sendo fundamental para determinar o estado dos atuadores com base nas condições de operação, e no modo selecionado:

```
/// If the water pump is on and the machine is in auto mode or the water pump is on in manual mode and the water level is not at the maximum, turn on the water pump
bool BMB = (BMB_FLAG && getIndAuto() || getPumpWeb() && getManualMode() && !getWaterMax());

/// If the water heater resistor is on and the machine is in auto mode or the water heater resistor is on in manual mode and the alarm is not on, turn on the water heater resistor
bool RAQ = (RAQ_FLAG && getIndAuto() && getTimerStatus() || getResistorWeb() && getManualMode() && !getAlarm());

/// If the water inlet valve is on and the machine is in auto mode or the water inlet valve is on in manual mode and the alarm is not on, turn on the water inlet valve
bool V_IN = (RAQ_FLAG && getIndAuto() && getTimerStatus() || getValv_Water_InWeb() && getManualMode() && !getAlarm());

/// If the water outlet valve is on and the machine is in auto mode or the water outlet valve is on in manual mode and the alarm is not on, turn on the vapor outlet valve
bool V_OUT = (RAQ_FLAG && getIndAuto() && getTimerStatus() || getValv_Water_OutWeb() && getManualMode() && !getAlarm());
```

- **BMB:** Controla a bomba de água com base no modo de operação (automático ou manual) e o nível de água.
- **RAQ:** Controla a resistência de aquecimento, ligando-a quando o modo de operação e condições do sistema permitem.

- **V_IN e V_OUT:** Controlam as válvulas de entrada de água e saída de vapor, ativando-as quando os modos de operação e alarmes permitem, garantindo uma operação segura e eficiente.

- **Ajuste dos Estados dos Atuadores:**

- As funções `setPump(BMB)`, `setResistor(RAQ)`, `setValveWaterIn(V_IN)`, e `setValveWaterOut(V_OUT)` são chamadas para definir os estados dos respetivos componentes com base nos valores calculados no ponto anterior (3 - Lógica de Controlo dos Atuadores).

5.2.3. Modelo de Camadas do Firmware

Para que as chamadas sejam únicas, exatas e retornáveis para o ciclo de máquina, e o código seja de mais fácil manutenção, foi criada a arquitetura representada na Figura 29.

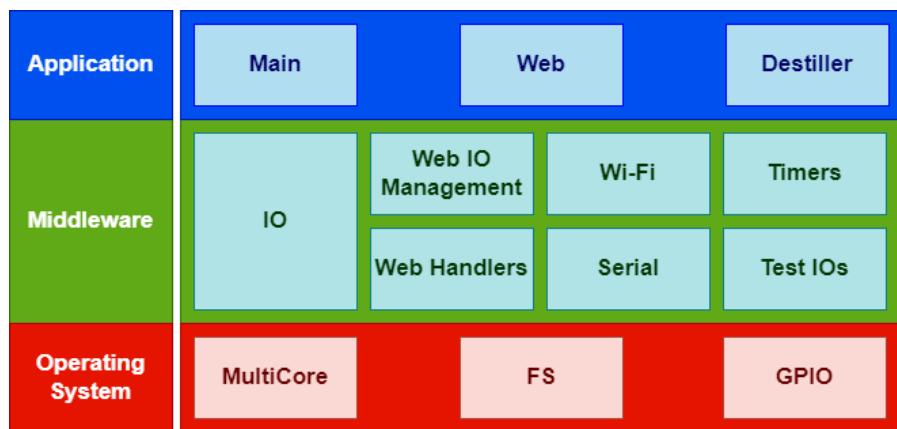


Figura 29 - Esquema Representativo das Camadas do Firmware

Esta é composta por 3 camadas, conforme se descreve de seguida.

a) Camada OS (Operating System)

- Aplicações que atuam no controle do funcionamento dos componentes físicos e universais do sistema.

- Inclui o *header* OS.h que contém as definições de constantes globais, exteriorizações de variáveis globais e disponibilização das funções públicas necessárias para o funcionamento do sistema.
 - **MultiCore** (OS multi core) – Classe responsável pelo início e configuração do segundo core do ESP32.
 - **FS** (OS FS) – Classe inicia o sistema de ficheiros (FS) SPIFFS para armazenamento de ficheiros.
 - **GPIO** (OS GPIO) – Classe responsável pela inicialização dos GPIO's do ESP32, é também aqui definido os *get* e *sets* para os IO's do sistema.

b) Camada MD (Middleware)

- Esta é a camada responsável por invocar todas as classes existentes na camada MD.
- Inclui o *header* MD.h que contém as definições de constantes globais, exteriorizações de variáveis globais e disponibilização das funções públicas necessárias para o funcionamento do sistema.
 - **IO** (MD IO) - Módulo responsável pela gestão dos indicadores, modo e trocar o estado do modo automático dependendo do estado atual.
 - **Web IO Management** (MD Web IO Mngmnt) – Esta módulo contém os *get* e *sets* dos IO's em semelhança à classe "OS_GPIO" mas relativo ao servidor web.
 - **Web Handlers** (MD Web Handlers) – É neste módulo que estão os *handlers* para os pedidos do servidor web.
 - **Wi-Fi** (MD wifi) – Módulo responsável pela inicialização e administração do Wi-Fi, é aqui que é configurado a ligação a uma rede Wi-Fi e/ou a criação de um *softAP* para conexão direta ao servidor web sem ser necessário ligar a uma rede Wi-Fi externa.

- **Serial** (MD Serial) – É neste módulo que é iniciado e configurado a comunicação com a porta de série para poder escrever no terminal.
- **Timers** (MD Timer) – Módulo responsável pela gestão do temporizador da página web.
- **Test IOs** (MD Test Ios) – Módulo dedicado para testar os *outputs* do sistema.

c) Camada AP (Application)

- Esta camada inclui todos os módulos que invocam as funções de aplicação do sistema.
- Inclui o *header* AP.h que contém as definições de constantes globais, exteriorizações de variáveis globais e disponibilização das funções públicas necessárias para o funcionamento do sistema.
 - **Main** (AP main) – É neste módulo inicial do projeto, contem a função “setup()” que é executada apenas uma vez na inicialização do sistema e os dois *loops* correspondentes aos dois cores do microprocessador.
 - **Destiller** (AP Destiler) – Módulo responsável pela máquina de estados do sistema.
 - **Web** (AP Web) – Módulo que define as rotas para as solicitações *HTTP GET* para vários recursos, como páginas HTML, arquivos CSS, arquivos JS, imagens, etc.

5.3. - Modelo de Software

Para garantir uma organização eficiente e facilitar a manutenção do código do projeto, foi adotada uma estrutura de ficheiros bem definida, distribuída em diretórios específicos para diferentes finalidades. Esta estrutura inclui pastas como "data", que contém os ficheiros relacionados com a página web, "Documentation", para toda a documentação associada ao projeto, "include" e "lib", que contêm os ficheiros de

cabeçalho e bibliotecas externas, respetivamente, e "src", que reúne os ficheiros de código em .CPP. Além disso, na pasta raiz, encontram-se ficheiros de configuração importantes para a definição de partições de memória e para a utilização da extensão *PlatformIO*.

5.3.1. Modelo de ficheiros

Para melhor organização e futura manutenção dos ficheiros, foram cuidadosamente distribuídos em pastas para que possa ser possível a sua localização. Na pasta raiz estão os diretórios referentes a "data", "Documentation", "include", "lib" e "src".

- *data* - Tem tudo o que é relacionado com a página de internet, tal como os ficheiros HTML, CSS, JS e PNG, este diretório é um requisito para a utilização do sistema de ficheiros SPIFFS [11].
- *Documentation* – Contém tudo o que está relacionado com a documentação, tal como o relatório, diagramas, etc.
- *Include* – Contém todos os ficheiros *headers* necessários para o funcionamento do *software*.
- *Lib* – Esta pasta, contém as bibliotecas externas necessárias para o *software*.
- *Src* – Contém os ficheiros '.cpp'
- Na pasta raiz, estão ainda 2 ficheiros necessários para as várias configurações descritas abaixo:
 - "partitions.csv", que contém a configuração das partições da memória do ESP32.
 - "platformio.ini", contém a configuração da extensão PlatformIO.

5.3.2. Dependências

O projeto também faz uso de várias dependências externas essenciais, como a biblioteca *ESPAsyncWebServer*, que gera o servidor web assíncrono e os *WebSockets*, a SPIFFS, que permite o uso de um sistema de ficheiros eficiente no ESP32, e a

WiFiManager, que facilita a gestão de conexões Wi-Fi e a criação de um ponto de acesso (AP) para o utilizador se conectar diretamente ao microcontrolador. As dependências são as seguintes:

- **ESPAsyncWebServer** - Esta biblioteca, é a responsável pela criação e gestão do servidor web assíncrono e dos *websockets* [12].
- **SPIFFS** - Com esta biblioteca, é possível usar o sistema de ficheiros SPIFFS para armazenar os ficheiros necessários para a web [6].
- **WiFiManager** - Esta biblioteca é responsável pela ligação a uma rede Wi-Fi, é também possível a criação de um AP para o cliente se poder ligar diretamente ao ESP32 sem ser necessária uma rede Wi-Fi externa [13].
- **Esp_netif** - Esta biblioteca é uma *interface* de rede do ESP32, é usada para configurar o IP, *gateway* e máscara para a rede Wi-Fi [14].
- **ESPmDNS** - Com esta biblioteca, é possível uma implementação simples de suporte a consultas DNS *multicast* para o ESP32, é usada para dar um *hostname* ao IP do ESP32 [15].

5.4. Ligar a Uma Rede Wi-Fi Externa Nova

Com a biblioteca acima mencionada (*WiFiManager*), se não for possível ligar à rede anteriormente configurada, é possível configurar uma nova rede Wi-Fi externa, seguindo os seguintes passos, como pode ser visto no Anexo 2:

- Encontrar a rede Wi-Fi gerada pelo ESP32 com o nome “*ConfDestiler AP*”, não é necessário inserir *password* (Figura 30a).
- Depois de entrar na rede “*Conf Destiler AP*”, abrir um *browser* e inserir o IP “192.168.4.1” que irá abrir a página demonstrada na Figura 30b e escolher “Configure WiFi” para seguir ao próximo passo,
- Como é mostrado na Figura 30c, é mostrada uma lista de redes Wi-Fi disponíveis e clicamos na rede pretendida e inserimos a *password* que com a ligação bem-sucedida, vai mostrar o ecrã na Figura 30d e o ESP32 vai reiniciar automaticamente e ligar-se à automaticamente à nova rede.

- O *Hostname* do destilador na rede externa é “<http://Destiler.local/>”.
- O utilizador tem um minuto para configurar a nova rede, caso contrário, a configuração é cancelada e é gerado o “*SoftAP*” descrito a seguir.
- É possível apagar as credenciais armazenadas no ESP32, para isso, é necessário ter acesso físico ao ESP32 e pressionar por três segundos o botão incorporado no microcontrolador com a designação “*BOOT*” (ver Figura 31), as credenciais vão ser apagadas e o ESP32 irá reiniciar automaticamente.

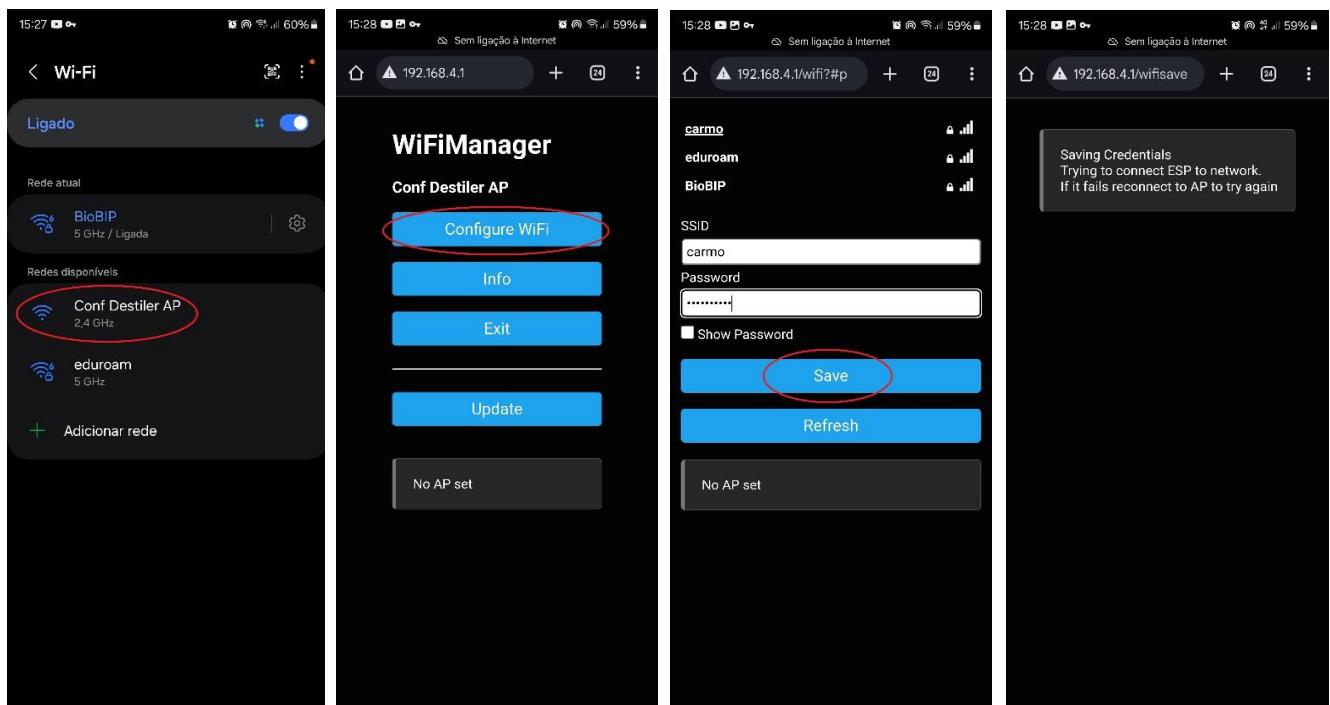


Figura 30 – Ligação à Rede Externa



Figura 31 - Botão de Reset ao Wi-Fi

5.5. Ligar ao AP do Destilador (SoftAP)

O destilador permite também, como pode ser visto no Anexo 2, que o utilizador se ligue diretamente ao destilador sem ser necessário ligar a uma rede Wi-Fi externa, para isso, o utilizador deve procurar no seu dispositivo pela rede Wi-Fi "Destiller-AP" e password "Destiller-AP", depois da ligação efetuada, em um browser deve ser inserido o IP predefinido "192.168.1.100". O Hostname do SoftAP é "<http://DestilerAP.local/>".

5.6. Página web

No presente projeto, o servidor web é implementado no ESP32 utilizando tecnologias como HTML, CSS, JavaScript, AJAX e *WebSockets* para fornecer uma interface de utilização interativa e responsiva tendo em conta as limitações do microcontrolador. Por isso, foi decidido ser criadas duas páginas web com as seguintes características:

- (i) A página web representada na Figura 32 não é possível de ser responsiva pela sobreposição de imagens e pelo tamanho da imagem do esquema de processos. No entanto, é mais intuitiva, já que possui a imagem do esquema de processos e a imagem de cada atuador para ser possível clicar no atuador e alterar o seu estado. Tem, no entanto, o problema de, se a conexão Wi-Fi entre o destilador e o dispositivo com a página web for fraca, demora mais a carregar a página web.
- (ii) A página web representada na Figura 33, é responsiva. Os atuadores tomam forma de botões e é mais rápida e estável a carregar a página se a ligação Wi-Fi for fraca, situação usual tendo em conta o fraco desempenho da antena do módulo ESP32 (antena integrada na PCB).

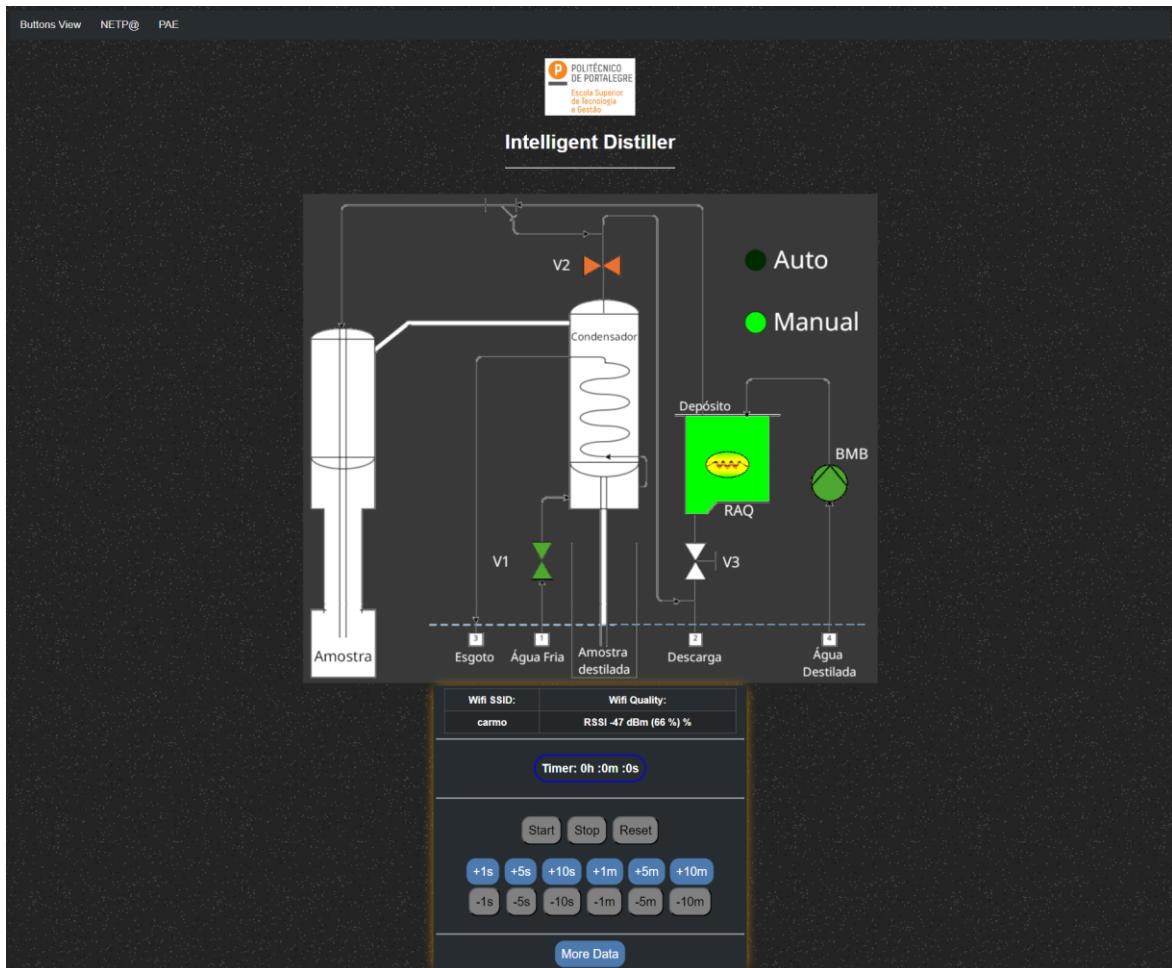


Figura 32 - Página Web com Imagens

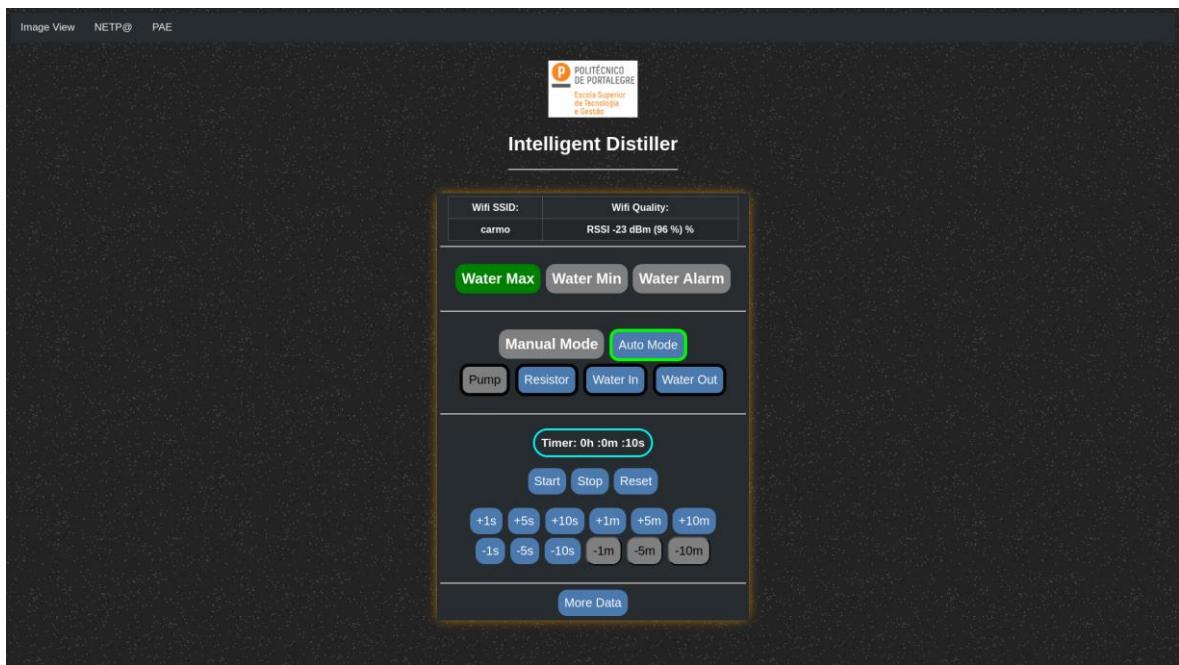


Figura 33 - Página Web com Botões

O HTML e o CSS são usados para estruturar e estilizar a página web que controla e monitoriza o destilador, garantindo uma apresentação visual clara e organizada [16]. O JavaScript é utilizado para adicionar interatividade e dinamismo à página, permitindo a manipulação de elementos e o envio de pedidos assíncronos ao servidor [17] [18].

AJAX (*Asynchronous JavaScript and XML*) desempenha um papel essencial na comunicação assíncrona com o servidor web. Ele permite que os dados sejam enviados e recebidos sem a necessidade de atualizar a página inteira, proporcionando uma experiência de utilização mais fluida e rápida [19]. Isto é especialmente útil no projeto para a atualização em tempo real dos parâmetros e estados do destilador, como I/O's e estado dos modos do sistema.

Os *WebSockets*, por sua vez, são usados para uma comunicação bidirecional contínua entre o cliente (a página web) e o servidor ESP32. Ao contrário do AJAX, que funciona com pedidos pontuais. Os *WebSockets* permitem que o servidor envie atualizações automáticas para o cliente sempre que houver alterações nos parâmetros em questão [20]. Isto é crucial para garantir que os dados exibidos na página web estejam sempre atualizados sem a necessidade de múltiplos pedidos ao servidor [21]. Assim, a combinação destas tecnologias oferece uma interface eficiente para o controlo e monitorização do destilador, garantindo interatividade, atualizações em tempo real, e uma experiência de utilizador mais otimizada [22]. (ver Figura 32 e Figura 33).

5.7. Modelo de Funcionamento

Para melhor percepção do funcionamento do destilador ao nível do utilizador, foi elaborado o diagrama de funcionamento representado na Figura 34.

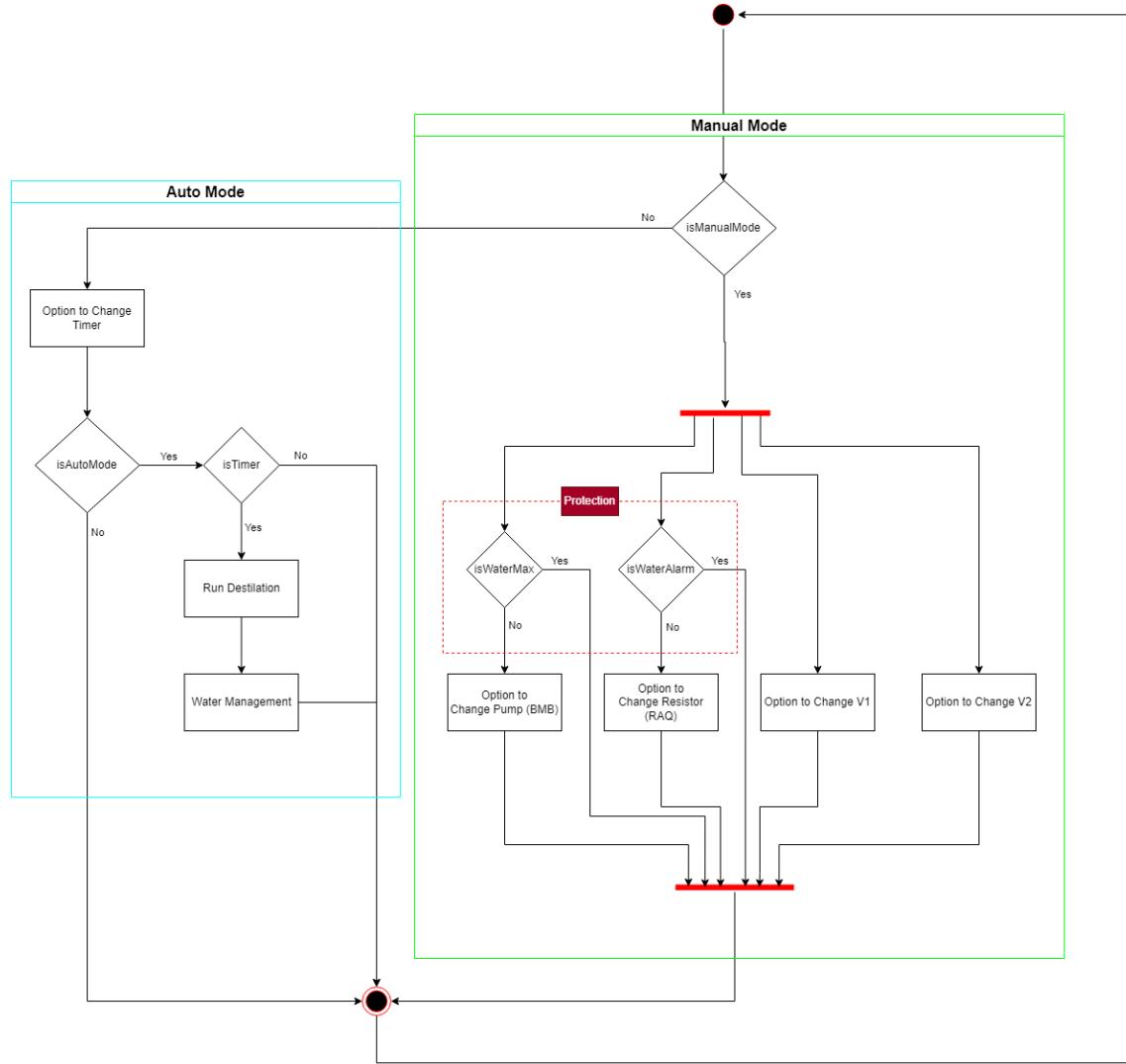


Figura 34 – Diagrama de Sequência do funcionamento de Utilizador

Portanto, como é verificado na Figura 35, se o modo manual estiver ativo, o utilizador pode através da página web ativar/desativar a válvula de entrada de água fria (V1), a válvula de saída de vapor (V2), a bomba de água (BMB) se o nível de água não estiver no nível máximo e a resistência de aquecimento (RAQ) se a água não estiver no nível de alarme.

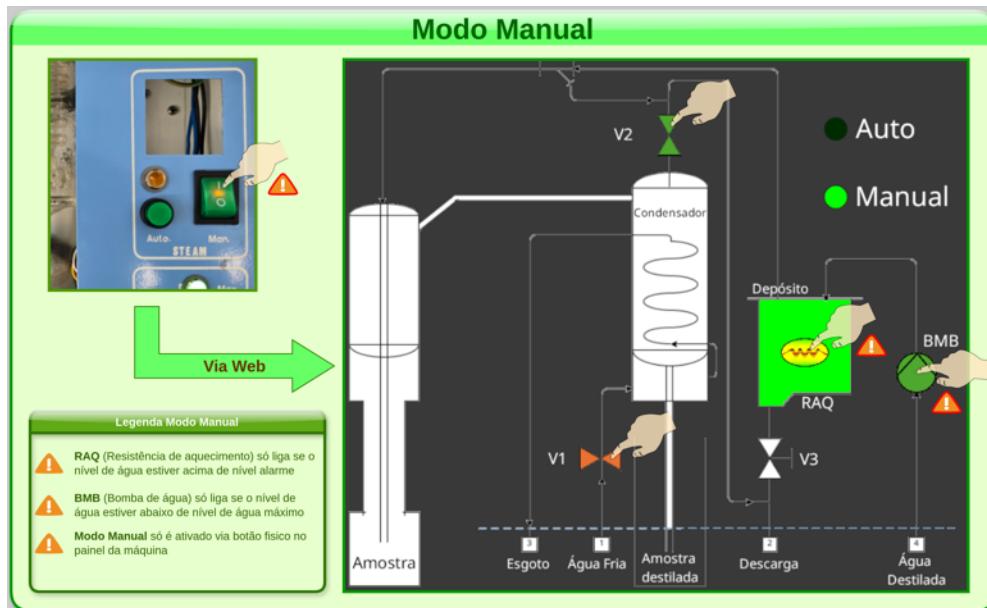


Figura 35 - Uso em Modo Manual

Por sua vez, se o modo manual estiver desativado, é possível ativar o modo automático, em que se o modo automático estiver ativo, como consta na Figura 36, é possível alterar o temporizador e ativar o mesmo. Ao ativar o temporizador, começa o processo de destilação e o nível de água é gerido automaticamente. A gestão do nível de água consiste em ligar a bomba (BMB) se o nível de água estiver em nível mínimo ou inferior, ao nível de água chegar ao nível máximo, a bomba é desligada. A resistência (RAQ) está sempre em funcionamento se o temporizador estiver ativo e o nível de água estiver acima do nível de alarme. Se chegar ao nível de alarme, a resistência (RAQ) desliga e o processo de destilação é interrompido por falta de água. Ambas as válvulas (V1 e V2) são geridas em paralelo com a resistência (RAQ), já que são igualmente essenciais no processo de destilação à resistência de calor (RAQ).

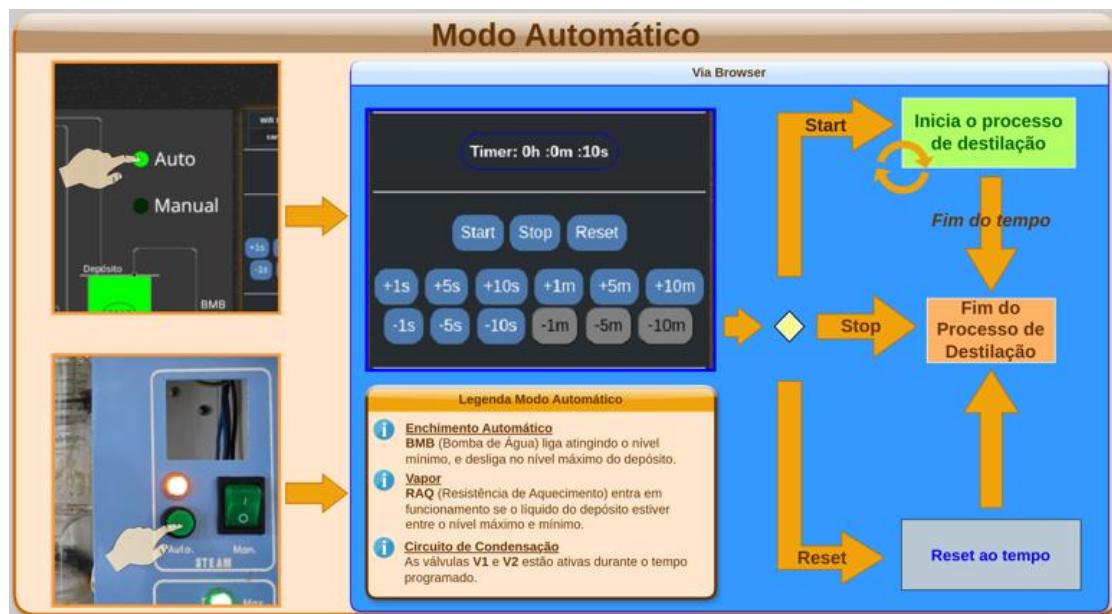
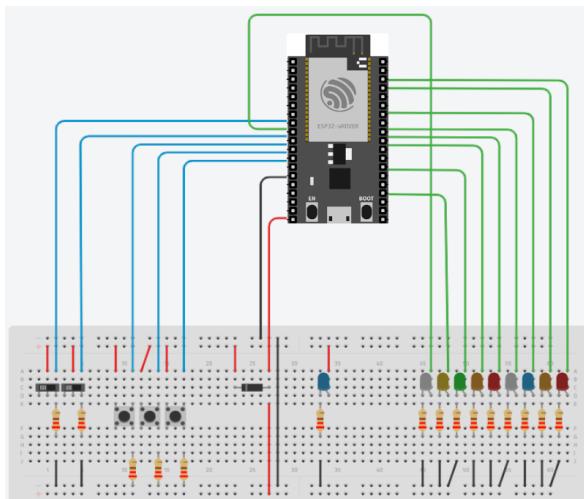


Figura 36 - Uso em Modo Automático

6. TESTES

6.1. Prototipagem em Placa de Ensaio

Para ter um primeiro protótipo e ter uma plataforma independente do destilador para testes, foi simulado todo o destilador numa placa de ensaio com botões, interruptores e indicadores LED para simular os I/Os da máquina real, em que os botões e interruptores, simulam os *inputs* e os indicadores LED os *outputs*.



da rede elétrica, e existem componentes que necessitam de condições ótimas para funcionar, como é o exemplo da bomba de água (BMB) em que é usada a própria água bombeada para arrefecer a própria bomba (BMB), e a resistência (RAQ) que se não tiver água no reservatório vai sobreaquecer e queimar.

7. CONCLUSÃO

O presente projeto revelou-se uma solução eficaz para modernizar e automatizar um equipamento analógico que se encontrava obsoleto e fora de funcionamento. A substituição dos componentes antigos por tecnologias digitais, e o acréscimo de novos componentes como o microprocessador ESP32, permitiu não só restaurar o funcionamento do destilador, mas também introduzir novas funcionalidades, como o controlo automático e remoto através de uma interface web.

Os testes realizados comprovaram que o destilador modernizado atende aos requisitos funcionais e não funcionais definidos, operando de forma estável tanto em modo manual como em modo automático. A integração de sensores de nível, relés e outros atuadores com o ESP32 proporcionou um controlo preciso e seguro do processo de destilação, eliminando problemas de desgaste e falhas mecânicas associados ao sistema original.

Além disso, o uso de tecnologias como *AJAX* e *WebSockets* na interface web garantiu uma comunicação eficiente entre o utilizador e o equipamento, facilitando a monitorização e o ajuste dos parâmetros de operação em tempo real. As dificuldades encontradas, como o ruído eletromagnético nas entradas de sinal, foram superadas com soluções de filtragem adequadas, contribuindo para a robustez do sistema.

Em suma, o "Destilador Inteligente" representa uma importante melhoria na eficiência e no controlo dos processos laboratoriais, abrindo possibilidades para futuras expansões e otimizações. Este projeto não só revitaliza um equipamento essencial, mas também demonstra o potencial da integração de sistemas *embedded* e tecnologias de automação na modernização de equipamentos tradicionais.

REFERÊNCIAS BIBLIOGRÁFICAS

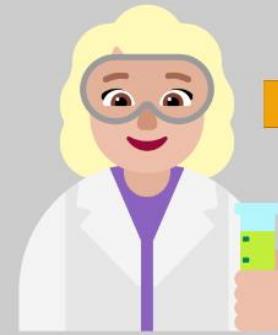
- [1] Espressif, “Espressif-About,” [Online]. Available: <https://www.espressif.com/en/company/about-espressif>. [Acedido em 08 2024].
- [2] Microsoft, “Visual Studio Code,” [Online]. Available: <https://code.visualstudio.com/>. [Acedido em 08 2024].
- [3] P. Labs, “PlatformIO,” [Online]. Available: <https://platformio.org/>. [Acedido em 08 2024].
- [4] espressif, “espressif ESP32,” [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wrover-e_esp32-wrover-ie_datasheet_en.pdf. [Acedido em 08 2024].
- [5] L. Harvie, “Embedded Systems Memory Types: Flash vs SRAM vs EEPROM,” [Online]. Available: <https://medium.com/@lanceharvieruntime/embedded-systems-memory-types-flash-vs-sram-vs-eeprom-93d0eed09086>. [Acedido em 08 2024].
- [6] espressif, “SPIFFS Filesystem,” [Online]. Available: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/spiffs.html>. [Acedido em 08 2024].
- [7] V. Muthukrishnan, “electrical4u,” 27 05 2024. [Online]. Available: <https://www.electrical4u.com/electromagnetic-interference/>. [Acedido em 08 2024].
- [8] E. Schematics, “Electro Schematics,” [Online]. Available: <https://www.electroschematics.com/optical-liquid-level-sensor/>. [Acedido em 23 08 2024].
- [9] R. P. (. Ltd., “Raspberry Pi Datasheet,” [Online]. Available: <https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf>. [Acedido em 08 2024].
- [10] microcontrollerslab.com, “ESP32 Dual Core with FreeRTOS and Arduino IDE,” [Online]. Available: <https://microcontrollerslab.com/esp32-dual-core-freertos-arduino-ide/>. [Acedido em 08 2024].
- [11] “ESP32 with VS Code and PlatformIO: Upload Files to Filesystem (SPIFFS),” [Online]. Available: <https://randomnerdtutorials.com/esp32-vs-code-platformio-spiffs/>. [Acedido em 08 2024].
- [12] M. N. Dev, “ESPAsyncWebServer,” GitHub, [Online]. Available: <https://github.com/me-no-dev/ESPAsyncWebServer>.
- [13] tzapu, “WiFiManager,” GitHub, [Online]. Available: <https://github.com/tzapu/WiFiManager>. [Acedido em 08 2024].

- [14] espressif, “GitHub ESP-NETIF,” [Online]. Available: <https://github.com/espressif/esp-idf/tree/master>. [Acedido em 09 2024].
- [15] espressif, “GitHub ESPmDNS,” [Online]. Available: <https://github.com/espressif/arduino-esp32/blob/master/libraries/ESPM DNS/src/ESPM DNS.h>. [Acedido em 09 2024].
- [16] W3C, “HTML & CSS Standards.,” *World Wide Web Consortium (W3C)*, 2023.
- [17] J. Robbins, *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics.*, O'Reilly Media, 2018.
- [18] D. Flanagan, *JavaScript: The Definitive Guide.*, O'Reilly Media, 2020.
- [19] J. J. Garrett, *Ajax: A New Approach to Web Applications*, Adaptive Path, 2005.
- [20] A. Pérez, *ESP32 Development Using Web Technologies.*, Packt Publishing, 2022.
- [21] B. a. A. B. Lubbers, *WebSockets: A Modern Web Standard for Real-Time Communication*, Apress, 2021.
- [22] S. Monk, *Programming the ESP32: Developing IoT Projects with Wi-Fi, Bluetooth, and Low-Power Devices*, McGraw-Hill, 2021.
- [23] M. Nardi, “Elimine INTERFERÊNCIAS dos seu projetos feitos com ARDUINO!,” YouTube, 2022. [Online]. Available: <https://www.youtube.com/watch?v=uyltlebdCg>. [Acedido em 08 2024].
- [24] A. Ahmad, “Calculating RC Low-Pass Filter Cut-Off Frequency and Transfer Function,” 17 07 2023. [Online]. Available: <https://eepower.com/technical-articles/calculating-rc-low-pass-filter-cut-off-frequency-and-transfer-function/>. [Acedido em 08 2024].
- [25] E. Wings, “Digital GPIO of Arduino,” [Online]. Available: <https://www.electronicwings.com/arduino/digital-gpio-of-arduino>. [Acedido em 08 2024].
- [26] Microcontrollerslab, “ESP32 Dual Core with FreeRTOS and Arduino IDE,” [Online]. Available: <https://microcontrollerslab.com/esp32-dual-core-freertos-arduino-ide/>. [Acedido em 08 2024].

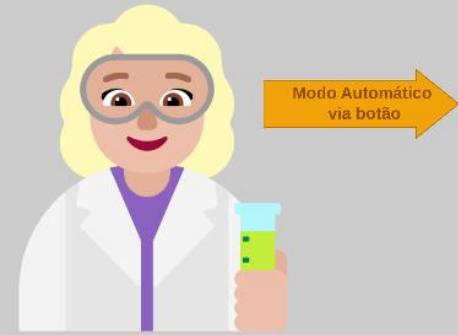
ANEXOS

Anexo 1 – Instruções de Uso Geral

Instruções de uso



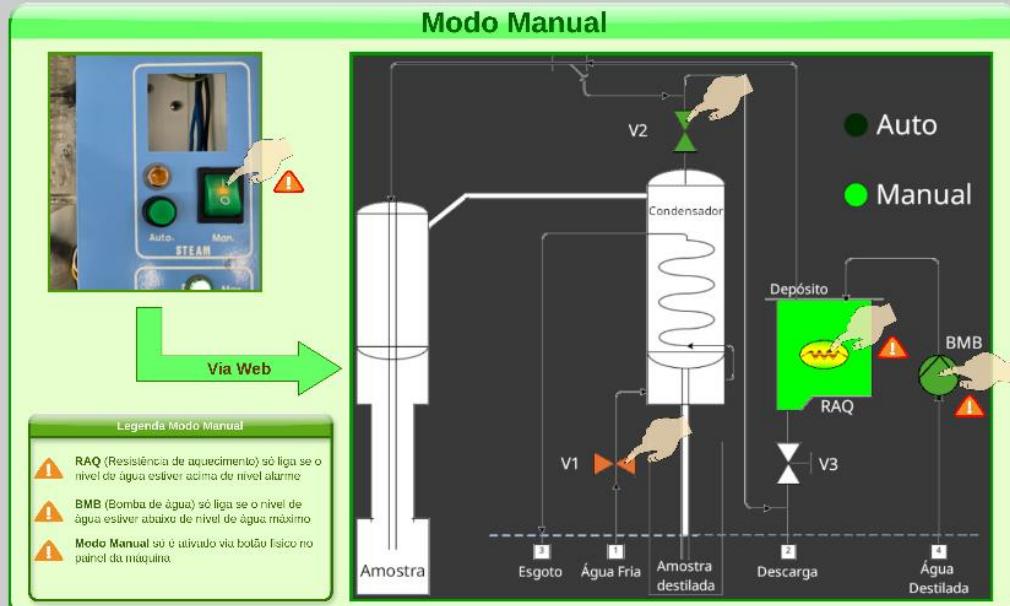
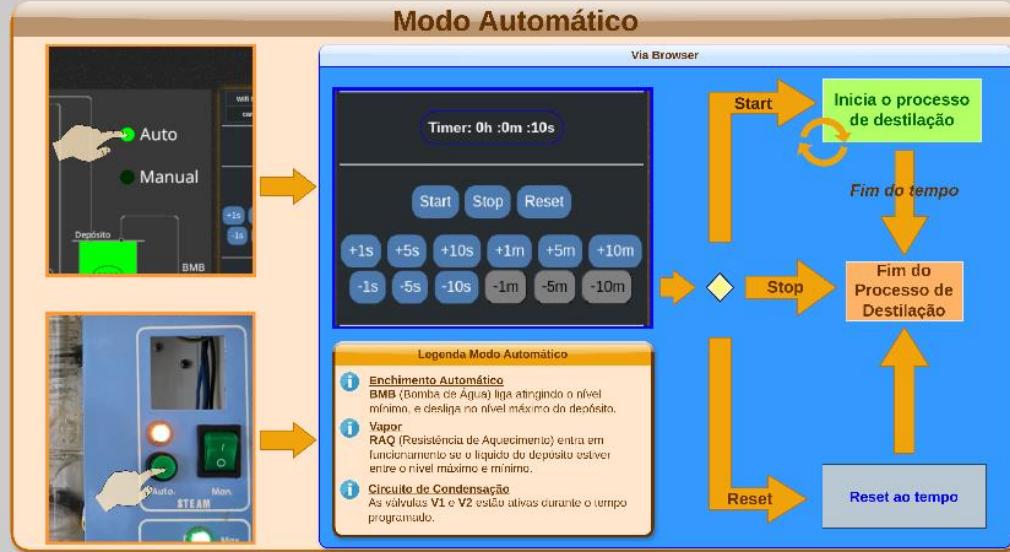
Modo Automático via web



Modo Automático via botão



Modo Manual via botão



Anexo 2 – Instruções de Uso do Wi-Fi

Instruções de Uso do Wi-Fi

Configurar uma Rede Local

Será necessário saber o IP atribuído ao equipamento. Contactar o administrador da rede



Ligar Diretamente ao Equipamento

Se após 1 minuto não for configurada uma rede WiFi, o equipamento cria a sua própria rede (modo AP). Um telemóvel ou PC pode ligar-se a esta rede.

Configurar uma Rede Wi-Fi Local

Escolher a Rede Wi-Fi "Conf Destilar AP"

Inserir no Browser o IP "192.168.4.1" e Escolher a Opção "Configure WiFi"

Selecionar a Rede Pretendida, Inserir a Password e Clicar "Save"

Ecrã Final e o Destilador vai Reiniciar e Ligar à Rede Pretendida.

Ligar ao AP Interno

Escolher a Rede Wi-Fi "Destiler-AP" com a Password "Destiler-AP"

Inserir o IP "192.168.100.100" no Browser do telemóvel, PC ou tablet.

Apagar a Rede Wi-Fi Guardada

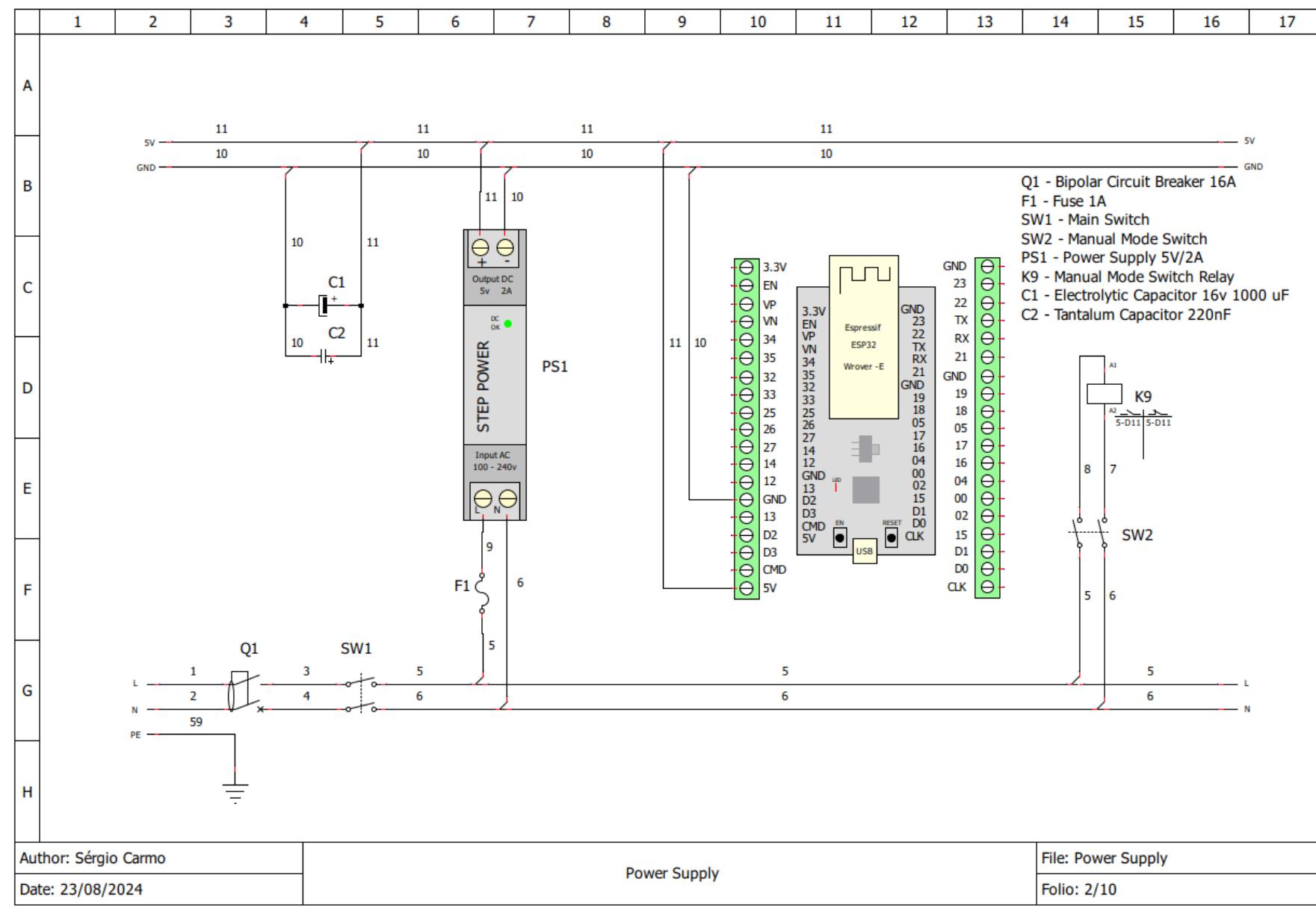
Pressionar por 3 Segundos o Botão "BOOT" para Apagar a Rede Wi-Fi Guardada

Anexo 3 – Esquema Elétrico

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A																	
B	 Instituto Politécnico Portalegre																
C																	
D																	
E																	
F																	
G																	
H																	
Author: Sérgio Carmo				Cover								File: Cover					
Date: 23/08/2024												Folio: 1/10					

Intelligent Destiller

DOCENTE: Sérgio Correia
DISCENTES: Sérgio Carmo
 N.º 19749

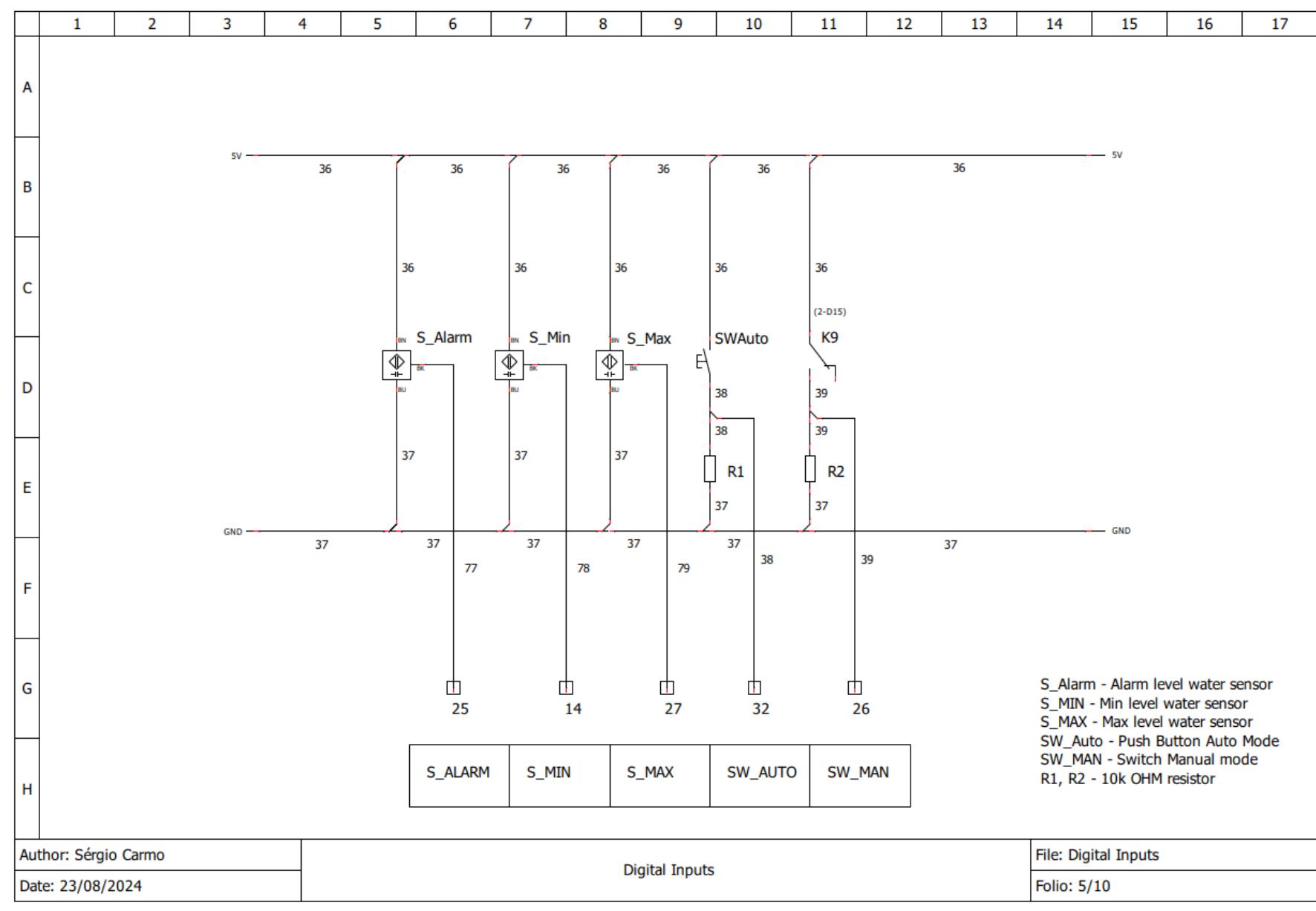


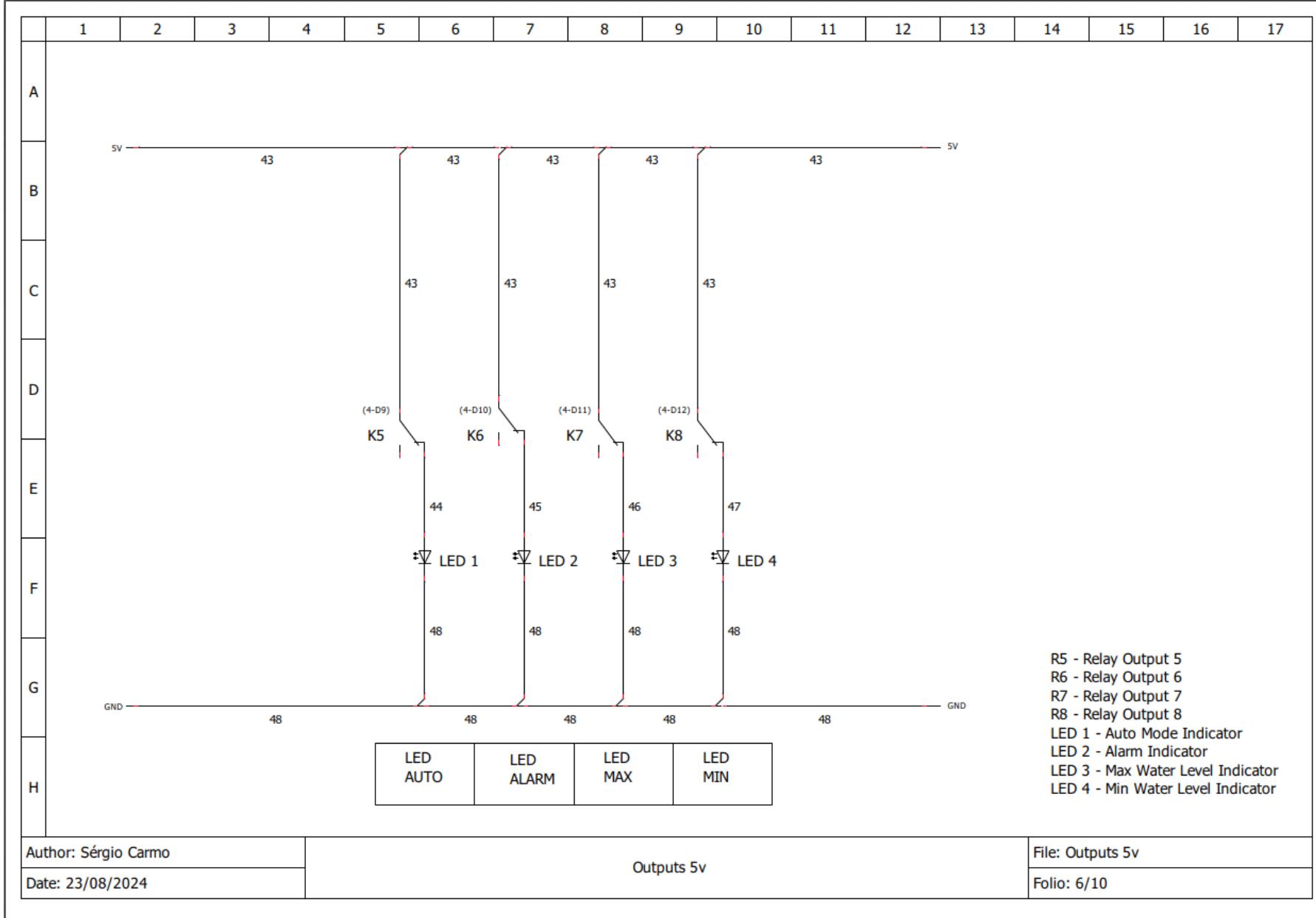
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A																	
B																	
C																	
D																	
E																	
F																	
G																	
H																	

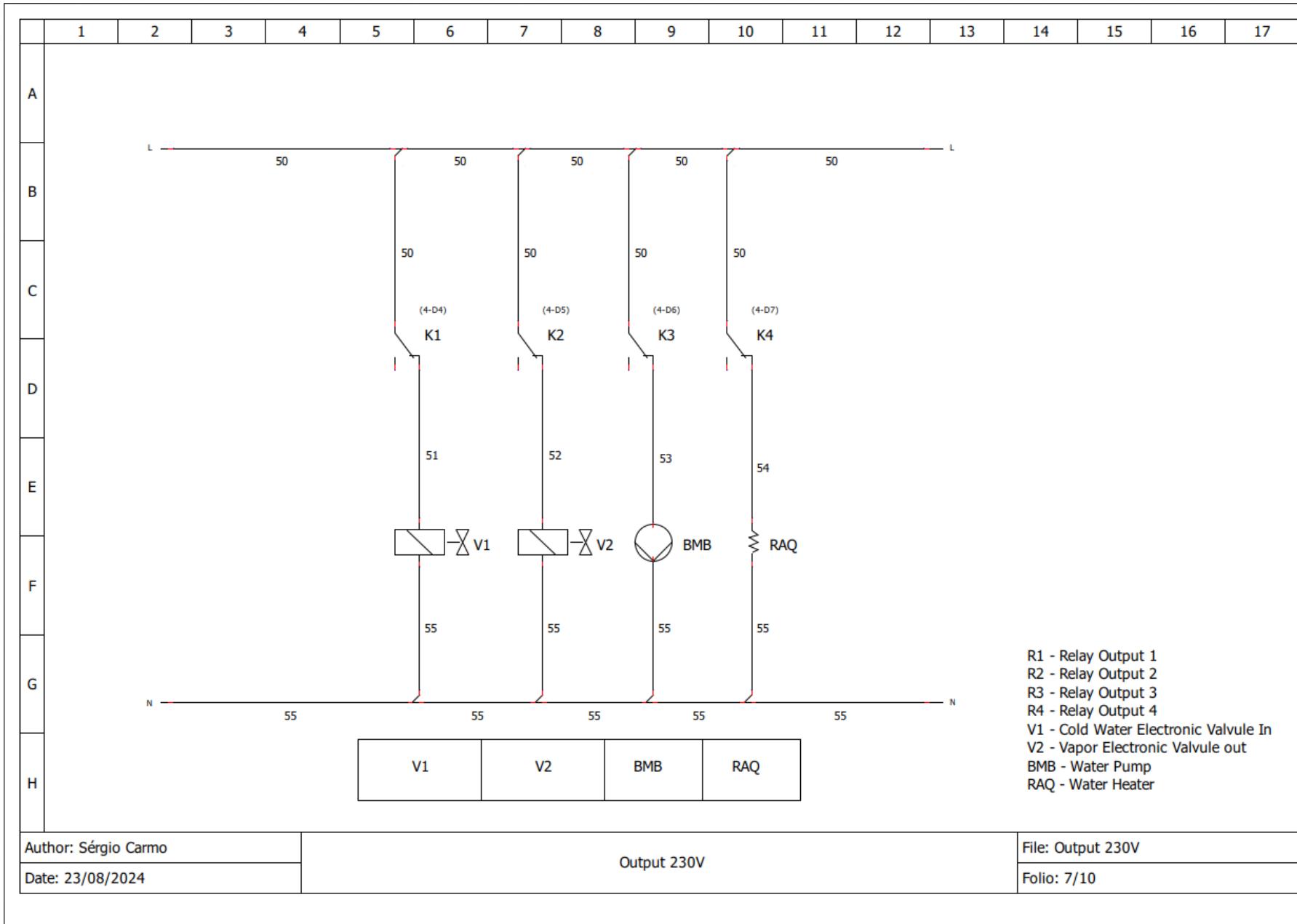
The diagram illustrates a control circuit with the following connections:

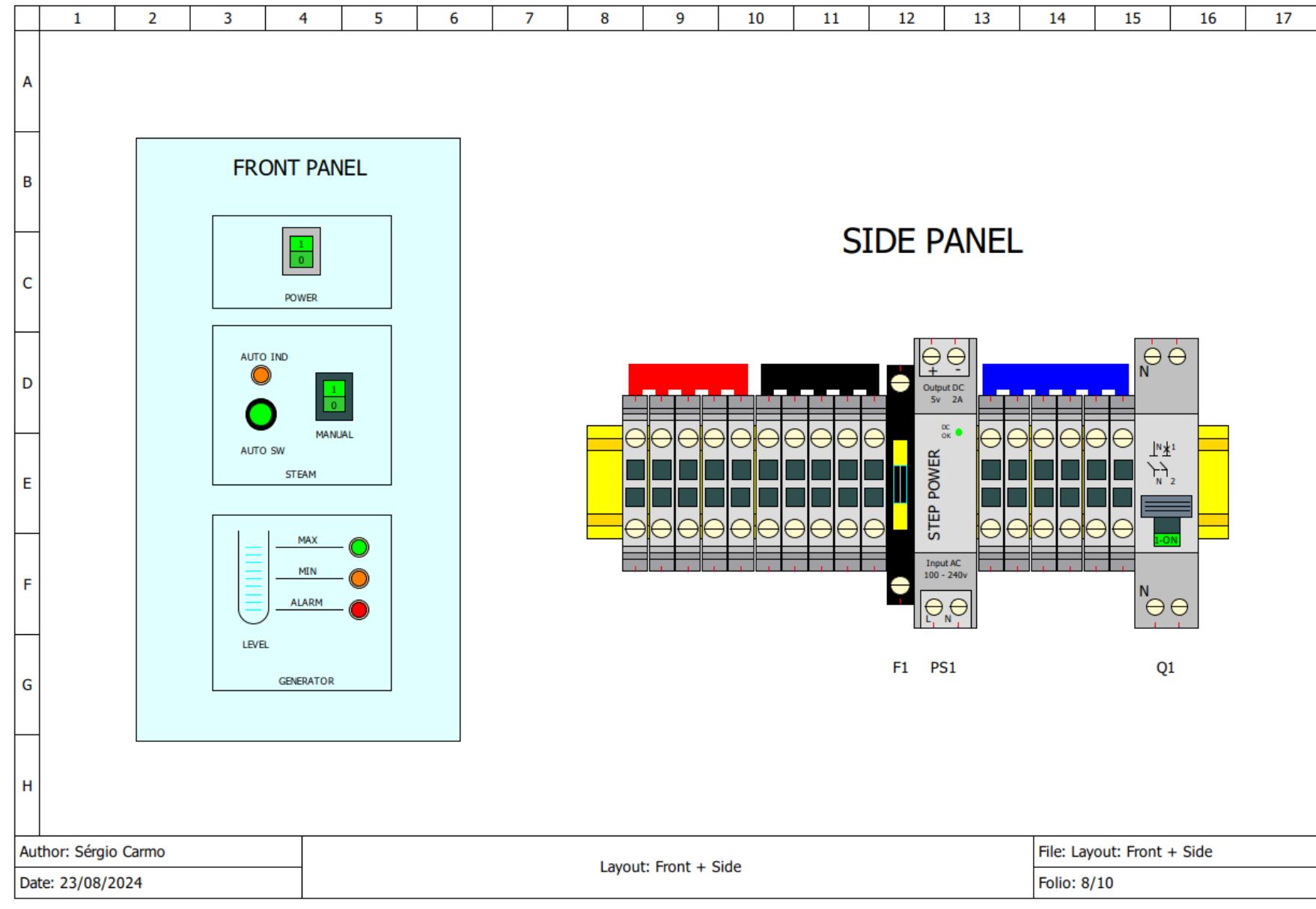
- Pins 19, 21, 22, 23, 15, 18, 04, and 05 are connected to components K1 through K8 respectively.
- Components K1 through K8 are connected to ground via resistors labeled 35 ohms.
- Components K1, K2, K3, and K4 share a common connection point labeled A1.
- Components K5, K6, K7, and K8 share a common connection point labeled A1.
- Components K1, K2, K3, and K4 are connected to pins 27, 28, 29, and 30 respectively.
- Components K5, K6, K7, and K8 are connected to pins 31, 32, 33, and 34 respectively.
- Resistors labeled 7-D5, 7-D7, 7-D8, 7-D9, 6-D5, 6-D7, 6-D8, 6-D9, and 6-D9 are connected between the A1 points and the respective component pins.
- Below the schematic, there are two horizontal boxes. The left box contains four buttons labeled V1, V2, BMB, and RAQ. The right box contains four LED indicators labeled LED AUTO, LED ALARM, LED MAX, and LED MIN.

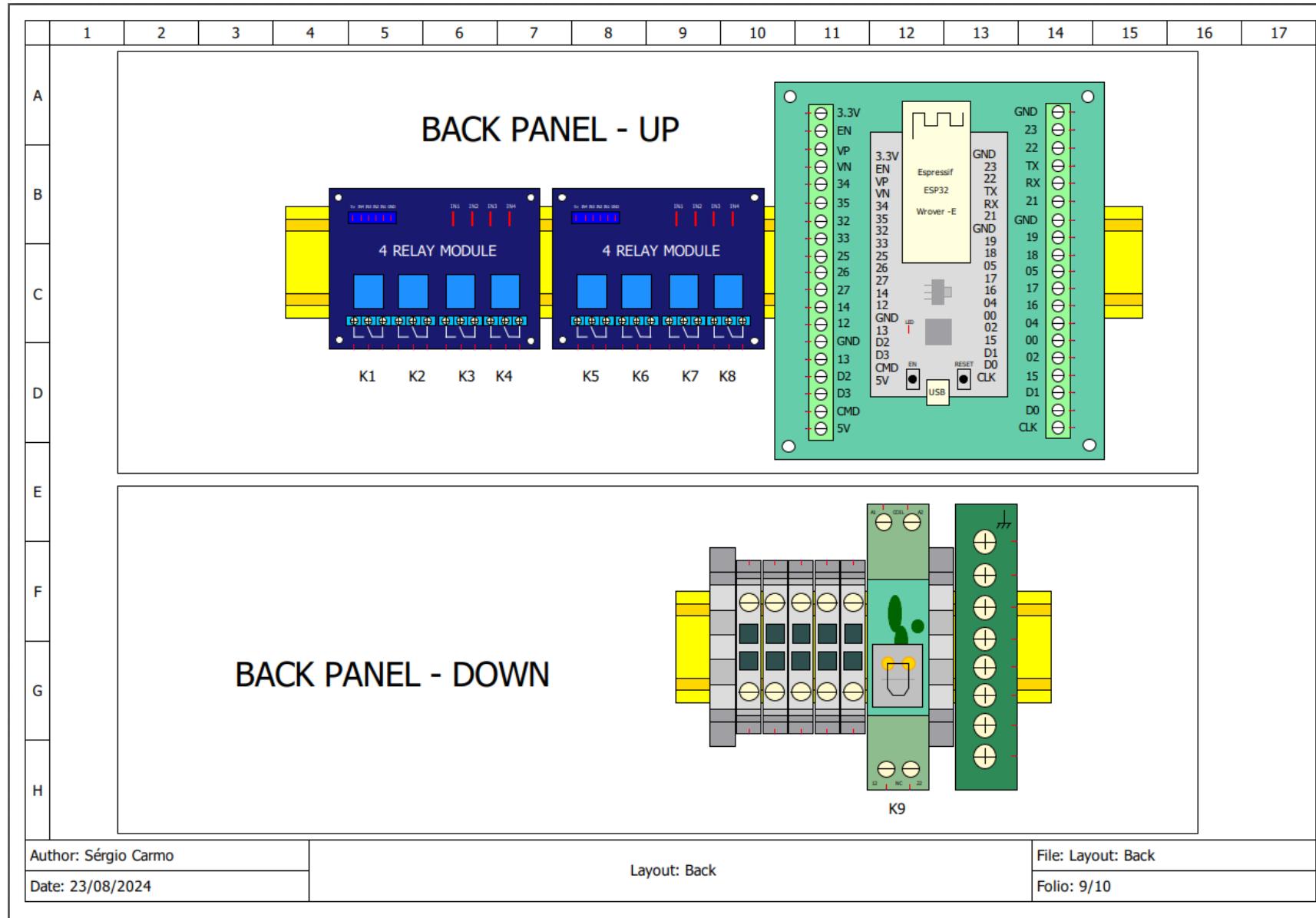
V1 - Cold Water Electronic Valve In
V2 - Vapor Electronic Valve out
BMB - Water Pump
RAQ - Water Heater
LED AUTO - Auto Mode Indicator
LED ALARM - Alarm Indicator
LED MAX - Max Water Level Indicator
LED MIN - Min Water Level Indicator











	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
A																		
B	Quantity	Description				Manufacturer	Ref. Manufacturer	Supplier	Ref. Supplier									
	1	Circuit Breaker 16A				Schneider Eletric	iDPN N C 16A	se.com	A9N21547									
	1	Main Switch With Built-in 240V Neon Indicator																
	1	STEP-PS/ 1AC/ 5DC/2 - Power supply unit				Phoenix Contact	2320513	Mouser.com	651-2320513									
	1	DIN Rail Terminal Blocks UK 5-HESI N (Fuse 1A)				Phoenix Contact	651-3000539	Mouser.com	3000539									
	1	Esp32-Wrover-E				Espressif	ESP32_DevKitc_V4	Mouser.com	356-ESP32WRVE22864PC									
	1	Terminal Adapter for ESP_32 (38-pin)					616											
	2	4-Channel 5V Relay Module				SAINSMART	2PH63083A											
	3	Optical liquid level sensor				Cynergy3	OLS710D3SH	Mouser.com	285-OLS710D3SH									
D	1	Push Button																
	1	Relay 250V 2x2A				Phoenix Contact	2987972	Mouser.com	651-2987972									
	1	Relay base 300VAC 12A				Phoenix Contact	28 33 521	Mouser.com	651-2833521									
E	2	10k ohm Resistor																
	4	5V LED Indicator Red/Orange/Green (10mm)				Gebildet	E738A											
	2	240V Water Valve 0-12 Bar				M&M	B297DVE											
	1	240V Oscillating Pump				Gorman Rupp Industries	14825-669											
F	1	240V Water Heater																
	1	Push Switch Button With Built-in 240V Neon Indicator																
	29	DIN Rail Terminal Blocks SRL 3 (Bus Connector)				Weidmuller	2614930000	Mouser.com	470-2614930000									
	1	DIN Rail Connector 8 Outputs																
	4	DIN Rail Terminal Blocks SRL 3				Weidmuller	2614930000	Mouser.com	470-2614930000									
G																		
H																		
Author: Sérgio Carmo				Component List						File: Component List								
Date: 23/08/2024										Folio: 10/10								