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

.

Resumo

Este projeto, tem como finalidade modernizar um destilador. Este destilador, era completamente analógico, foi realizado uma modernização na máquina removendo-se todos os componentes antigos e substituídos por componentes digitais, tal como LEDs (*light emitting diode*), relé e um microprocessador chamado de ESP32.

Devido ao ESP32 ser um microprocessador potente, é assim possível ter o *firmware* e um servidor WEB em execução num único microprocessador.

O *firmware* controla todo o funcionamento da máquina, tal como a administração do tanque de água. O *firmware* é programado em linguagem de programação C

A página WEB, é uma interface intuitiva para controle e monitorização da máquina.

A página WEB é programada em linguagem de programação HTML, JavaScript e AJAX.

Palavras-Chave: Destilador, HTML, C, Arduino, HTML, JS, AJAX, ESP32.

Abstract

g.

Keywords: Lorem, Ipsum, Lorem, Ipsum, Lorem, Ipsum.

LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS

AJAX - Asynchronous JavaScript and XML,

XML - Extensible Markup Language,

HTML - Hypertext Markup Language,

JS – JavaScript,

CSS - Cascading Style Sheets,

IDE - Integrated Development Environment,

JSON - JavaScript Object Notation,

SoC - System-on-Chip,

SRAM - Static Random Acess Memory,

LED – Light-emitting diode,

V – Volts,

A – Amps,

AC - Alternating current,

DC - Direct current,

SPI - Serial Peripheral Interface,

SRAM - Static Random Acess Memory,

NC - Normally Closed,

NO - Normally Open,

GPIO - General-Purpose Input Output,

SPDT - Single Pole, Double Throw,

ÍNDICE GERAL

ÍNDICE DE ANEXOS

Não foi encontrada nenhuma entrada do índice de ilustrações.

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1. INTRODUÇÃO

Lorem, felis ut adipiscing.

2. ENQUADRAMENTO E JUSTIFICAÇÃO DO TEMA

3. METODOLOGIA E MEIOS UTILIZADOS

O *firmware* da destiladora é programado em linguagem C, inclui os ficheiros de inclusão (ficheiros *header* 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.

Todo o projeto foi elaborado no IDE *Visual Studio Code* com as extensões *PlatformIO IDE, GitHub Copilot* e *Doxygen.*

Todo o projeto foi também elaborado no sistema operativo Fedora 40 (Linux).

4. LIMITAÇÕES DA PESQUISA/TRABALHO

Não encontrei muitas dificuldades/limitações na pesquisa acerca do trabalho, a comunidade Arduino é muito grande e por consequência, existe muita documentação disponível, talvez por haver tanta documentação, tenha havido uma certa dificuldade em filtrar o que realmente se adequasse às minhas necessidades, tal como o método de comunicação (AJAX) entre o *firmware* e a página WEB.

5. ESTRUTURA GERAL DO TRABALHO

6. PLATAFORMA FÍSICA

6.1. 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, etc.

Figura 1 - Destiladora

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 ebulição. 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.

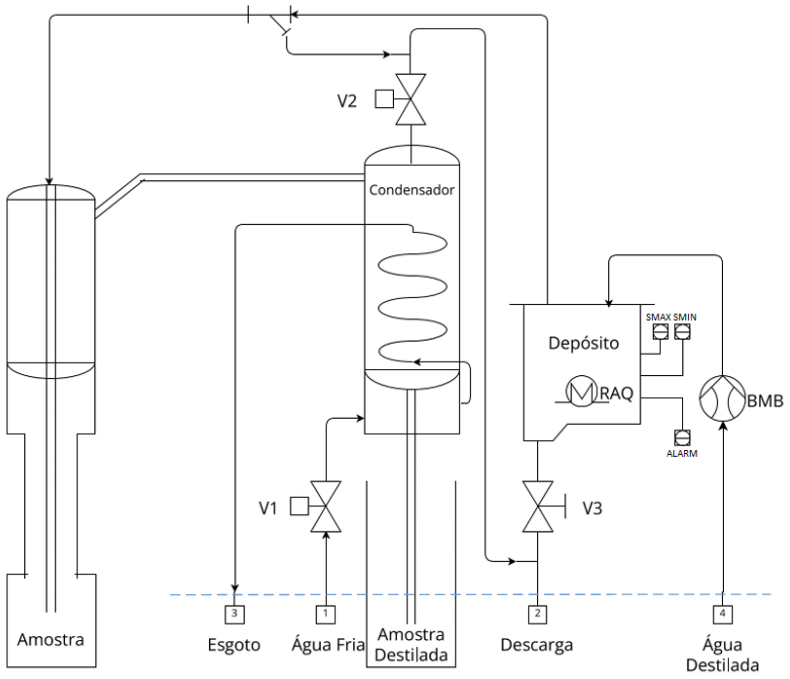


Figura 2 – Arquitetura Geral da Destiladora

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

6.2. Controlador Digital

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

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.

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 [5] é um tipo de memória volátil usada em sistema 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 as estas serem redimensionadas às necessidades do programador. 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 [6] é onde podem ser armazenados ficheiros diversos. 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

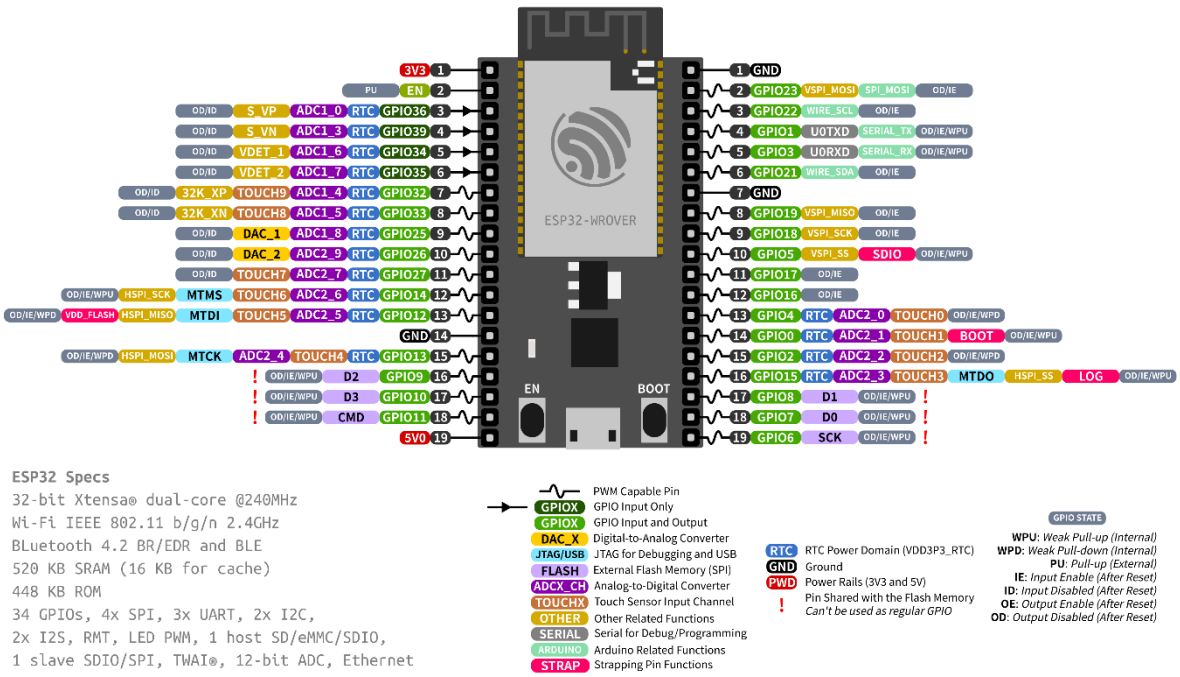


Figura 6 – Layout dos pins do ESP32

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

6.3. 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 6.1 (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).

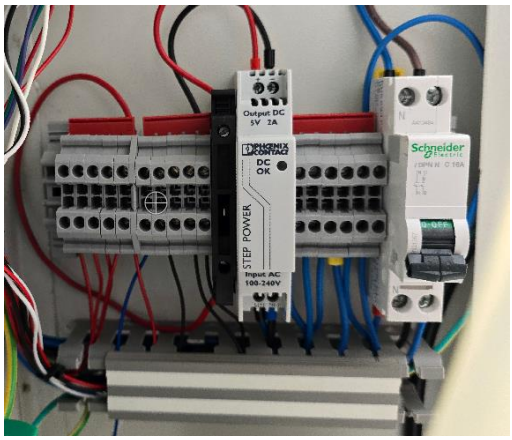


Figura 10 - Barramento Lateral

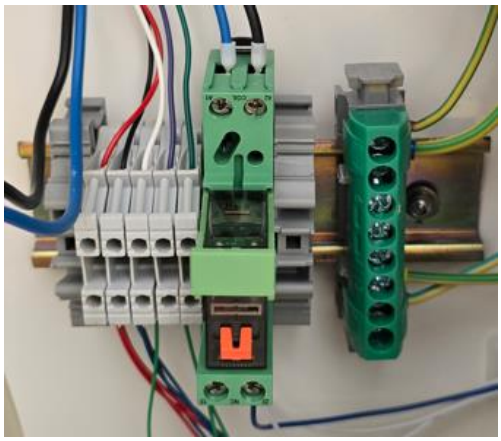


Figura 9 - Barramento Central

Figura 8 - Ponteiras Isoladoras e Alicate de Cravar

Figura 7 - Fonte 5VDC e proteção DC

É usada uma fonte de alimentação de 5VDC e 2A para fornecer energia ao microprocessador e componentes de baixa tensão. O circuito de 5VDC está também protegido por meio de um fusível de 1A ligado em série à saída da fonte DC. Foi adicionado um disjuntor bipolar de 16A para proteção do circuito elétrico de 230VAC (Figura 7).

6.4. Sensores

6.4.1. 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 bobines dos relês [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 bobines dos relés. Esta ação gerava um pico de corrente que por sua vez, gerava ruído nos fios de baixa tensão.

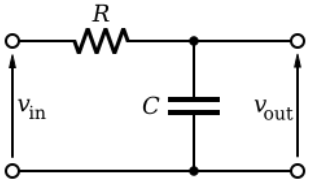


Figura 11 - Esquema Elétrico do Filtro Passa-Baixo Passivo

Foi então aplicada a Expressão (1) e como foi usado um condensador de tântalo de 220nF e uma resistência de 47OHMs, conforme Figura 12, o resultado da equação é aproximadamente ~15392Hz, 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.

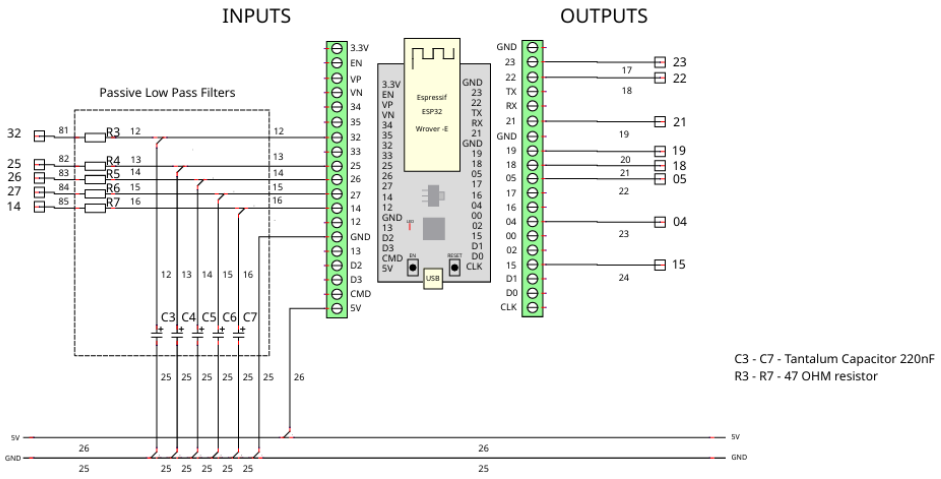


Figura 12 - Esquema Elétrico do ESP32 com o Filtro Passa-Baixo Implementado

6.4.2. 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).

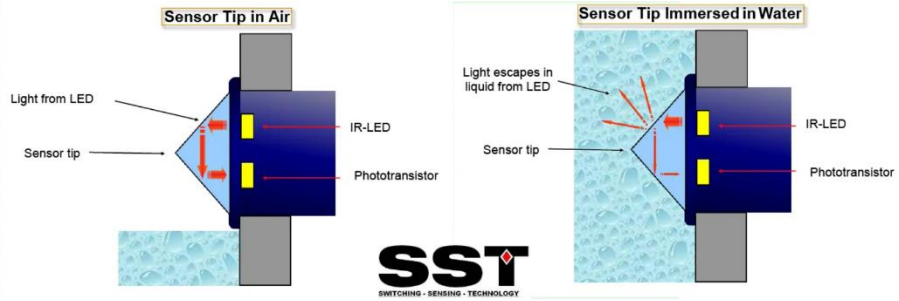


Figura 13 - Funcionamento dos Sensores de Nível [8]

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

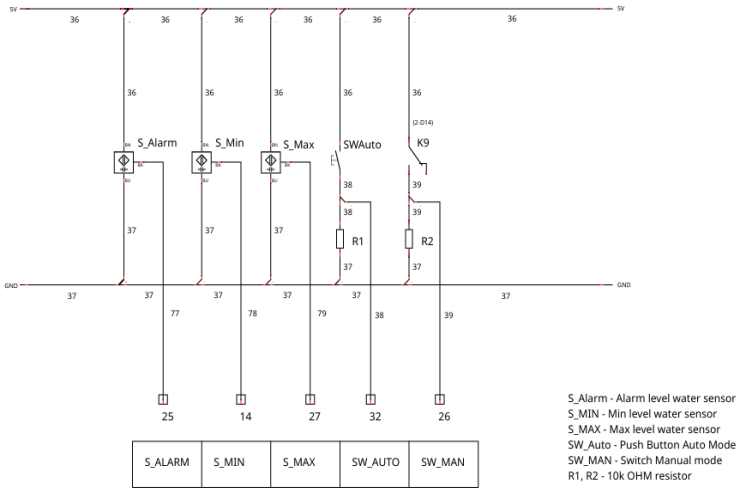


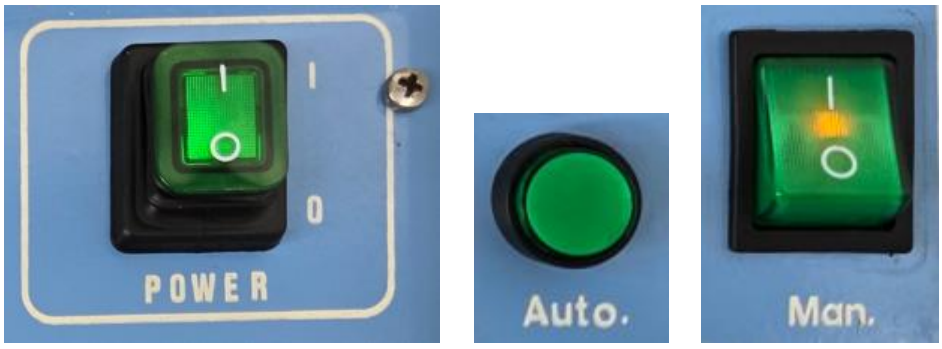
Figura 16 - Digital Inputs

6.4.3. 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

6.5. 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.

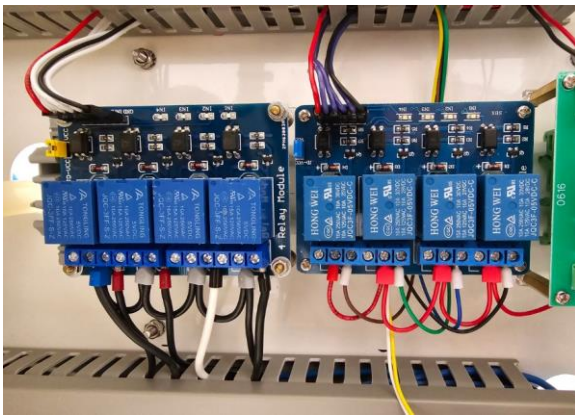


Figura 19 – Módulos de Relês

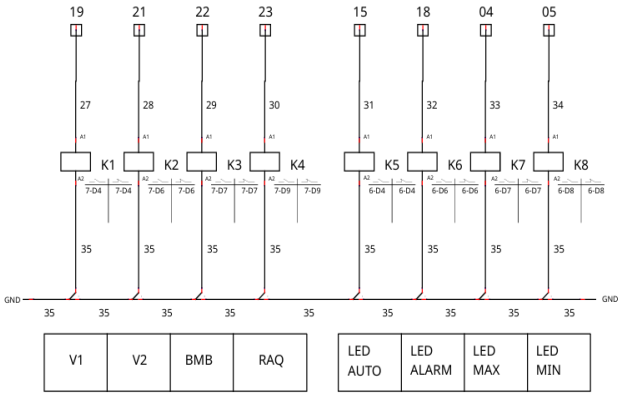


Figura 18 - Esquema Elétrico das Bobines dos Relés

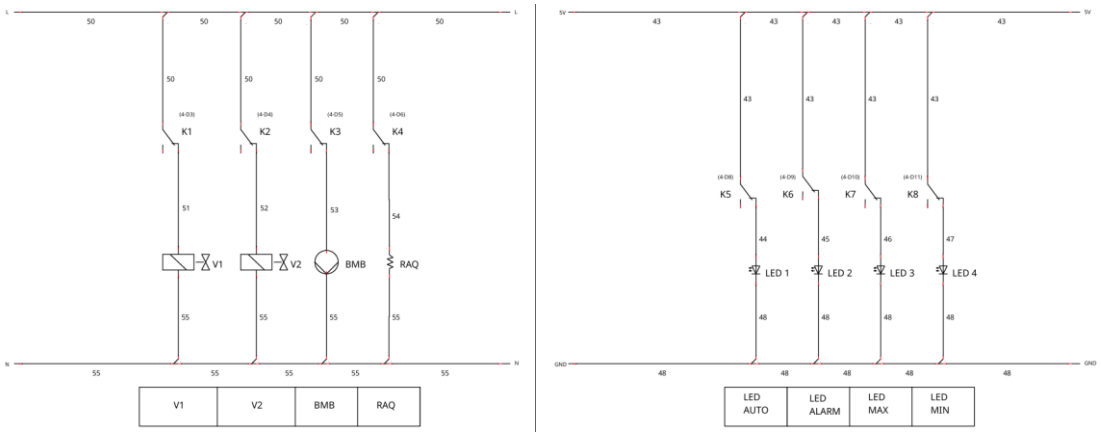


Figura 20 - Esquema Elétrico dos Contactos dos Relés

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 referenciado 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.

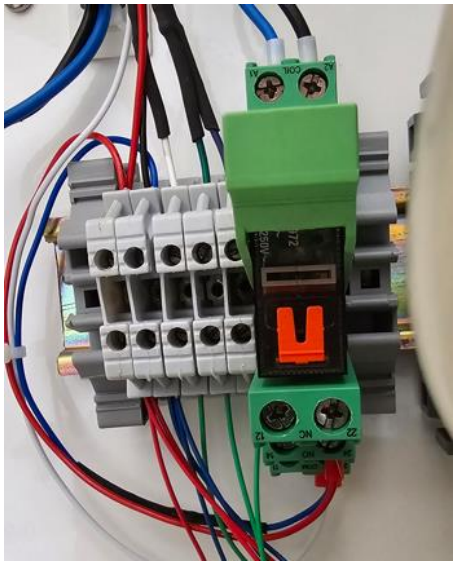


Figura 21 - Relé do Modo Manual

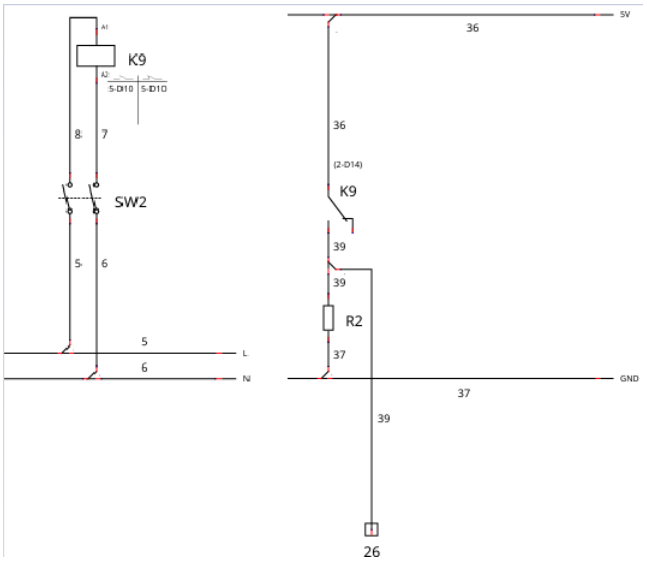


Figura 22 - Esquema Elétrico do Relé do Modo Manual

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

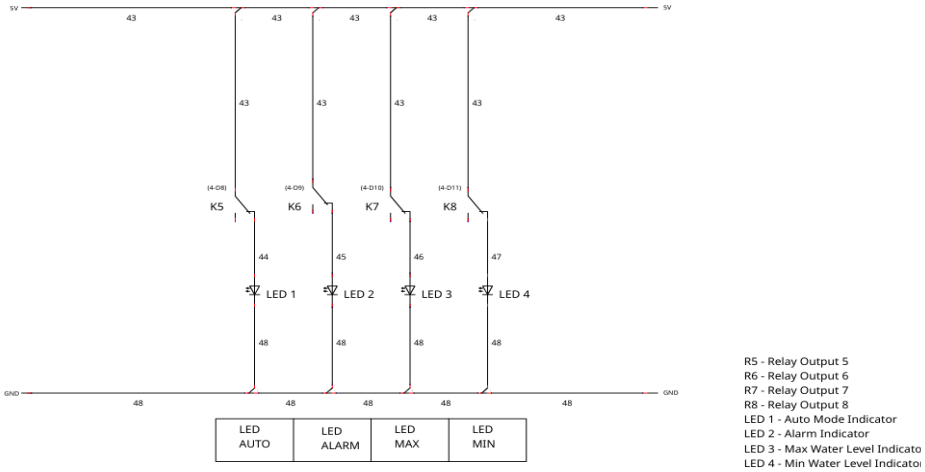


Figura 25 - Esquema Elétrico Indicadores

7. SOFTWARE

7.1. 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 como 26 pinos de I/O utilizáveis, conectividade Wi-Fi integrada de até 150 Mbps e uma CPU dual-core que permite uma separação eficiente entre o firmware e o servidor web. Apesar dessas vantagens, há limitações significativas como a memória restrita (8MB), que impede o uso de módulos adicionais como cartão SD e 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 ou menor desempenho de CPU. A página web será desenvolvida utilizando a biblioteca "ESPAsyncWebServer.h" para gestão do servidor assíncrono, e a "WiFiManager.h" será responsável pela gestão da conexão Wi-Fi, incluindo a criação de um *access point* para a configuração de redes desconhecidas, armazenando as credenciais na EEPROM para uso futuro.

7.1.1. Requisitos Gerais

1) Modo Automático
   a) Liga/desliga a máquina com o temporizador
2) Modo Manual
   a) Máquina sempre ligada em modo manual
3) Gestão de nível de água
   a) Nível máximo desliga a bomba
   b) Nível mínimo liga a bomba de água, liga resistência de calor e abre as válvulas
   c) Nível de alarme desliga a resistência de calor, liga o led luminoso de alarme, fecha todas as válvulas
4) Controlo da destiladora via página WEB

7.1.2. Requisitos específicos

1) Uso do *ESP32 WROVER-E DevKitC V4 board*
   a) Vantagens:
      i) 26 I/O pins usáveis
      ii) WIFI integrado - 802.11n (2.4 GHz), até 150 Mbps
      iii) CPU com 2 cores (core 0 corre *firmware* da máquina e core 1 o servidor WEB)
      iv) CPU trabalha a 40-MHz
      v) *Firmware* desenvolvida em C
      vi) Arquitetura do CPU é *open source*, é possível programar ao nível do CPU se necessário, ao contrário se fosse usado, por exemplo, um raspberry pi
   b) Desvantagens:
      i) Memória para *firmware* + página WEB disponível muito limitada (8MB)
      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 CPU (40MHz)
   c) Alternativas ao *ESP32 WROVER-E DevKitC V4*:
      i) Raspberry pi
         (1) Uma opção boa seria usar um RP, teria muito mais memória e poder de processamento, mas os I/O não seriam suficientes (14 pins usáveis) para o projeto
      ii) Arduino Mega

(1) O Arduino Mega tem muitos I/O (54 I/O), mas não tem WIFI e tem um CPU muito inferior (16MHz e 1 só core)

2) Página WEB
   a) Página WEB vai ser desenvolvida em HTML, CSS, JavaScript e AJAX
      i) 2.1.2. É usada a biblioteca "ESPAsyncWebServer.h" para criação e gestão de um servidor assíncrono WEB.
3) WIFI
   a) É usada a biblioteca "WiFiManager.h" para gestão do WIFI
      i) Com esta biblioteca, se não é conhecida a rede WIFI, ela cria um AP (*access point*) para o utilizador poder escolher, ligar e guardar as credenciais da nova rede WIFI na memória EEPROM e vai ler as credenciais quando a máquina é ligada para se ligar à nova rede.

7.2. Arquitetura Geral

7.2.1. Diagrama de Casos de Uso

O diagrama de casos de uso representado pela Figura 27, representa as relações dos processos decorridos no sistema. O utilizador controla o modo manual (*Manual Mode*) fisicamente e o modo automático (*Auto Mode*) da máquina tanto fisicamente como na web, o nível de água (*Water Lvl Management*) é administrado 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 e*mbedded*, este é também manipulado pelos sensores, em que 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*).

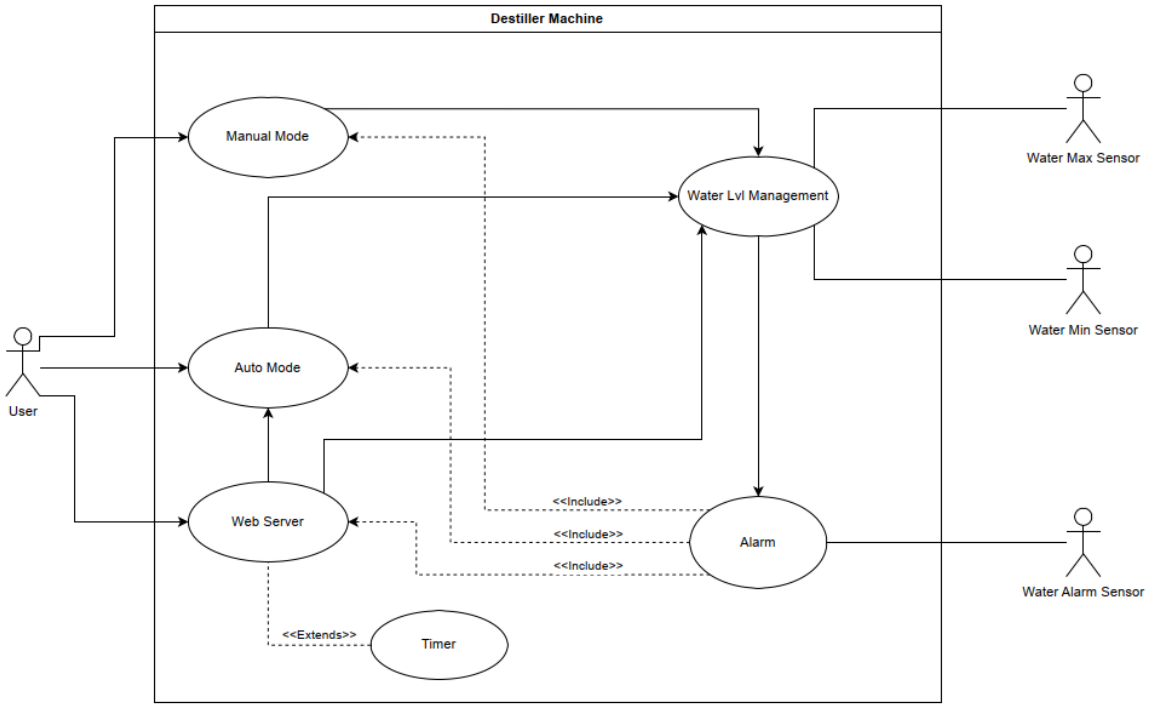


Figura 27 - Diagrama de Casos de Uso

7.2.2. Diagrama de Atividade do Ciclo de Programa

O Diagrama de Atividade do Ciclo de Programa representado na Figura 2, representa simplificadamente os dois ciclos presentes no projeto, em que o primeiro (Loop 1) é responsável por executar o *firmware* da destiladora (gerir indicadores, modos e a máquina de estados) e o segundo, por executar servidor web, o temporizador e gestão do Wi-Fi.

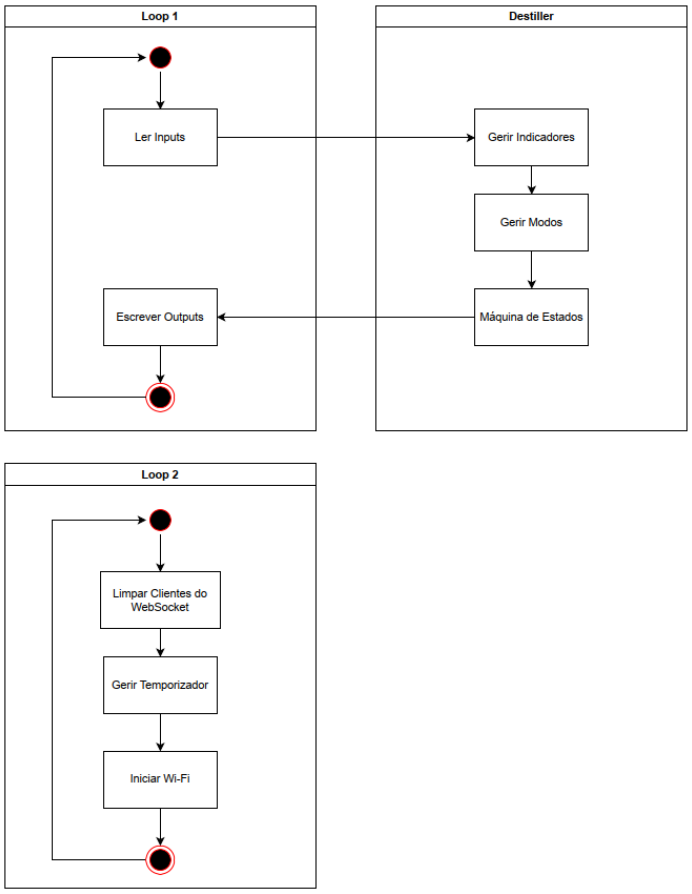


Figura 28 - Diagrama de Atividade do Ciclo de Programa

Ver se faz sentido mais algum UML

7.2.3. Modelo de Camadas do Firmware

A seguinte arquitetura do sistema foi criada para que as chamadas sejam únicas, exatas e retornáveis para o ciclo de máquina e o código seja de mais fácil manutenção (Figura 29).

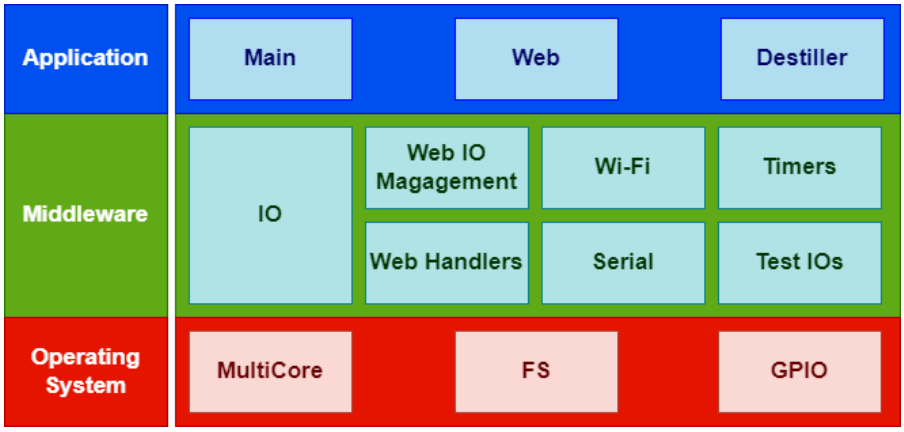


Figura 29 - Esquema Representativo das Camadas do Firmware

- **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.
        - **OS_multi_core –** Classe responsável pelo início e configuração do segundo core do ESP32.
        - **OS_FS –** Classe inicia o sistema de ficheiros SPIFFS para armazenamento de ficheiros.
        - **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.
- **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.
        - **MD_IO** - Classe responsável pela gestão dos indicadores, modo e trocar o estado do modo automático dependendo do estado atual.

- **MD_Web_IO_Mngmnt –** Esta classe contém os *get* e sets dos IO's em semelhança à classe "OS_GPIO" mas relativo ao servidor web.
- **MD_Web_Handlers –** É nesta classe que estão os *handlers* para os pedidos do servidor web.
- **MD_wifi –** Classe 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.
- **MD_Serial –** É nesta classe que é iniciado e configurado a comunicação com a porta de série para poder escrever no terminal.
- **MD_Timer –** Classe responsável pela gestão do temporizador da página web.
- **MD_Test_IOs –** Classe dedicada somente a testar os *outputs* do sistema.

- **Camada AP (*Application*)**
  - Esta camada inclui todas as classes 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.
    - **AP_main –** É esta a classe 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.
    - **AP_Destiler –** Classe responsável pela máquina de estados do sistema.

7.3. - modelo de software, ficheiros, dependências

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.
- Documentation – Contém tudo o que está relacionado com a documentação, tal como o relatório.
- Include – Contém todas os ficheiros *headers* necessários para o funcionamento do *software*.
- Lib – Esta pasta, contém as bibliotecas necessárias para o *software*.
- Src – Contém os ficheiros ‘.cpp’
- Na pasta raiz, estão ainda 2 ficheiros necessários para o funcionamento:
    - “partitions.csv”, que contém a configuração das partições da memória do ESP32.
    - “platformio.ini”, contém a configuração da extensão PlatformIO.

7.4. 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. 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 [9]. O JavaScript é utilizado para adicionar interatividade e dinamismo à página, permitindo a manipulação de elementos e o envio de pedidos assíncronos ao servidor [10] [11].

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 [12]. 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 [13]. Isto é crucial para garantir que os dados exibidos na página web estejam sempre atualizados sem a necessidade de múltiplos pedidos ao servidor [14].

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 [15]. (ver Figura 30 e Figura 31).

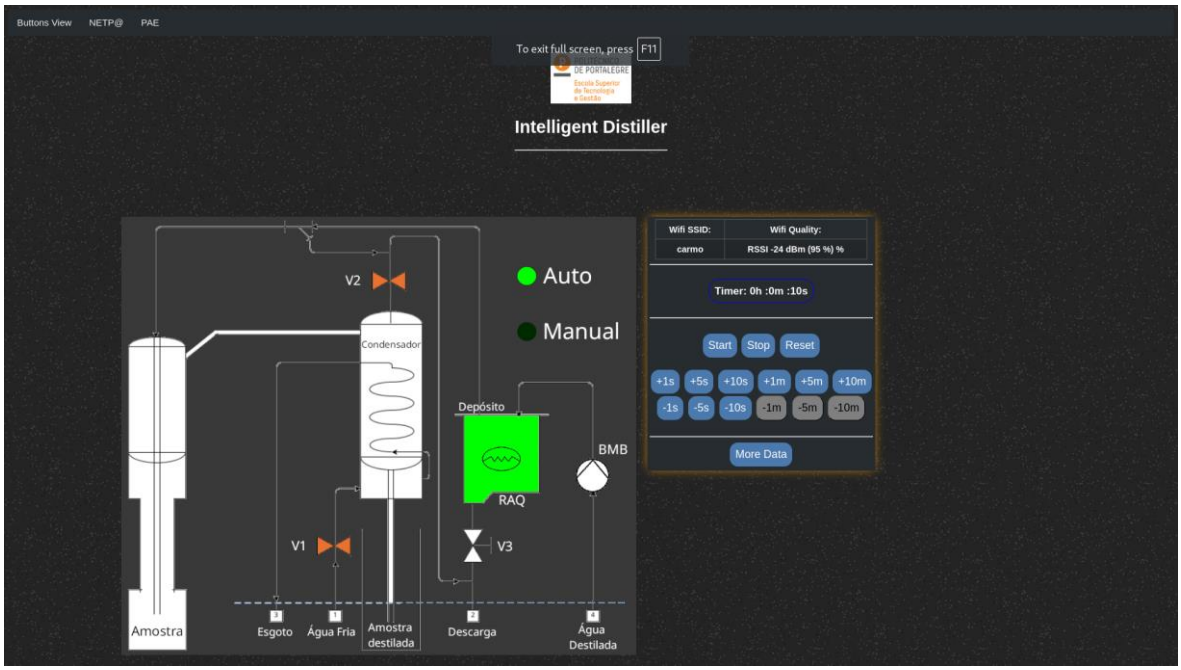


Figura 30 - Página Web com Imagens

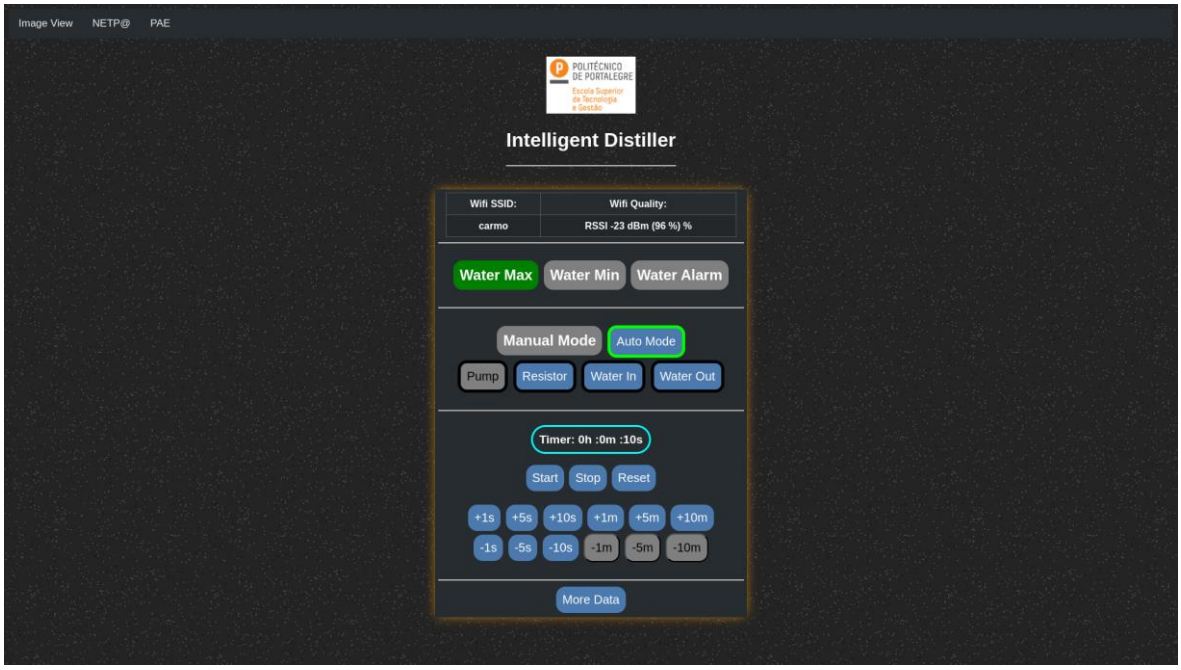


Figura 31 - Página Web com Botões

7.5. Modelo de Funcionamento

Diagrama de Sequência do funcionamento (user level) UML

Explicar o modo manual, automático, controlo de nível, …

9. TESTES

9.1. Prototipagem em Placa de Ensaio

Para ter um primeiro protótipo e ter uma plataforma independente da destiladora para testes, foi simulado toda a destiladora em uma 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*.

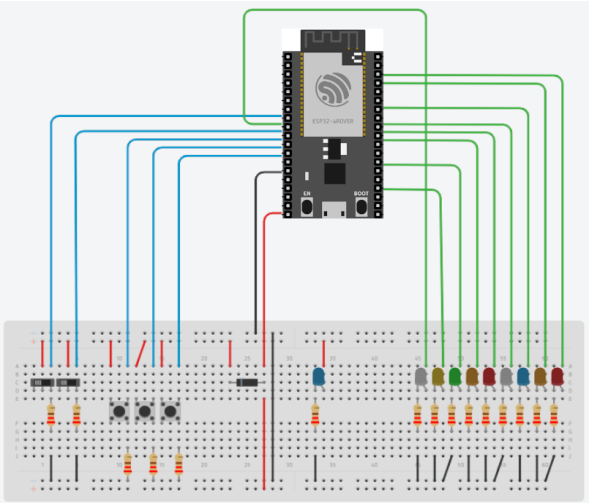


Figura 33 - Esquema de Ligação do ESP32 na Placa de Ensaio

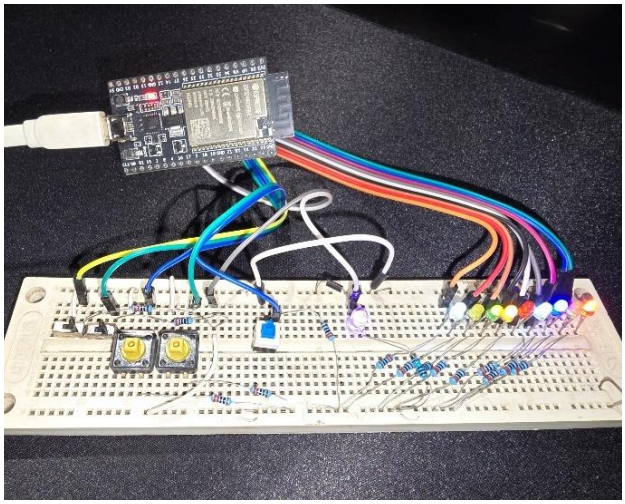


Figura 32 - ESP32 na Placa de Ensaio

Tabela 4 - Componentes Usados Placa de Ensaio

Quantidade	Descrição
17	Fios Macho-Femea
15	220 ohm Resistor
1	Esp32-Wrover-E
1	Placa de ensaio
3	Push-button
2	Interruptor
9	LED 2,5mm cores várias
1	LED 3,5mm
1	Diodo 1N4001

Foi criada esta plataforma de testes para ser desenvolvido todo o *software* completamente independente da destiladora, com isto traz a vantagem de que se houver algum bug no desenvolvimento do *software*, não cause nenhum problema ao *hardware* da máquina, já que existem componentes a trabalhar com tensão direta da

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

10. CONCLUSÃO

Lorem.

11. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Espressif, “Espressif-About,” [Online]. Available: https://www.espressif.com/en/company/about-espressif. [Acedido em 08 2024].

[2] Microsoft, “Visual Studo 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] W3C, “HTML & CSS Standards.,” *World Wide Web Consortium (W3C),* 2023.

[10] J. Robbins, Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics., O'Reilly Media, 2018.

[11] D. Flanagan, JavaScript: The Definitive Guide., O'Reilly Media, 2020.

[12] J. J. Garrett, Ajax: A New Approach to Web Applications, Adaptive Path, 2005.

[13] A. Pérez, ESP32 Development Using Web Technologies., Packt Publishing, 2022.

[14] B. a. A. B. Lubbers, WebSockets: A Modern Web Standard for Real-Time Communication, Apress, 2021.

[15] S. Monk, Programming the ESP32: Developing IoT Projects with Wi-Fi, Bluetooth, and Low-Power Devices, McGraw-Hill, 2021.

[16] M. Nardi, “Elimine INTERFERÊNCIAS dos seu projetos feitos com ARDUINO!,” YouTube, 2022. [Online]. Available: https://www.youtube.com/watch?v=uyItIeIbdCg. [Acedido em 08 2024].

[17] 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].

[18] E. Wings, “Digital GPIO of Arduino,” [Online]. Available: https://www.electronicwings.com/arduino/digital-gpio-of-arduino. [Acedido em 08 2024].

ANEXOS

Anexo 1 – “Instruções de Uso”

Instruções de uso

Modo Automático via web

Modo Automático via botão

Modo Manual via botão

Modo Automático

Via Browser

Start → Inicia o processo de destilação

Fim do tempo → Fim do Processo de Destilação

Stop → Fim do Processo de Destilação

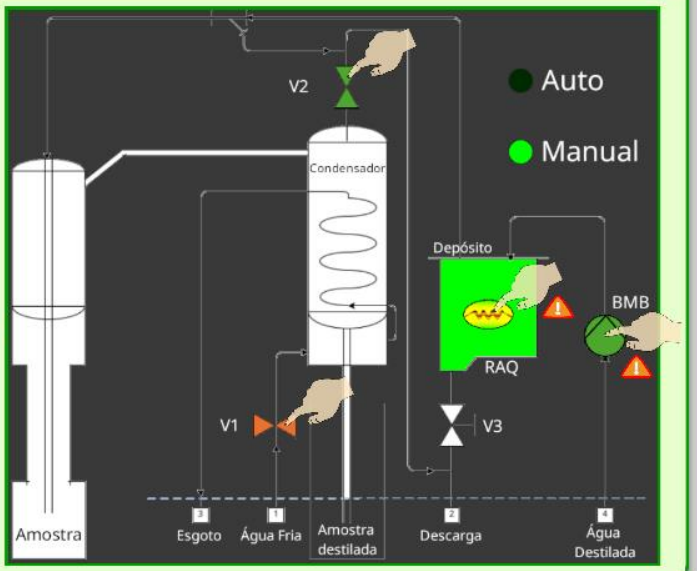
Reset → Reset ao tempo

Legenda Modo Automático

- **Enchimento Automático**
  BMB (Bomba de Água) liga atingindo o nível mínimo, e desliga no nível máximo do depósito.
- **Vapor**
  RAQ (Resistência de Aquecimento) entra em funcionamento se o líquido do depósito estiver entre o nível máximo e mínimo.
- **Circuito de Condensação**
  As válvulas **V1** e **V2** estão ativas durante o tempo programado.

Modo Manual

Via Web



Legenda Modo Manual

- **RAQ** (Resistência de aquecimento) só liga se o nível de água estiver acima de nível alarme
- **BMB** (Bomba de água) só liga se o nível de água estiver abaixo de nível de água máximo
- **Modo Manual** só é ativado via botão físico no painel da máquina

Anexo 2 – Esquema Elétrico