



Instituto Superior de Engenharia

Licenciatura em Engenharia Eletrotécnica

Sistema automático de Micro Hidroponia

Relatório da Unidade Curricular de Projeto de Automação

COIMBRA

2021



Instituto Superior de Engenharia

Licenciatura em Engenharia Eletrotécnica

Sistema automático de Micro Hidroponia

Relatório da Unidade Curricular de Projeto de Automação

Aluno

Filipe Manuel Batista Martins 2008007485

Orientador

Professor Marco José da Silva

COIMBRA

2021

Agradecimentos

Para a realização deste projeto foi de extrema importância o acompanhamento e apoio de duas pessoas pelas quais me posso orgulhar de ter ao meu lado.

Agradeço assim à minha namorada e melhor amiga, Jéssica Andrade pelo apoio e pressão constantes para que este projeto fosse terminado e concluído da melhor forma, apesar de não ser esta a sua área de conhecimentos, sempre se dispôs a ajudar em tudo o que fosse necessário e sempre com palavras de incentivo.

Deixo aqui também um agradecimento a um ex-aluno do ISEC e grande amigo, Daniel Almeida, por todo o apoio presencial e virtual a que se dispôs para que todas as minhas dúvidas e problemas fossem resolvidos da melhor forma. Um grande obrigado por toda a orientação e colaboração ao longo deste projeto.

Resumo

Neste documento é apresentada a metodologia e abordagem usada num projeto que visou criar uma pequena estufa de hidroponia que se demonstrasse exequível no enquadramento da unidade curricular de Projeto de Automação da Licenciatura em Engenharia Electrotécnica, do Instituto Superior de Engenharia de Coimbra. Foi também objetivo que a estufa desenvolvida produzisse resultados pelo cultivo de vários tipos de plantas. Este trabalho compreende assim um estudo e projeto detalhado da montagem da estrutura de uma estufa, e um estudo analítico do desempenho da circulação da solução nutritiva ao longo de toda a estrutura.

Foram caracterizados empiricamente os parâmetros do circuito hidropónico da estufa. O trabalho desenvolvido resulta de um estudo prático que procura a inovação, o que se mostra difícil uma vez que nos dias de hoje surgem diariamente novas tecnologias. Assim, a meta a alcançar prendeu-se com o facto de se querer elaborar um sistema prático para utilização em qualquer ambiente, tanto para espaços pequenos/grandes como para interior/exterior que permita ao utilizador uma maior praticidade no decorrer da plantação e apresentando-se como uma solução completamente automatizada. Articulado com este conceito de automação surgiu o desenvolvimento de uma *interface web* que torna acessível ao utilizador a criação de uma base de dados de produtos para cultivo e do controlo eletrónico remoto da estufa.

Todo este projeto assenta em dois componentes principais que controlam todo o sistema, nomeadamente um sistema Arduíno Micro e uma *Raspberry Pi*. O primeiro recebe os dados obtidos pelos sensores da estufa e faz atuar as saídas do sistema designadamente os motores, electroválvulas e lâmpadas, enquanto comunica através de uma porta série com o segundo componente. Por sua vez, o Raspberry Pi é usado como o servidor *WEB* da estufa, é o supervisor do sistema, fazendo a análise dos dados obtidos pelo Arduíno e que são apresentados numa base de dados, alcançáveis numa página *web*. Para concluir o ciclo, é retornado ao Arduíno a informação necessária que fará atuar as suas saídas.

A componente teórica e analítica do trabalho de projeto foi comprovada experimentalmente através de dois métodos de execução diferentes. Nestes Ensaios foram analisadas as principais vantagens e desvantagens para um funcionamento seguro e regular do sistema na plantação e criação de alfaces que foi usada como exemplo para os testes.

Palavras chave: Hidroponia, Arduíno, *Raspberry Pi*, *Software*.

Abstract

This document presents the methodology and approach used in a project that aimed to create a small hydroponics greenhouse that proved to be feasible within the framework of the Automation Project curricular unit of the degree in Electrotechnical Engineering, of the Coimbra Institute of Engineering. It was also an objective that the developed greenhouse would produce results by cultivating various types of plants. This work thus includes a detailed study of the assembly of the greenhouse structure, and an analytical study of the performance of the circulation of the nutrient solution throughout the entire structure.

The parameters of the hydroponic circuit of the greenhouse were empirically characterized. The work developed is the result of a practical study that seeks innovation, which is difficult since these days new technologies emerge daily. Thus, the goal to be achieved was related to the fact of wanting to develop a practical system for use in any environment, both for small/large spaces and for indoor/outdoor that is user-friendly during the planting and presenting itself as a fully automated solution. Linked with this concept of automation came the development of a web interface that makes the creation of a database of products for cultivation and the remote electronic control of the greenhouse accessible to the user.

This entire project is based on two main components that control the entire system, namely an Arduino Micro system and a Raspberry Pi. The first receives the data obtained by the greenhouse sensors and activates the system outputs, namely the motors, solenoid valves and lamps, while communicating through a serial port with the second component. In turn, the Raspberry Pi is used as a webserver from the greenhouse, is the supervisor of the system, analyzing the data obtained by Arduino and which are presented in a database, accessible on a web page. To complete the cycle, the necessary information is returned to the Arduino that will make adjusts its outputs.

The theoretical and analytical component of the project work was experimentally proven through two different execution methods. In these tests the main advantages and disadvantages for a safe and regular operation of the system in the planting and growing of lettuce were analyzed, which was used as an example for the tests.

Keywords: Hydroponics, Arduino, Raspberry Pi, Software

Índice

Índice	9
Lista de Abreviaturas	11
Lista de Figuras	13
Lista de Tabelas	17
1 Enquadramento Teórico do Estudo	21
1.1 A Hidroponia	21
1.1.1 Breve história da Hidroponia	21
1.2 Hidroponia em Portugal	23
1.2.1 Jungle Greens	24
1.2.2 Hidroflora	24
1.2.3 Horta Grande	25
1.3 Tipos de Sistemas Hidropónicos	26
1.3.1 <i>Drip System</i> - Sistema Hidropónico por Gota	26
1.3.2 Ebb & Flow - Sistema de Inundação e Drenagem	27
1.3.3 N.F.T. - Sistema de Técnica de Filme Nutriente	28
1.3.4 Water Culture - Sistema de Cultura em Água	29
1.3.5 Aeroponics - Sistema Aeropónico	30
1.3.6 <i>Wick</i> - Sistema de Pavio	31
1.3.7 Comparação dos Sistemas Hidropónicos	32
1.4 Análise da solução nutritiva por medição da condutividade elétrica	33
1.5 Controlo de pH da solução nutritiva	35
2 Elaboração de um novo sistema hidropónico	37
2.1 Componentes de Hardware integrados no sistema Hidropónico	39
2.1.1 Sistema Arduino Micro	40
2.1.2 Raspberry Pi	42
2.1.3 Módulo de Relés	43
2.1.4 Fonte de Alimentação	44
2.1.5 Módulo de pH	45
2.1.6 Medidor de EC - Versão 1	46
2.1.7 Medidor de EC – Versão 2	47
2.1.8 Módulo LDR	48
2.1.9 Bomba Peristáltica	49
2.1.10 Bomba Submersível	50

2.1.11	Lâmpada Espectro Completo	51
2.1.12	Electroválvula Solenoide	52
2.2	Software utilizado nos módulos do sistema hidropónico	53
2.2.1	Supervisão e Controlo de sensores e atuadores em linguagem C/C++	54
2.2.2	Interface entre Base de dados e sistema Arduíno em Python	54
2.2.3	Base de Dados do sistema existente na Raspberry Pi	55
2.2.4	WordPress - Interface gráfica WEB com o utilizador	57
3	Ensaio do Sistema	61
3.1	Tabelas da Base da Dados de Suporte ao Sistema	62
3.2	Página WEB, Hidroponia de Interior	64
3.3	Ensaio de funcionamento com gestão no Arduino – Ensaio 1	67
3.3.1	Semente Inserida	68
3.3.2	Reservatório Cheio	69
3.3.3	Controlo Luminosidade	69
3.3.4	Controlo de EC	71
3.3.5	Controlo de pH	72
3.3.6	Bombear Solução	73
3.3.7	Escrever valores na base de dados	74
3.3.8	Sleep Mode	74
3.3.9	Ler Base de Dados	75
3.4	Esquema Ensaio 1	75
3.5	Conclusões sobre Ensaio 1	75
3.6	Ensaio funcionamento com gestão na Raspberry Pi – Ensaio 2	77
3.6.1	Semente inserida	80
3.6.2	Reservatório Cheio	80
3.6.3	Controlo Luminosidade	81
3.6.4	Controlo de EC	81
3.6.5	Controlo de pH	82
3.6.6	Bombear Solução para as Plantas	82
3.6.7	Escrever na Base de Dados	82
3.6.8	Tempo de Espera	82
4	Conclusões	85
	Referências Bibliográficas	87
	Apêndices	89

Lista de Abreviaturas

a.C	Antes de Cristo
CO ₂	Dióxido Carbono
EC	Condutividade Elétrica
LDR	<i>Light Dependent Resistor</i> – Resistência dependente de Luz
pH	Unidade de medição da acidez relativa ou alcalinidade de uma solução.
RPi	Raspberry Pi
SGC	Sistema Gestão de Conteúdo (da expressão inglesa “Content Manager System”)
TDS	<i>Total Dissolved Solids</i> – Sólidos dissolvidos totais (na água)

Lista de Figuras

Figura 1 - Drip System	27
Figura 2 - Ebb & Flow	28
Figura 3 - N.F.T.	29
Figura 4 - Water Culture	30
Figura 5 - Aeroponics	31
Figura 6 - Wick	32
Figura 7 - Diagrama funcional do sistema	37
Figura 8 - Diagrama Ilustrativo do Sistema	40
Figura 9 - a) Arduíno Mega e b) Arduíno Micro	41
Figura 10 - Raspberry Pi	42
Figura 11 - Módulo Reles Esquemático	44
Figura 12 - Módulo pH	45
Figura 13 - Esquema Módulo pH	46
Figura 14 - Sensor de Condutividade	47
Figura 15 - Esquema do sensor de condutividade	48
Figura 16 - Modulo LDR	49
Figura 17 - Esquema do sensor LDR	49
Figura 18 - Bomba Peristáltica	50
Figura 19 - Exemplo da BD do sistema	56

Figura 20 - Estrutura da Estufa	61
Figura 21 - Tabela Demonstrativa CULTURES	62
Figura 22 - Tabela Demonstrativa INPUT	63
Figura 23 - Tabela Demonstrativa LEVEL	63
Figura 24- Tabela Demonstrativa OUTPUT	64
Figura 25 - Página WEB Plantar	65
Figura 26- Página WEB Dados	65
Figura 27 - Página WEB Manual	66
Figura 28 - Nível de Líquidos	66
Figura 29 - Fluxograma representativo da rotina do Ensaio 1	67
Figura 30 - Página Web - Secção Plantar (Nova Semente)	68
Figura 31 - Interface para Ligar/Desligar Lâmpada	70
Figura 32 - Interface para aumentar EC	71
Figura 33 - Interface para diminuir pH	73
Figura 34 - Esquema de ligações ao Arduino	76
Figura 35 - Fluxograma representativo da rotina ensaio 2	77
Figura 37 - Foto do andar superior da estufa	89
Figura 38 - Reservatório de solução	90
Figura 39 - Deposito de Nutrientes	90
Figura 40 - Componente de Controlo	90

Figura 41 - Raspberry Pi	91
Figura 42 - Arduino Micro	91
Figura 43 - Módulo de pH (Sem ponta de prova)	92
Figura 44 - Módulo de EC (Sem ponta de prova)	92
Figura 45 - Exemplo da SHELL Python com ciclo da rotina	93

Lista de Tabelas

Tabela 1 - Comparação entre sistemas hidropônicos	33
Tabela 2 - Características Arduino Micro	41
Tabela 3 - Características Raspberry Pi	43
Tabela 4 - Características Módulo de Relés	44
Tabela 5 - Características Fonte de Alimentação	44
Tabela 6 - Características Módulo pH	46
Tabela 7 - Características módulo EC	48
Tabela 8 - Características módulo LDR	49
Tabela 9 - Características Bomba Peristáltica	50
Tabela 10 - Características Bomba Submersível	50
Tabela 11 - Características Lâmpada Espectro Completo	51
Tabela 12 - Características Válvula Solenoide	52
Tabela 13 - Comparação entre blocos e função de estado	79

1 Introdução

A hidroponia permite o crescimento de plantas em solução mineral nutritiva sem a presença de solo. Em hidroponia o solo é substituído por um meio inerte de suporte, com ou sem substrato, uma solução nutritiva dissolvida, e um rizoma bem arejado, quer pela oxigenação da solução por difusão à superfície, injeção de bolhas no corpo da solução ou pelo envolver de uma porção relevante do rizoma em meio atmosférico [1].

A principal desvantagem das técnicas hidropónicas prende-se com a necessidade, em algumas técnicas, de ter sempre uma fonte de energia ligada a bombear água ou a injetar solução nutritiva. Isto sem considerar outros equipamentos usados para controlar ativamente o ambiente climático interior à unidade.

Quanto às vantagens da hidroponia, o facto de o cultivo ser desenvolvido em ambiente controlado possibilita a regulação otimizada do meio ambiente no qual cresce a colheita. O controlo é feito tanto no ambiente dos rizomas, como na estrutura aérea da planta, permitindo maior produtividade e período alargado de crescimento, ou seja, é possível manter a época de crescimento, durante todo o ano.

Todos os parâmetros do ambiente de crescimento podem ser controlados usando a técnica da hidroponia, desde a temperatura do ar e dos rizomas, à concentração da solução nutritiva, passando pela humidade do ar, a composição em CO₂ atmosférico, a radiação luminosa e o sistema hidráulico [1].

Com vista à finalização da unidade curricular de Projeto de Automação e consequente conclusão da Licenciatura em Engenharia Eletrotécnica, no Instituto Superior de Engenharia de Coimbra, este trabalho visa aplicar os conhecimentos adquiridos ao longo de toda a licenciatura, salientando, sobretudo, a componente de desenvolvimento de Hardware, e a componente de Software, utilizando diferentes linguagens de programação.

Neste trabalho é apresentado o desenvolvimento de uma estufa de Hidroponia que incorpora um servidor web. O servidor web é suportado por uma plataforma Raspberry Pi, que em conjunto com um Arduíno Micro têm o propósito de fazer a gestão do sistema de cultivo.

O objetivo principal do trabalho foi o desenvolvimento de uma pequena estufa com sistema automático de controlo de solução nutritiva. O processo de controlo baseia-se na análise de pH

e condutividade da solução. Todo o processo pode ser supervisionado através de uma interface web que ajusta uma base de dados do sistema. No desenvolvimento do sistema foi usado C *Python*, PHP, HTML e JavaScript.

Foi elaborado um estudo teórico-prático baseado em publicações científicas como livros, dissertações de mestrado e doutoramento relacionadas com a temática. Realizaram-se ainda contactos com duas entidades, nomeadamente, um agricultor que integra o grupo de produtores fornecedores da empresa Sonae, que detém uma grande estufa hidropónica de cultivo de alfaces, na região de Lisboa; e ainda um vendedor de produtos hidropónicos, na zona de Mira.

O presente trabalho está organizado em quatro capítulos. O primeiro capítulo descreve o desenvolvimento da Hidroponia ao longo dos anos, alguns tipos de sistemas hidropónicos, explicando resumidamente o seu princípio básico de funcionamento. Também são apresentados três casos recentes de sucesso da implementação de sistemas hidropónicos em Portugal. Para finalizar faz-se ainda referência a dois componentes importantes dos sistemas hidropónicos: a análise e controlo de solução nutritiva por medição de condutividade elétrica e pH.

No segundo capítulo, é exposta uma análise descritiva dos componentes de *hardware* e o *software*/linguagens de programação utilizadas. É descrito o princípio de funcionamento do sistema e posteriormente são apresentadas e justificadas a escolha de módulos de Hardware que instrumentam o sistema hidropónico. A nível de Software é descrito a interligação entre os vários módulos do sistema.

Em seguida, no capítulo três são apresentados dois ensaios de desenvolvimento com base de arquitetura diferentes. No primeiro ensaio é usado o módulo Arduino como supervisor do sistema enquanto que a Raspberry é usada apenas como módulo de interface e armazenamento dos dados recolhidos. No segundo ensaio o módulo Arduino faz controlo de baixo nível (sensores e atuadores) e interface com a Raspberry Pi. A Raspberry Pi incorpora agora o armazenamento de dados, a supervisão do sistema e interface com o utilizador.

Por último são apresentadas as conclusões sobre a arquitetura do sistema evidenciando que a hidroponia é mais segura e fiável que os métodos de cultivo tradicionais. É concluído que foi possível ultrapassar as dificuldades encontradas na elaboração do projeto e as suas mais valias.

2 Enquadramento Teórico do Estudo

Neste capítulo é apresentada a definição de hidroponia acompanhada de uma breve evolução histórica. Seguidamente, é dada enfoque à hidroponia em Portugal, destacando o seu impacto positivo na economia, e descritos os seis tipos de sistemas hidropónicos mais utilizados. Por fim, dá-se a conhecer o trabalho recente de três empresas nacionais nesta área.

2.1 A Hidroponia

Segundo [2], a hidroponia é um termo que descreve a produção de plantas sem solo, sendo uma técnica em que as plantas crescem, com ou sem meio de suporte, graças aos nutrientes que existem na solução usada para rega.

Existem sistemas hidropónicos onde as raízes podem estar suspensas em líquido e outros onde estão apoiadas em substrato inerte como, por exemplo, a fibra de coco.

Os diversos tipos de sistemas de hidroponia diferem entre si pelo modo em que a solução nutritiva entra em contato com as raízes. Como tal, para obter um conjunto hidropónico é essencial uma estrutura para suportar a planta, um reservatório para a solução nutritiva, e um meio de contato entre as raízes e a solução nutritiva.

2.1.1 Breve história da Hidroponia

Estudiosos e arqueólogos estudam há muito tempo os intrincados sistemas de rega que sustentavam os enormes jardins suspensos da Babilónia, criados pelo rei Nabucodonosor II como um presente para a sua esposa, datados de 500 a.C. Com camadas de pedras segurando as plantas, a água era fornecida através de um fluxo consistente dos reservatórios centrais de água. As plantas eram alimentadas e abastecidas com bastante aeração através do fluxo consistente de água até às suas raízes [3].

Segundo o estudo empírico de [4], no início do cultivo de arroz, houve várias tentativas de o cultivar no solo. Não obstante, após inúmeras inundações sazonais significativas, muitas outras culturas de sustento foram destruídas, sendo que o arroz, não apenas resistiu às condições aquosas, como prosperou. Face a esta revelação, o arroz foi intencionalmente cultivado em sistemas organizados de água, crescendo melhor, e

resistindo a mais doenças e pragas do que outras culturas. Arqueólogos declaram o uso da hidroponia em Roma desde o século I d.C. O imperador romano Tibério usava o conceito hidropónico para desfrutar de alguns vegetais fora de estação, nomeadamente pepinos. Acredita-se que seria uma forma inicial de estufa, pois, ao usar um material transparente (como pedra ou vidro), os pepinos fora de estação teriam ampla luz solar, calor e retenção de humidade. Outra civilização notável que usou a hidroponia foram os Astecas. O sistema hidropónico era semelhante a uma fazenda de jangada flutuante. Ao criar balsas densas a partir de juncos protegidos com raízes secas e duras, as colheitas flutuavam nos canais. Leonardo da Vinci estava à frente do seu tempo nas suas revelações sobre o crescimento das plantas, determinando que as plantas precisavam de minerais para crescer, que absorvem do solo. Notavelmente, percebeu que isso ocorre apenas com a ajuda da água, cuja ausência impedia a absorção de nutrientes. Da Vinci, abordou a forte importância da irrigação, estabelecendo alguns dos princípios básicos que guiam a hidroponia atualmente. A partir do ano 1600, as pessoas começaram a tentar criar métodos para proteger as culturas de intempéries e aumentar as capacidades de colheita. Embora essas técnicas de crescimento não fossem estritamente hidropónicas, sinalizavam o início de um interesse crescente no progresso de métodos mais avançados de colheita. Sir Francis Bacon detém o título do primeiro livro "moderno" publicado sobre cultivo de plantas num ambiente sem solo.

Na década de 1920, William Gericke, um professor da Universidade da Califórnia em Berkeley começou a promover o crescimento de culturas sem solo e mais tarde cunhou o termo 'hidropónico'. Gericke criou uma das primeiras oportunidades de relações públicas para hidroponia quando apareceu com plantas de tomate cultivadas verticalmente na sua nova cultura de soluções nutricionais. As estufas vivenciaram um impulso significativo nos anos 50. O plástico abriu caminho para muitas inovações, permitindo a introdução de sistemas de rega gota a gota, construção de filtros, reservatórios de água, entre outros sistemas [5].

Nos anos 60, surge a evolução de tipos mais específicos de sistemas hidropónicos. O primeiro a surgir foi a *Nutrient Film Technique*¹, concebida em Inglaterra por Allan Cooper.

¹ <https://www.thespruce.com/hydroponic-gardens-nutrient-film-technique-1939220>

Na história da hidroponia, observam-se milénios de inovações e progressos. A hidroponia literalmente revolucionou o processo de cultivo ao longo da história, trazendo prosperidade como resultado. Estamos numa era de rápido crescimento, portanto pode-se esperar que a hidroponia acompanhe a colonização espacial, como é possível observar em registos de pesquisas aeroespaciais. Certamente veremos um influxo de métodos hidropónicos mais novos e acessíveis.

Devido à alta eficiência e aos benefícios ambientais da hidroponia, já é analisada para resolver problemas como o da agricultura tradicional em solo. Podemos olhar para a história da hidroponia e concluir que tem sido um marco da inovação humana. Isso sempre continuará, assim como o impulso da natureza humana para melhorar, inovar e encontrar novas maneiras de trazer prosperidade para as sociedades [6].

2.2 Hidroponia em Portugal

A hidroponia em Portugal tem vindo a conquistar novos adeptos, por ser um método limpo, extremamente simples, produção económica e de baixo custo de manutenção.

Portugal tem um clima favorável e propício ao desenvolvimento da agricultura Hidropónica que permite um custo de produção menor devido á maior facilidade de manter o ambiente ideal para as culturas. Também o fácil acesso a recursos hídricos potencia o desenvolvimento desta atividade

Atualmente, em Portugal, a alface mantém-se como o vegetal mais cultivado em sistemas hidropónicos. Contudo, pode observar-se o cultivo de muitas outras espécies nomeadamente rúcula, agrião, salsa, coentros, manjerição, orégãos, tomate, tomate cherry, morangos, curgetes, couve-chinesa, pimentos, melão, entre muitas outras espécies.

A pratica desta atividade produz inúmeros benefícios no país, tais como:

- Aumento da produtividade, permitindo uma maior rapidez na colheita da plantação devido à redução dos ciclos vegetativos;
- Garantia de uniformidade das culturas e um maior controlo produtivo;

- A independência do solo permite o cultivo próximo ao consumidor final, minimizando os custos com logística e transporte.

Nos últimos anos, o Governo tem apostado na criação de sistemas de apoio aos investimentos de jovens agricultores o que despoletou uma maior curiosidade e iniciativa pelo conceito de hidroponia, pelo que seguidamente se destacar-se-ão três projetos de sucesso em Portugal.

2.2.1 Jungle Greens

A cerca de 10 metros da porta principal do Jumbo em Sintra situa-se a primeira unidade de produção da Jungle Greens² onde são cultivadas ervas aromáticas e microgreens, vendidos nesse mesmo supermercado. Recentemente a ideia da agricultura em recinto fechado começou a ganhar destaque, devido à importância de reduzir a pegada de carbono e à crescente popularidade da produção junto do consumidor.

O ambiente de cultivo da empresa permite um alto nível de rastreabilidade, por isso desejam tornar todo o processo, da semente ao prato, totalmente transparente e rastreável para os consumidores. Com as suas embalagens sem plástico, a Jungle Greens contribui ainda para reduzir a poluição relacionada às embalagens de plástico.

2.2.2 Hidroflora

A Hidroflora³ é uma empresa de produção e comercialização de produtos em hidroponia, que nasceu pelas mãos de Vasco Barbosa e Flora Vieira. Na Hidroflora, as plantas são colocadas em espuma fenólica, em canaletes por onde circula uma solução nutritiva em espaços de tempo pré-determinados, composta de água potável e nutrientes dissolvidos em quantidades que satisfazem integralmente as necessidades de cada espécie vegetal cultivada, realçando que os nutrientes que a planta precisa para o seu desenvolvimento e produção são fornecidos somente por água enriquecida (solução nutritiva) com os elementos necessários: azoto, potássio, fósforo, magnésio, entre outros.

² <https://observador.pt/2019/04/09/agricultura-vertical-contentores-estufa-e-o-futuro-como-a-jungle-greens-quer-salvar-o-mundo-com-vegetais-e-ervas-aromaticas/>

³ <https://www.facebook.com/hidrofloraptl/>

Por outro lado, a solução nutritiva é alvo de um controlo rigoroso, que inclui a monitorização do pH e da concentração das soluções nutritivas de sais minerais, permitindo que a planta se desenvolva mais forte e saudável, sem deficiências nutricionais e com uma qualidade e sabor “melhor que os produtos produzidos através das práticas tradicionais”.

A Hidroflora começou com o cultivo de nabças, mas atualmente também produz alface (frisada, roxa e multi-folhas), agrião e alho francês. O escoamento do produto é feito através da venda direta à restauração e a pequenas superfícies comerciais.

2.2.3 Horta Grande

A Horta Grande⁴ nasce em 1896, na cidade de Abrantes. Esta quinta de origem familiar dedicou-se desde sempre à produção de frutas e legumes, porém dada a excelente qualidade e abundância das suas águas, especializou-se nos últimos anos na produção de morangos em sistema hidropónico. Este projeto é gerido por Inês Pereira da Silva que complementou a sua licenciatura na Universidade Nova de Lisboa com o curso de jovem agricultor e, pelos seus dois irmãos, Afonso e Tomás Pereira da Silva, estudantes na Escola Superior Agrária de Santarém

A estufa da “Horta Grande” destaca-se por ter 2 níveis e 1 subnível suspensos, numa área de meio hectare. Tem capacidade para 150 mil pés de morangueiros, o que no sistema extensivo de produção seriam necessários cerca de três hectares para o mesmo efeito.

Para aproveitamento dos morangos mais maduros, dedicam-se atualmente à realização de compotas de morangos esmagados, xarope de morango e vinagrete de morangos. Em alturas festivas, fazem uns cabazes especiais para a data, como é o caso do Dia dos Namorados em que junto com os morangos vem uma caixinha com chocolate líquido.

Os gelados de algumas geladarias mais tradicionais de Lisboa são também feitos com estes morangos em fresco. Ainda assim, 60% da produção tem como destino exportação, nomeadamente para Espanha.

⁴ <https://www.facebook.com/Horta-Grande-1433283516886668/>

2.3 Tipos de Sistemas Hidropónicos

Segundo [6] existem seis tipos de sistemas hidropónicos, o *Drip System*, *Ebb & Flow*, *N.F.T.*, *Water Culture*, *Aeroponics* e *Wick*, que são abordados seguidamente nos próximos subcapítulos.

As raízes das plantas precisam de três elementos: água/humidade, nutrientes e oxigénio. O que diferencia os seis tipos de sistemas hidropónicos é apenas a forma como os mesmos distribuem estes três elementos essenciais às raízes das plantas [7].

2.3.1 *Drip System* - Sistema Hidropónico por Gota

O *Drip System* - (Figura 1) sistema hidropónico por gota é um dos tipos de sistemas hidropónicos mais amplamente utilizados em todo o mundo, tanto para os agricultores domésticos quanto para os comerciais. Um dos principais motivos prende-se pelo facto de ser um facilmente exequível pois precisa de poucos componentes estruturais para ser colocado em prática, ainda assim é um tipo de sistema hidropónico muito versátil e eficaz.

Neste sistema apenas se goteja a solução nutritiva nas raízes das plantas para mantê-las húmidas. São especialmente úteis para plantas maiores que ocupam muito espaço nas raízes, pois não necessita de grande volume de água para inundar o sistema, e as linhas de gotejamento são fáceis de percorrer em espaços mais longos.

O funcionamento de um sistema hidropónico por gota é relativamente simples. A água (solução nutritiva) é bombeada do reservatório através da tubagem para o topo do meio de cultivo (onde estão as raízes das plantas). A solução nutritiva é absorvida pelas raízes e pelo meio de cultivo até ao fundo do recipiente de cultivo. A partir daí, a solução nutritiva flui através de uma abertura, e a gravidade permite que a solução nutriente flua até ao reservatório.

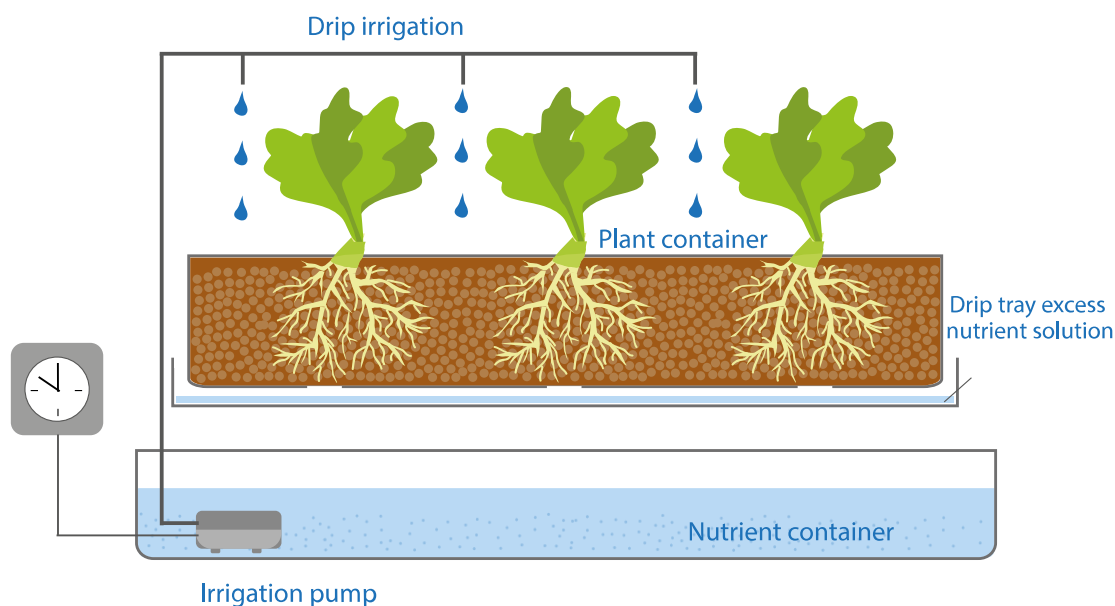


Figura 1 - Drip System⁵

2.3.2 Ebb & Flow - Sistema de Inundação e Drenagem

Os sistemas de inundação e drenagem (Figura 2) são muito populares entre os produtores hidropônicos domésticos por várias razões. Além de serem fáceis de construir para qualquer um, podem usar-se quase todos os materiais disponíveis para construí-los, evitando gastar muito dinheiro para cultivar plantas hidroponicamente.

Além disso, eles podem ser construídos para caber em qualquer espaço disponível que se possa ter (em ambientes internos ou externos), e não há limite para a imaginação ao projetá-los. Além disso, as plantas crescem muito bem neste tipo de sistema, que funciona inundando simplesmente o sistema radicular das plantas com solução nutritiva. Apenas periodicamente e não continuamente.

A parte principal deste sistema é composta pelos recipientes em que as plantas crescem. Pode ser apenas uma planta ou várias em série. Um temporizador liga a bomba e a solução nutritiva é bombeada através da tubagem do reservatório para a parte principal do sistema usando uma bomba submersível.

A solução nutritiva continua a encher o sistema até atingir a altura predefinida do tubo de transbordo, ajustado à altura das raízes das plantas.

⁵ <https://fungiweed.com/cultivo-cannabico-hidroponico/>

Quando a solução nutritiva do sistema atinge a altura do tubo de transbordo, ela volta ao reservatório, onde circula novamente pelo sistema. O tubo de transbordo define a altura do nível da água no sistema, garantindo que a água não derrama pela parte superior do sistema enquanto a bomba estiver ligada. Quando a bomba é desligada, a água volta para o reservatório através da bomba (drenando completamente o sistema).

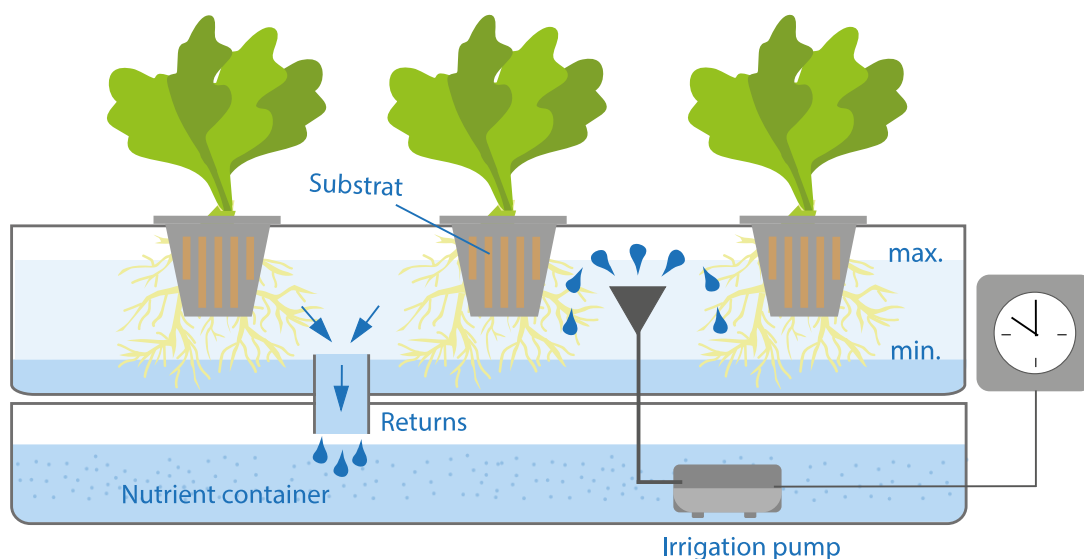


Figura 2 - Ebb & Flow⁶

2.3.3 N.F.T. - Sistema de Técnica de Filme Nutriente

Como representado na (Figura 3), este sistema é mais adequado para o cultivo de plantas de crescimento rápido como diferentes tipos de alface. Embora existam muitas maneiras diferentes de projetar este sistema, todos eles apresentam a mesma característica, uma solução nutritiva muito superficial em cascata através da tubagem, onde as raízes das plantas entram em contato com a água e podem absorver os nutrientes. A principal desvantagem é que as plantas são muito sensíveis a interrupções no fluxo de água devido a quedas de energia. As plantas começarão a murchar muito rapidamente sempre que a água parar de fluir pelo sistema.

Assim, a solução nutritiva é bombeada do reservatório, para um coletor que conecta o tubo maior a um tubo menor. Uma camada fina (filme) da solução nutritiva flui através de cada canal com plantas, passando por cada planta e molhando as raízes no fundo. A

⁶ <https://fungiweed.com/cultivo-cannabico-hidroponico/>

solução nutritiva flui de um lado para o outro, porque o canal é ligeiramente inclinado, de modo que a água flui canal abaixo. As raízes das plantas ficam penduradas no fundo do tubo, onde obtêm nutrientes da solução nutritiva que flui pelo canal. O excesso de solução nutritiva que flui da extremidade inferior de cada um dos canais é drenada para outro tubo e guiado de volta ao reservatório, onde é circulada pelo sistema novamente.

Embora a solução nutritiva que flui pelos canais seja muito superficial, toda a massa das raízes das plantas permanece húmida, podendo absorver a humidade do lado de fora das raízes, bem como a humidade mantida dentro do tubo. As raízes que estão suspensas entre a base da planta e o nível da água no canal não só têm humidade para aceder, mas também são capazes de obter bastante oxigénio do ar que os cerca dentro do tubo.

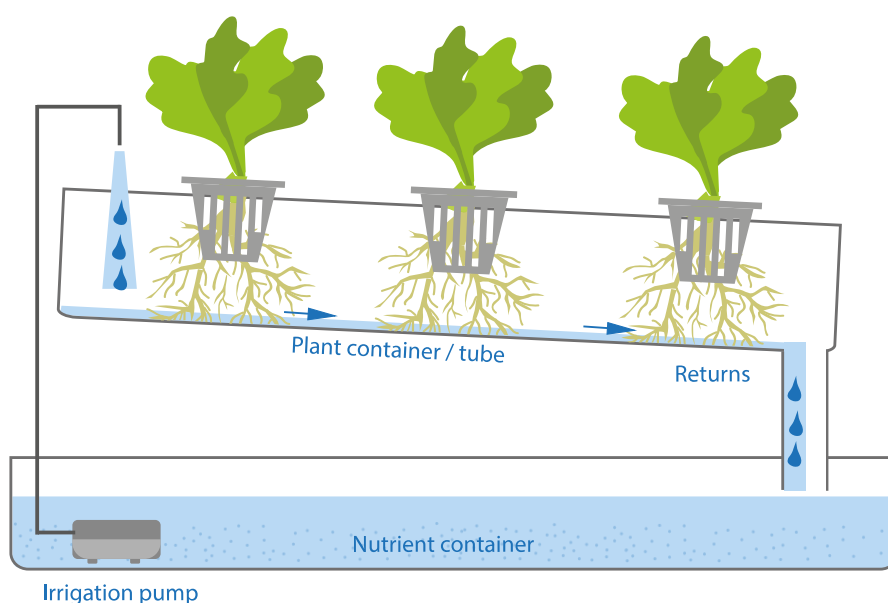


Figura 3 - N.F.T.⁷

2.3.4 Water Culture - Sistema de Cultura em Água

Neste sistema (Figura 4) a planta é suspensa em cestos acima da solução nutritiva no reservatório. As raízes pendem diretamente na solução nutritiva onde estão submersas. Estas não sufocam pois recebem o ar e o oxigénio de que precisam devido às bolhas de ar que sobem através da solução nutritiva, bem como ao oxigénio dissolvido na água. As bolhas devem subir e fazer contato direto com as raízes, para serem mais eficazes para as

⁷ <https://fungiweed.com/cultivo-cannabico-hidroponico/>

plantas. Existem duas formas de fornecer aeração⁸ e oxigénio dissolvido à solução nutritiva.

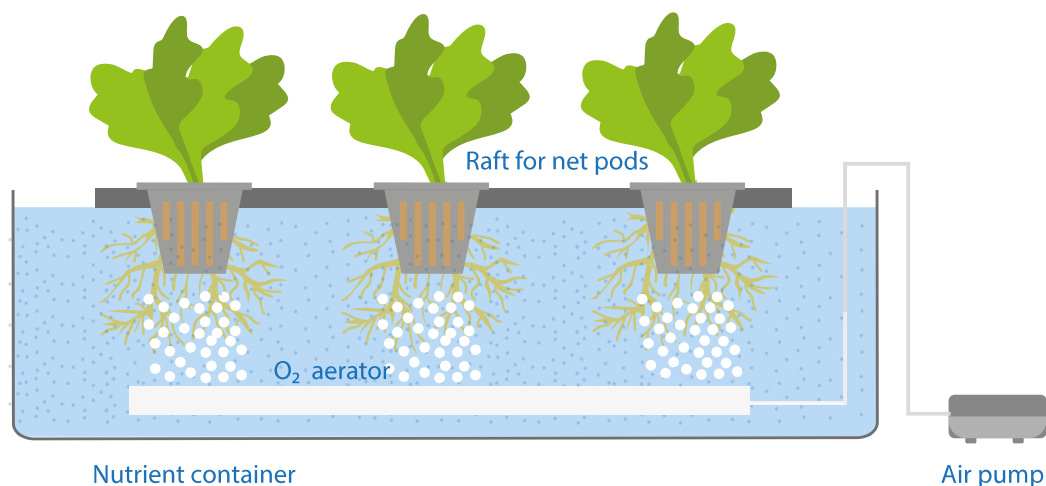


Figura 4 - Water Culture⁹

2.3.5 Aeroponics - Sistema Aeropónico

Umas das vantagens deste sistema é que normalmente usa pouco ou nenhum meio de cultivo. As raízes obtêm o máximo de oxigénio e, como resultado, as plantas crescem mais rápido. Os sistemas aeropónicos (Figura 5) também usam menos água do que qualquer outro tipo de sistema e a colheita também é geralmente mais fácil, especialmente para as raízes.

No entanto, existem também desvantagens pois além de ser um pouco mais caro de construir, os aspersores podem entupir devido ao acumular de elementos minerais dissolvidos na solução nutritiva. Também porque as raízes das plantas estão suspensas no ar, são muito mais vulneráveis à secagem se houver alguma interrupção no ciclo de rega. Portanto, mesmo qualquer falta de energia temporária pode fazer com que as plantas morram muito mais rapidamente do que em qualquer outro tipo de sistema.

⁸ Refere-se à zona do solo onde os interstícios são ocupados pelo ar depois da água de infiltração se ter escoado. A aeração dum solo é muito importante porque sem o fornecimento de oxigénio as raízes não podem captar os iões e a água de que as plantas necessitam ([https://www.infopedia.pt/\\$aeracao](https://www.infopedia.pt/$aeracao)).

⁹ <https://fungiweed.com/cultivo-cannabico-hidroponico/>

No sistema Aeropónico as plantas são suspensas em pequenos cestos que se comprimem ao redor do caule das plantas. Esses cestos encaixam-se em pequenos orifícios na parte superior da câmara de cultivo, onde as raízes ficam penduradas, sendo pulverizadas com solução nutritiva através dos expressores em ciclos curtos regulares. Os ciclos regulares de rega mantêm as raízes húmidas fornecendo os nutrientes que as plantas precisam para crescer. A câmara de crescimento em que as raízes estão deve ser à prova de luz e protegido contra possíveis ventos exteriores. No entanto é necessário permitir a entrada de ar fresco para que as raízes possam receber bastante oxigénio.

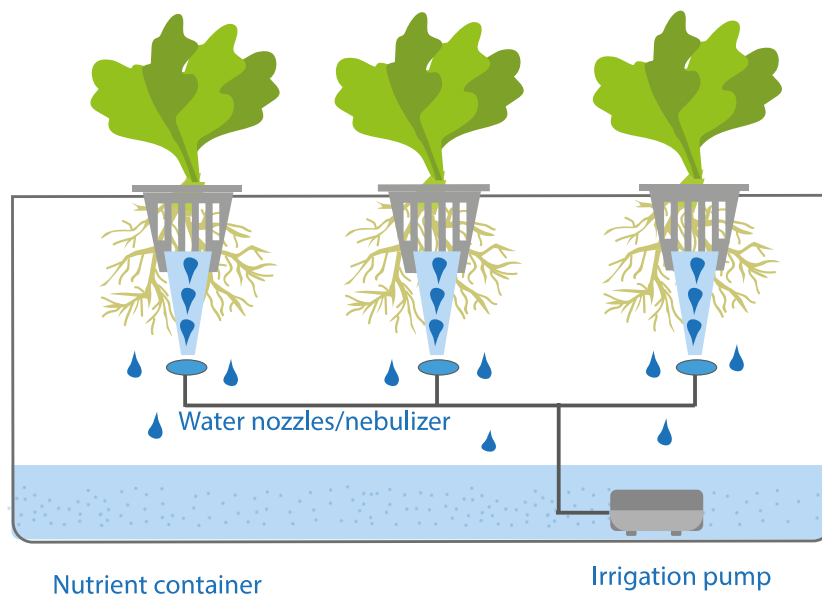


Figura 5 - Aeroponics¹⁰

2.3.6 Wick - Sistema de Pavio

O sistema de pavio (Figura 6) é um dos chamados sistemas passivos, um sistema que não tem partes móveis. Não existe fluxo de água ou nenhuma parte que se desloque no decorrer da alimentação das plantas, sendo que os nutrientes são atraídos até às raízes através de um pavio ou mecha de tecido. Praticamente qualquer tipo de tecido absorvente serve, podendo utilizar-se, por exemplo, pavio de vela ou de lanterna a petróleo. Este sistema é muito simples e ocupa pouco espaço. São essenciais dois contentores, um para albergar uma solução com nutrientes adicionados e outro, no topo do contentor mais pequeno, com material de cultivo.

¹⁰ <https://fungiweed.com/cultivo-cannabico-hidroponico/>

Fazem-se pequenos furos para passar os pavios que devem estar mais que 60% dentro de água. No contentor de cima, enquanto está vazio, fazem-se pequenos rolos do pavio e espera-se até ficarem molhados. O passo seguinte é colocar as plantas no sítio de cultura, limpando as raízes de qualquer vestígio de terra. Aconselha-se a limpeza em água corrente mexendo o mínimo possível na raiz, e que seja de noite, sem luzes fortes. Coloca-se cada planta por cima de uma mecha, e enche-se à volta com o material de cultivo até a planta se manter direita. Aliás, o objetivo deste material de cultivo é de facto o de segurar as plantas e bloquear a luz. O único problema deste sistema é a planta pedir mais nutrientes que aqueles que a mecha consegue fornecer. As plantas com maiores necessidades ou de maior porte deverão ter mais que uma mecha ou serem mudadas quando começarem a mostrar sinais de que não se estão a “alimentar” o suficiente [6].

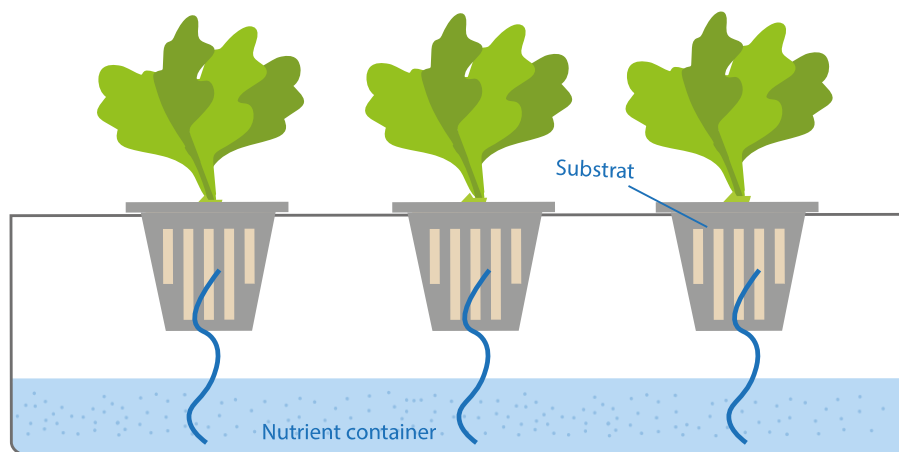


Figura 6 - Wick¹¹

2.3.7 Comparação dos Sistemas Hidropónicos

Para resumir as principais vantagens e desvantagens dos sistemas hidropónicos foi construído a Tabela 1.

¹¹ <https://fungiweed.com/cultivo-cannabico-hidroponico/>

Tabela 1 - Comparação entre sistemas hidropónicos

	VANTAGENS	DESVANTAGENS
DRIP	<ul style="list-style-type: none"> • Custo de construção baixo. • Possibilidade de alimentação constante de solução ou sistema cronometrado. • Reutilização da solução nutritiva. • Inexistência de acumulação de sais. 	<ul style="list-style-type: none"> • Necessidade de utilização de solo. • Complexidade na construção. • Absorção mais lenta.
EBB & FLOW	<ul style="list-style-type: none"> • Reutilização da solução nutritiva. • De fácil construção e manutenção. • Custo de construção baixo. • Utilização de qualquer tipo de meio de cultivo. 	<ul style="list-style-type: none"> • Acumulação de sais e minerais no fundo do meio de cultivo, que poderá gerar uma alteração nutritiva. • Possível necessidade de ajuste de altura dos vasos no estágio inicial, para as raízes alcançarem a solução nutritiva.
N.F.T.	<ul style="list-style-type: none"> • Custo de construção baixo. • O movimento constante da água impede a sua estagnação. • Diversidade de meios de cultivo para segurar a planta. • Reutilização da solução nutritiva. 	<ul style="list-style-type: none"> • Requer controlo de pH. • Dependente do funcionamento da bomba. • Possível necessidade de ajuste de altura dos vasos no estágio inicial, para as raízes alcançarem a solução nutritiva.
WATER CULTURE	<ul style="list-style-type: none"> • Tamanhos flexíveis do local da planta. • Reutilização de água. • Custo de construção baixo. 	<ul style="list-style-type: none"> • Propenso a doenças de raiz. • Requer controlo de pH. • Possível bloqueio na irrigação devido ao crescimento das raízes.
AEROPONICS	<ul style="list-style-type: none"> • Grande necessidade de oxigénio na água. • Reutilização da solução nutritiva. • Maior absorção da solução nutritiva. • Crescimento acelerado. 	<ul style="list-style-type: none"> • Requer controlo de pH. • Necessidade de ciclos de rega muito precisos e constantes. • Propenso a doenças de raiz. • Dependente do funcionamento da bomba.
WICK	<ul style="list-style-type: none"> • Adequado para pequenos espaços. • De fácil construção e manutenção. • Custo de construção baixo. • Irrigação constante e adequada. 	<ul style="list-style-type: none"> • Apenas apropriado para plantas de irrigação constante. • Insuficiência nutritiva para plantas maiores e de crescimento mais rápido. • Propenso a doenças de raiz.

2.4 Análise da solução nutritiva por medição da condutividade elétrica

Dado que não se faz o uso do solo no cultivo de plantas, na hidroponia, é imprescindível ter uma solução nutritiva equilibrada que contém, além da água, todos os nutrientes indispensáveis para o crescimento da planta. As raízes ficam emergidas nessa solução e dali elas retiram tudo o que precisam para crescerem saudáveis e fortes.

Uma solução nutritiva equilibrada depende de fatores como a temperatura da solução, o nível de oxigénio, o pH e a quantidade de nutrientes. Dentre destes, manter a uma quantidade de nutrientes balanceada é tão importante quanto qualquer outro passo. Para isso usa-se a condutividade elétrica pois esta determina a quantidade de iões, e

consequentemente nutrientes, na solução nutritiva e, quanto mais íões, maior a condutividade elétrica. Ao adicionar nutrientes à água, transformando ela em uma solução nutritiva, ocorrem quebras das moléculas dos nutrientes e a liberação de íões. Desse modo, a adição de íões e suas cargas elétricas faz a CE da água subir. [8]

A Condutividade Elétrica pode ser expressa em: Siemens por metro (S/m, Sistema Internacional), porém a maior parte dos aparelhos medidores utilizam as medições em $\mu\text{S}/\text{cm}$ ou mS/cm . As medidas ideais estão na faixa de 1,5 a 3,5 milisiemens/cm, todavia esses valores devem variar de acordo com o cultivo, bem como com as condições climáticas, como tal, é indispensável realizar o teste de Condutividade Elétrica regularmente para controlá-la. Para tal utiliza-se um aparelho medidor denominado Condutivímetro. Este é o equipamento mais eficiente e produz os melhores resultados, sendo que ajuda a controlar a CE, indica e compensa automaticamente a temperatura [9].

Para o condutivímetro funcionar corretamente, é necessário fazer a sua calibração antes e depois de o utiliza. Os elétrodos polarizados ou sujos devem ser lavados para remover possíveis erros de leitura. Água quente com detergente líquido é aconselhado para o trabalho de limpeza, todavia para a remoção de qualquer matéria orgânica recomenda-se o uso de acetona e, para remover algas, bacterianas ou fungos as soluções com cloro podem ajudar. Existem comercialmente soluções de referência para calibração do Condutivímetro, simplificando o processo.

É importante realçar que a relação entre CE e temperatura é direta, a cada 1°C de aumento de temperatura a CE aumenta 2%. Não obstante, o teste de CE é ótimo para entender o que a planta necessita, por isso é preciso observar e realizar diariamente para adaptar a quantidade de nutrientes que cada planta necessita.

No projeto implementado, a opção escolhida passa por um composto de nutrientes básico, constituído por duas partes (A e B). Estão separados em duas partes porque alguns elementos da parte A e B se adicionados em conjunto formam combinações insolúveis que a planta não consegue absorver. Por isso deve-se primeiro adicionar a parte A e depois a B.

2.5 Controle de pH da solução nutritiva

Para o alcance de ótimos resultados em sistemas hidropônicos, com plantas saudáveis e vigorosas, é crucial monitorizar regularmente o pH da solução nutritiva.

O pH é a medida do nível de alcalinidade e acidez de uma solução de água. Este é medido numa escala de 1 a 14, onde a metade inferior é ácida e a metade superior é alcalina. Sendo que água perfeitamente pura tem pH 7.

Para a hidroponia, o pH ideal ou padrão, está na faixa entre 6,0 e 6,5. Esta faixa de pH é a ideal para uma melhor absorção de micro e macronutrientes pelas raízes das plantas. Nesta faixa de pH há maior absorção de macronutrientes como Nitrogénio, Fósforo e Potássio, com isso as plantas tem uma condição ótima para se desenvolverem. Se o nível de pH da solução não estiver nesta faixa ótima as plantas não irão se desenvolver e se o pH estiver em níveis extremos as plantas poderão morrer.

Existem dois métodos simples para medir o pH da solução nutritiva. Para iniciantes o método mais simples é o uso de fitas de papel indicadoras de pH. Estas fitas são revestidas com um corante especial, que, quando mergulhado na solução irá indicar uma cor. Deverá comparar essa cor com uma cartela de cores com o pH equivalente de cada cor. Esse método, no entanto, não é o mais preciso.

Os produtores com projetos mais avançados usam um medidor de pH digital. Esses medidores são de simples operação, basta imergir a sonda na solução nutritiva e terá o valor do pH. Estes medidores de pH digital estão disponíveis em inúmeros modelos e em ampla gama de preços.

Assim, é crucial conhecer as propriedades químicas da água a ser utilizada para hidroponia, bem como as características do adubo quanto à influência do pH final da solução para um melhor manuseamento dessa variável a ser controlada diariamente. Neste caso, a análise das concentrações de carbonato, bicarbonato, Ca, Mg, S, Fe, Na, NO₃, entre outros elementos presentes na água é um passo inicial e fundamental para a implementação de um projeto de cultivo hidropónico (GroHo Hidroponia, 2021)

3 Elaboração de um novo sistema hidropónico

O objetivo fulcral deste trabalho visa a implementação, numa pequena estufa, de um circuito que controla um reservatório com solução nutritiva e fazê-la chegar às plantas que estão numa calha de crescimento. Este processo realiza-se, tendo em conta, a quantidade de nutrientes e o pH da solução para um crescimento sustentável. Foi implementada uma base de dados e página web para que exista uma interface de comunicação entre o utilizador e a estufa.

Outro objetivo que se pretendeu alcançar foca a versatilidade deste projeto, pois com pequenas alterações, o sistema pode ser aplicado a vários tipos de plantas em cultivo.

A elaboração da estufa foi um processo constituído em duas fases. Primeiro a escolha e instalação de todo o *hardware* usado no projeto e, em segundo, o desenvolver de toda a programação para que se verifique um crescimento sustentável das plantas.

Por forma a planificar o projeto da estufa, foi estudada uma estrutura de suporte para as plantas, assente em dispositivos de controlo e gestão, sensores e atuadores, como é apresentado na Figura 7.

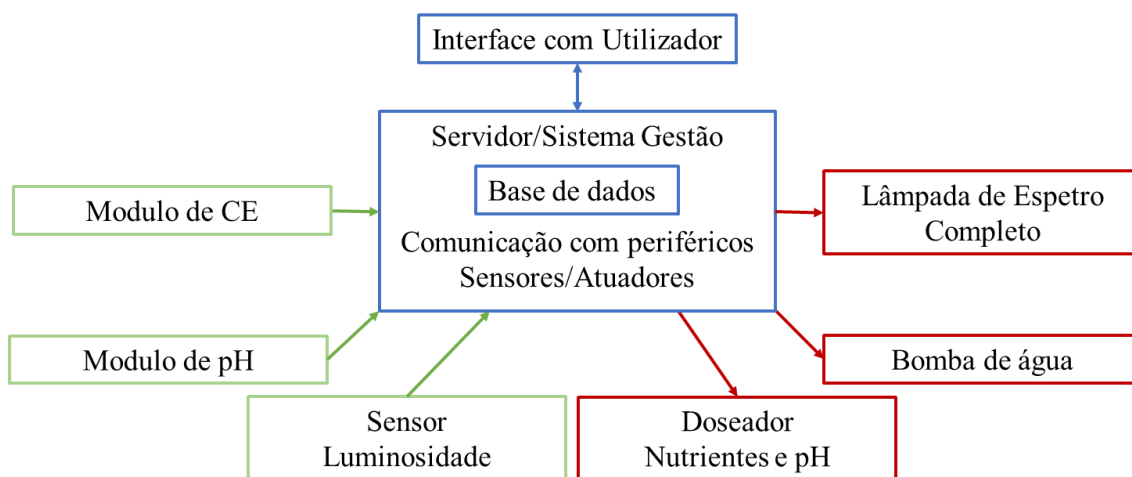


Figura 7 - Diagrama funcional do sistema

Todo este projeto tem como base um servidor que funciona como o cérebro do sistema, é aqui que além da página web de interface do utilizador também se encontra, o *script* que permite a troca de dados com o Arduino que gere a leitura de sensores e controlo de

atuadores. Os sensores vão obtendo informação sobre o estado das variáveis no interior da estufa, sendo possível atuar de acordo com o seu estado.

Todo este projeto tem como base um sistema servidor que funciona como o cérebro do sistema. É aqui que além de uma interface web para o utilizador também se deve encontrar a um sistema que faça a leitura de sensores e o controlo de atuadores. Os sensores vão obtendo informação sobre o estado das variáveis no interior da estufa, sendo possível atuar de acordo com o seu estado.

Os sensores que foram considerados neste projeto são 3:

Módulo de Condutividade Elétrica – a função principal é medir a condutividade elétrica entre dois polos que ficam submersos, através deste sensor é possível fazer uma leitura da quantidade de sais e nutrientes que estão dissolvidos na solução nutritiva. líquido.

Módulo de pH – É um sensor que ao ser introduzido na solução, vai gerar um sinal em tensão entre 0 e 5V que depois de calibrado deve informar o valor de pH com referência em 0 (ácido), 7 (neutro) e 14 (básico).

Sensor de Luminosidade – Módulo com LDR (*Light Dependent Resistor*) para determinar a quantidade de luz natural e dar a informação ao controlador do sistema quando deverá ou não ativar a luz artificial.

Estes sensores são os responsáveis pela leitura das condições da estufa em tempo real, tendo em consideração a solução nutritiva existente no tanque e a luz solar que as plantas estão a receber. Após este processo, dá-se início ao ajuste da solução nutritiva, para isso são usados motores para adicionar nutrientes ou ácido ao tanque e por fim é analisada a luminosidade exterior para atuar ou não a lâmpada como é descrito mais à frente.

Controlador de Nutrientes e pH – De uma forma relativamente precisa e económica, foi estudado o uso de bombas peristálticas que usam a sucção por vácuo e o tempo de rotação do motor para adicionar pequenas quantidades de nutrientes à solução existente no tanque, como será explicado mais à frente.

Bomba de Água – Foi projetado o uso de pequenas bombas de aquário, uma para fazer a solução ser transferida do tanque para a calha de crescimento e para manter a solução

do tanque em movimento e assim, além de misturar melhor todos os da solução nutritiva, impedir que os mesmo fiquem estagnados no fundo.

Iluminação Artificial– Para este efeito foi analisada a utilização de uma lâmpada com espectro de cor 660nm e 450nm, isto é, ao contrário de uma lâmpada convencional em que o espectro pode variar entre os 400nm e os 700nm, esta usa apenas o espectro preciso para o crescimento da planta. Neste caso 660nm (cor vermelha) que promove a floração e combinação de 660nm com 450nm (cor azul) que promove o crescimento. Esta última combinação resulta numa cor em tons de magenta.

Toda a análise e comando do sistema é feita através da Raspberry Pi (responsável pela análise de dados e interface com o utilizador) e pelo Arduino Micro (responsável pelo controlo dos periféricos acima descritos).

3.1 Componentes de Hardware integrados no sistema Hidropónico

Para a execução deste projeto foi tido em conta sobretudo dois aspetos, o económico e a versatilidade para possíveis alterações. Para isso foi realizado uma pesquisa sobre os produtos existentes no mercado e analisados os que de alguma forma poderiam ser possíveis de usar em várias situações diferentes e ao mesmo tempo que fosse economicamente viável para um projeto que se requer de baixo custo.

Salientado a Raspberry Pi onde assenta tudo o que diz respeito a Servidor Web assim como o script de Python responsável pelo controlo de toda a operação e base de dados, o Arduino Micro, que faz a ligação entre o sistema de comando com os atuadores e sensores, o sensor de PH de onde se obtém a leitura do devido PH e temperatura, o sensor de EC que consiste em fazer uma análise ao nível de nutrientes no depósito e a lâmpada de espectro completo que serve para ajudar o crescimento em dias de menor luminosidade.

Tendo em consideração o diagrama anterior (Figura 7), a implementação do projeto ficou da seguinte forma (Figura 8)

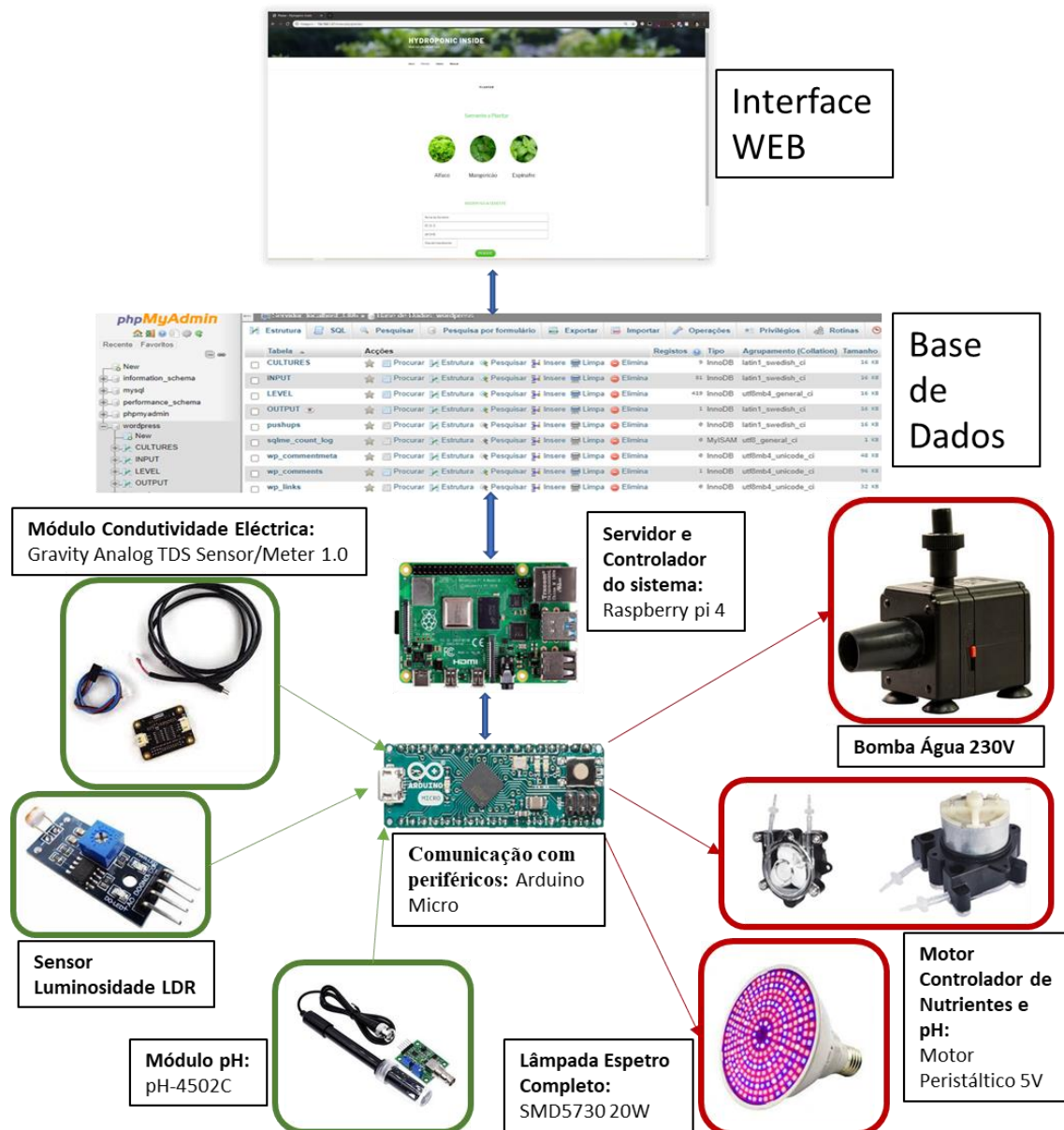


Figura 8 - Diagrama Ilustrativo do Sistema

Estes são os componentes usados para o controlo de todo o sistema e que são descritos nos pontos que se seguem.

3.1.1 Sistema Arduíno Micro

O Arduíno é uma plataforma de desenvolvimento, tem com elemento principal o microcontrolador ATmel (ATmega32U4) e suporte para entradas e saídas. É um sistema relativamente barato, bastante flexível e de uso intuitivo, razões pelas quais levaram a optar por este sistema.

Para o desenvolvimento de software pode ser usada a aplicação Arduino IDE¹² (*Arduino Integrated Development Environment*), isto é, a aplicação onde se escreve, edita, corrige e faz o envio para o Arduino de todo o código máquina gerado pelo IDE para gerar as funções desejadas. É através desta ferramenta que são escritas as funções e enviadas para o microcontrolador. Tem como linguagens de programação C, C++ e Java, mas que neste caso específico foi usado o C/C++.

Numa fase inicial utilizou-se o modelo MEGA (Figura 9 a)) por ser o microcontrolador disponível para ensaio, no entanto, no decorrer do projeto fez-se a alteração para o Arduino Micro (Figura 9 b)), uma vez que apresentava caraterísticas suficientes para o processo face ao número de entradas e saída requeridas fazendo os ajustes necessários para compatibilidade entre o processador ATmega2560 (Mega) e o ATmega32U4 (Micro).



Figura 9 - a) Arduino Mega e b) Arduino Micro

As principais características do Arduino Micro (escolhido para ensaio final) são apresentadas na Tabela 2.

Tabela 2 - Características Arduino Micro

- Microcontroller: ATmega32u4	- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V	- Input Voltage (limits): 6-20V
- Digital I/O Pins: 20	- PWM Channels: 7
- Analog Input Channels: 12	- DC Current per I/O Pin: 40 mA
- DC Current for 3.3V Pin: 50 mA	- SRAM: 2.5 KB (ATmega32u4)
- Clock Speed: 16 MHz	- EEPROM: 1 KB (ATmega32u4)
- Flash Memory: 32 KB (ATmega32u4) of which 4 KB used by bootloader	

¹² <https://www.arduino.cc/en/software/>

3.1.2 Raspberry Pi

A placa *Raspberry Pi*¹³ usada no projeto é do modelo 4 com 2Gb de RAM e que usa o CPU *Broadcom BCM2711*, *Quad core Cortex-A72* (ARM v8) 64-bit SoC @ 1.5GHz (Tabela 3).

A *Raspberry Pi* (RPi) (Figura 10) nasceu da necessidade de levar a programação para as escolas tornando-se um gadget muito versátil que pode fazer quase tudo com o *software* certo¹⁴.

É uma excelente opção para criar sistemas autónomos portáteis (devido à sua reduzida dimensão). A facilidade que apresenta em se conectar à *Internet of Things*¹⁵ faz com que tenha uma enorme flexibilidade, tanto a nível de *hardware*, como a nível de *software*, estando disponível uma grande quantidade de *software* gratuito e de vasta aplicação.

O sistema operativo desta máquina pode ser alterado com grande facilidade sendo que o mais comum é usar como base o *Linux* e neste caso específico foi adotado o *Raspbian*¹⁶. É a opção usada para dar também uma experiência gráfica ao utilizador sem que sejam consumidos grandes recursos.



Figura 10 - Raspberry Pi

¹³<https://www.raspberrypi.org/>

¹⁴ <https://www.raspberrypiportugal.pt/os-melhores-sistemas-operativos-raspberry-pi/>

¹⁵ *Internet of Things* é um conceito que se refere à interconexão digital de objetos cotidianos com a internet, conexão dos objetos mais do que das pessoas. <https://bit.ly/3hY1w5b>

¹⁶<https://www.raspberrypi.org/software/operating-systems/>

Tabela 3 - Características Raspberry Pi

- CPU: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz	- RAM: 2GB, LPDDR4-3200 SDRAM
- Conectividade: 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE Gigabit Ethernet	- Portas: 2 USB 3.0 ports; 2 USB 2.0 ports.
- Display: 2 × micro-HDMI ports (up to 4kp60 supported), 2-lane MIPI DSI display port	- Micro-SD card slot for loading operating system and data storage
- Audio: 4-pole stereo audio and composite video port	- Alimentação: over Ethernet (PoE) enabled (requires separate PoE HAT), 5V DC via USB-C connector (minimum 3A)

3.1.3 Módulo de Relés

Para proteger o Arduino, ligar atuadores e as pontas de medição de condutividade foi usado um módulo de 8 relés de 5V (Figura 11). O módulo de relés foi usado para ativar uma bomba submersível (230V), uma lâmpada de espectro completo (230V), uma válvula solenoide (230V), três motores peristálticos (5V) e por fim, foram usados dois relés para fazer a alteração das pontas de provas como será explicado no sensor de EC. Para ter uma melhor percepção dos relés ativos, este módulo possui LEDs indicadores do estado de cada relé. Esta placa é alimentada de forma independente do Arduino com 5V.

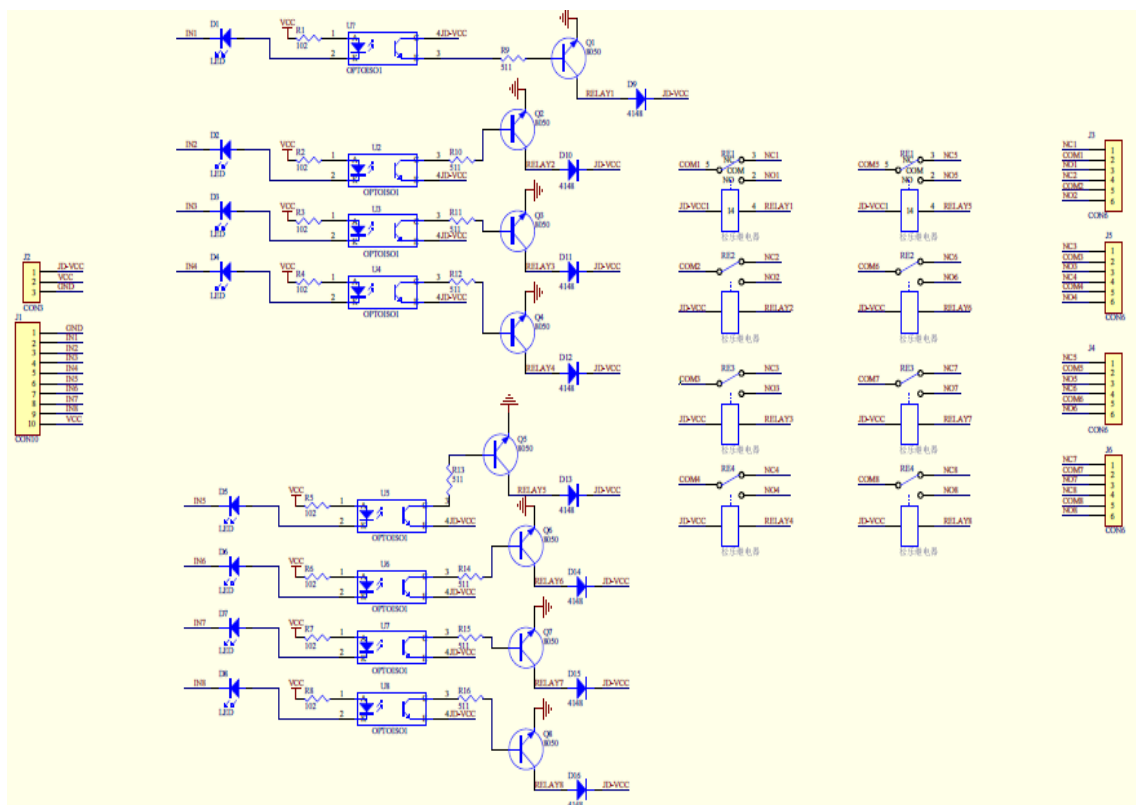


Figura 11 - Módulo Relés Esquemático

As principais características do módulo de relés são apresentadas na Tabela 4.

Tabela 4 - Características Módulo de Relés

- Canais Saída: 8	- Tensão Operação: 5 Vdc
- Corrente Operação: 480 ma	- Tensão Switch: 250 Vac at 10 Amps, 30 Vdc at 10 A

3.1.4 Fonte de Alimentação

Para alimentar todo o sistema, foi usada uma fonte de alimentação tipo computador. Com vários tipos de saídas, neste caso foram apenas usadas as de 5V CC e de 230V CA (Tabela 5).

Tabela 5 - Características Fonte de Alimentação

- Tipo ATX 12V v2.2	- Ventilação: 1 x 120 mm, preta
- Potencia: 500 W	- Dimensões: 140 x 150 x 86 mm
- Peso: 1.04 kg	

3.1.5 Módulo de pH

O módulo de pH, pH-4502C (Figura 12), é composto por um eletrodo de pH e um módulo eletrônico que faz a interface com o Arduino, podendo o eletrodo ficar submerso no recipiente com água, ficando somente com o cabo externo exposto. Para o eletrodo de pH existe uma interface BNC, e para a ligação ao microcontrolador existem 6 pinos, nomeadamente: 2 *Ground (GND)*, 1 *Vin (VCC)*, 1 *Digital pH Output (DOUT)*, 1 *Analog pH Output (p_AOUT)*, 1 *Analog Temperature Output (t_AOUT)* (Figura 13).

Por defeito, a saída de pH analógica está definida para 0 quando o pH lido no conector BNC é 7 portanto é feita a calibração através do potenciômetro R21, fazendo com que a saída em TCOM1 passe de 0V para 2,5V para que seja possível fazer a leitura de 0-14, que é a escala de pH.

Já a leitura da temperatura é feita de forma direta através do sensor de temperatura 18B20, com a saída a ser obtida de forma analógica em TCOM2.

Este módulo pode trabalhar sem recurso ao uso de microcontroladores e para isso usa-se o pino digital que é definido pela relação entre o valor obtido pelo eletrodo, com a regulação feita no potenciômetro R23.

Pelo menos, uma vez por mês, é importante fazer a calibração da ponta de prova e para isso é aconselhável usar soluções existentes com pH calibrado, habitualmente de 4, 6.86 e 9.18, a uma temperatura de 25°C juntando-a a 250 mililitros de água destilada.



Figura 12 - Módulo pH

As principais características do módulo de pH são apresentadas na Tabela 6.

Tabela 6 - Características Módulo pH

-Tensão 5 \pm 0.2V (AC/DC)	- Corrente de trabalho: 5-10mA
- Faixa de temperatura: 0-60°C	- Tempo de resposta: 5S
- Tempo de sedimentação: 60S;	- Componente Potência: 0,5 W
- Faixa de medição: 0,00 ~ 14,00 pH	- Erro alcalino: 0.2pH
- Resistência interna: <250MOhm	

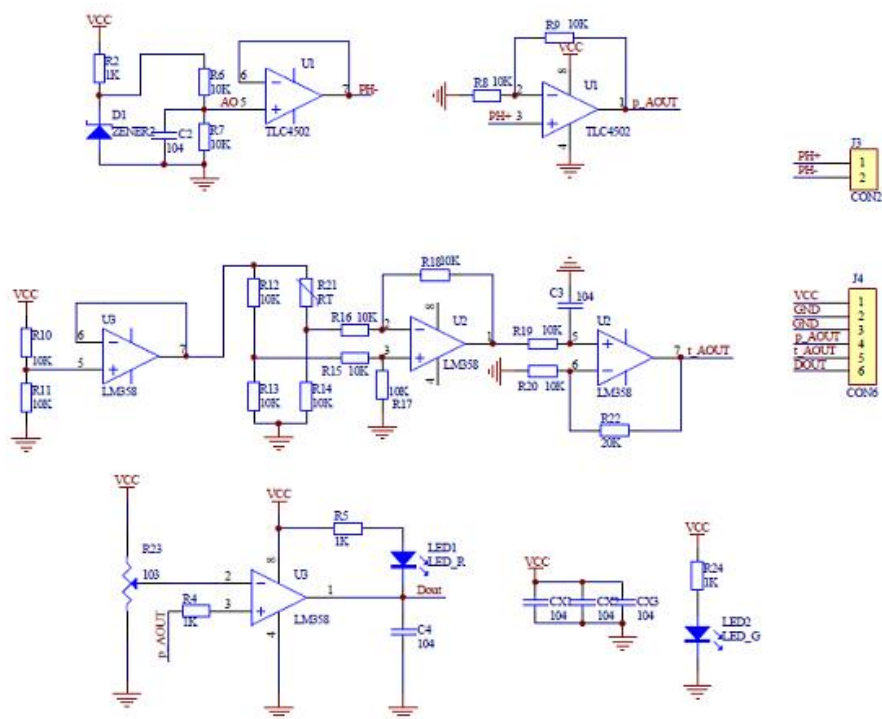


Figura 13 - Esquema Módulo pH

3.1.6 Medidor de EC - Versão 1

O medidor de EC foi estruturado na íntegra, segundo o projeto “Three Dollar EC”¹⁷ é composto por duas placas de metal, em INOX de 10mmx10mm, colocadas paralelamente a uma distância de 10mm e cada uma destas faces ligada a um condutor elétrico, devidamente isolado para não interferir com as leituras. Realiza a ligação a dois relés que nunca são ativos em simultâneo e a pinos analógicos que fazem a leitura de forma alternada. Isto é, um dos terminais do sensor é colocado a 5V colocando um pino como saída em estado alto, e o valor analógico é medido no outro pino que é colocado como entrada

¹⁷ <https://hackaday.io/project/7008-fly-wars-a-hackers-solution-to-world-hunger/log/24646-three-dollar-ec-ppm-meter-arduino>

Após testes com este sensor, verificou-se que, ao fim de algum tempo de utilização, o sensor ficava descalibrado, pelo que se procurou uma solução alternativa o *Gravity: Analog TDS Sensor*. O Medidor EC1 passou a ter a finalidade de detetor de nível do reservatório no sentido em que assim que este tiver solução entre os dois polos, apresenta uma condutividade muito superior do que quando está em ar. Depois de proceder ao um ensaio, foi possível perceber que a entrada digital ficava a 1 quando este estava com solução entre os dois polos e a 0 quando não tinha solução entre os mesmos.

3.1.7 Medidor de EC – Versão 2

O módulo *Gravity: Analog TDS Sensor/Meter 1.0 for Arduino* (Figura 14) é usado para fazer a leitura da quantidade de sais dissolvidos na água (TDS) Com base no esquema da Figura 15.

É um sensor de análise da água que consiste em mergulhar, igualmente, uma ponta de prova com dois condutores elétricos e aplicar uma tensão alternada obtendo a queda de tensão que existe entre esses dois pontos.

Esta tensão alternada é gerada pelo oscilador CD4060BM que faz chegar à ponta de prova uma onda quadrada que varia entre os +3V e os -3V. O sinal de saída da ponta de prova (pino 1 XH2.54-2P) é filtrado e ajustado para variar entre 0 e +2.3V. De notar que a tensão tem de ser alternada, caso contrário os sais iriam aglomerar-se no pino recetor de sinal e com isso, ao longo do tempo, as medições começariam a ser menos corretas.

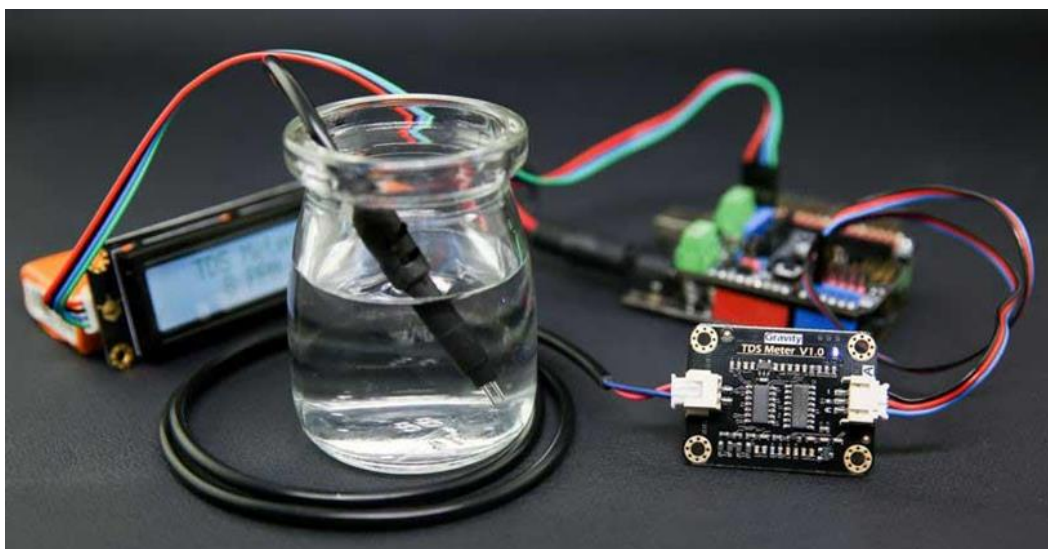


Figura 14 - Sensor de Condutividade

As principais características do módulo de CE são apresentadas na Tabela 7.

Tabela 7 - Características módulo EC

-Tensão Entrada: 3.3 ~ 5.5V	-Tensão Saída: 0 ~ 2.3V
-Corrente Trabalho: 3 ~ 6mA	-Alcance Medida TDS: 0 ~ 1000ppm
-Precisão medida TDS: $\pm 10\%$ F.S. 25 °C)	-Tamanho Módulo: 42 * 32mm
-Interface do Módulo: PH2.0-3P	-Interface: XH2.54-2P
-Número de agulhas: 2	-Comprimento Total: 83cm
-Interface Ligação: XH2.54-2P	-Outros: Ponta de prova a prova de água

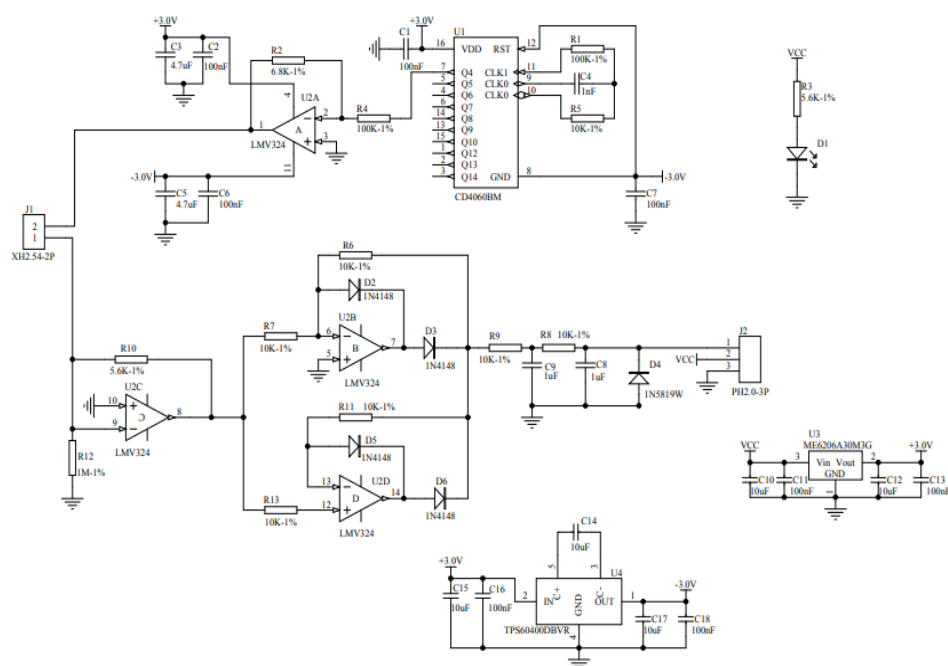


Figura 15 - Esquema do sensor de condutividade

3.1.8 Módulo LDR

Tal como o módulo de pH, o módulo de LDR (Figura 16) também pode trabalhar sem recurso a microcontroladores, tendo para isso 4 pinos de ligação, 2 para alimentação (5V e *Ground*), 1 para saída analógica e 1 para saída digital que vai atuar através da relação entre o sinal do LDR e o potenciômetro incorporado. Para isso é usado o comparador LM393 responsável pela relação de tensão entre o pino analógico com o valor selecionado no potenciômetro (Figura 17). Este módulo encontra-se ligado ao Arduino Micro e é responsável pela leitura da luz solar que incide sobre a estufa.

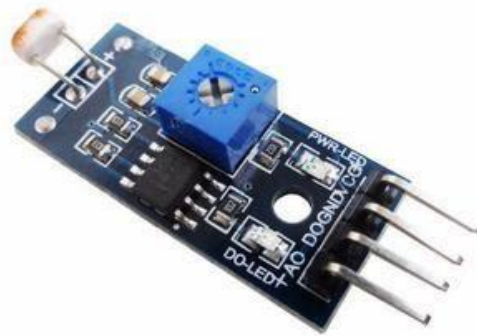


Figura 16 - Módulo LDR

As principais características do Módulo LDR são apresentadas na Tabela 8.

Tabela 8 - Características módulo LDR

- Diâmetro LDR: 5mm	- Potência máxima: 100mW
- Faixa Temperatura: -30°C a 70°C	- Espectro: 540nm
- Tensão de Operação: 3-5V	- Pico de resposta espectral (em 25°C): 540nm
- Saídas Digital e Analógica	- LED indicador para tensão
- Led indicador para saída digital	- Comparador LM393
- Dimensões: 30 x 13mm	

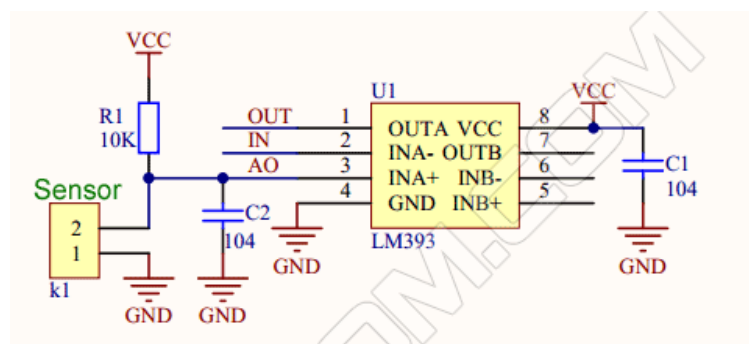


Figura 17 - Esquema do sensor LDR

3.1.9 Bomba Peristáltica

As bombas peristálticas (Figura 18) usam o princípio da alternância de compressões e relaxamento da mangueira para sugar o líquido. Esta mangueira atravessa uma roda com três saliências separadas, com distâncias iguais, criando assim vácuo, trabalhando contra a gravidade. Com tensão constante aplicada aos terminais do motor, pode-se fazer com que a dosagem de saída seja constante no tempo em que a bomba está acionada, e controlar, neste caso, a quantidade de nutrientes a serem adicionados ao depósito da estufa para ter a concentração da solução pretendida. Neste projeto foram usadas três bombas

iguais, duas para adicionarem os componentes de nutrientes A e B e a outra para elevar o pH com uma dosagem de 100 ml/min a uma tensão de 5V Tabela 9.



Figura 18 - Bomba Peristáltica

Tabela 9 - Características Bomba Peristáltica

– Tensão de Operação 4-6V	– Potência máxima: 1.8W
– Faixa de Temperatura: -5°C a 50°C	– Taxa de fluxo: 90-110 ml/min
– Velocidade de rotação: 0.1-60 rpm	– Tubo da bomba: diâmetro 2mm interior

3.1.10 Bomba Submersível

Como em diversos sistemas hidropônicos, aquapônicos¹⁸ e até aquários, é necessário um ponto de impulso para fazer movimentar a água, sendo que, neste caso, foi usado uma bomba submersa para desempenhar essa função (Tabela 10). Esta bomba é constituída por um motor que está isolado, mecanicamente, do líquido tendo apenas o veio que faz a ligação á bomba, onde se gera a força para fazer subir a solução de nutrientes, através da mangueira, até ao tubo de crescimento. Foi usada uma bomba de 230V com o fluxo máximo de 800L/hora podendo este ser ajustado através de um potenciômetro que, neste caso, não irá ser necessário.

Tabela 10 - Características Bomba Submersível

– Tensão de Operação 220-240V	– Frequência 50/60Hz
– Potência máxima: 13W	– Taxa de fluxo máxima: 800 L/hora
– Profundidade máxima: 1m	– Diâmetro Saída: 10mm interior

¹⁸<https://www.embrapa.br/busca-de-noticias/-/noticia/2767622/integrar-criacao-de-peixes-com-hortalicas-economiza-90-de-agua-e-elimina-quimicos>

3.1.11 Lâmpada Espetro Completo

As lâmpadas de espectro completo, emitem um espectro de luz próximo ao da luz natural, simulando a luz solar ao fornecer iluminação aos cultivos. As plantas florescem igualmente, tal como quando recebem a luz do sol. Tem o principal benefício de poder controlar a quantidade de horas da fotossíntese e respiração, ou apenas a respiração nas plantações. De um modo geral, as plantações precisam entre 12 a 18 horas de exposição a iluminação para terem um crescimento contínuo, sem prejudicar a saúde da planta.

Consoante o propósito, existem lâmpadas para os vários tipos de estágio da planta, entre eles o crescimento (tipo MH¹⁹), floração (tipo HPS²⁰) e o crescimento + floração (tipo SHPS²¹). Estas últimas são as que mais se adequam ao projeto em questão pois é uma utilização de baixa intensidade de produtividade.

Entre os vários tipos de lâmpada, variando entre potências e tipo de espectro, foi usada para este caso, em específico, o modelo de 18W (Tabela 11) com as cores vermelha e azul (comprimentos de onda relevantes situam-se entre os 400nm e os 700nm), uma vez que o plano a iluminar é de baixa área. Como a potência de uso é baixa, é aconselhável ficar a uma distância mínima de 20cm e a temperatura não deve exceder os 28°C. A cultura deve receber em média 1,4MJ/m² (118.6Mlx/s) segundo a informação obtida num estudo realizado por Márkilla Zunete Beckmann [10], pois valores inferiores estagnam o desenvolvimento das plantas. Neste caso irá ter-se sempre em consideração 10000lux.

Tabela 11 - Características Lâmpada Espetro Completo

– Material: Alumínio	– Tensão de operação: 85-265V
– Frequência: 50/60Hz	– Potencia: 20W
– Led chip: SMD5730	– Tipo de base: e27
– Quantidade de led: 18 leds	– Eficiência do refletor: 95%
– Vida da fonte:> 50,000 horas	– Comprimento de onda: vermelho 660nm; azul 554nm

¹⁹ Metal halide (MH) lamps – Lampada de iodeto metálico

²⁰ High Pressure Sodium (HPS) lamps – Lampada de Sódio de alta pressão

²¹ Super High Pressure Sodium (HPS) lamps – Lampada de Sódio de super alta pressão

3.1.12 Electroválvula Solenoide

A válvula solenoide nada mais é do que uma válvula eletromecânica controlada. O seu nome deve-se ao facto de o componente principal ser uma bobina elétrica com um núcleo ferromagnético móvel no centro, sendo este núcleo chamado de êmbolo. Numa posição de repouso, o êmbolo fecha um pequeno orifício por onde é capaz de circular um fluido.

Quando uma corrente elétrica circula através da bobina, esta corrente cria um campo magnético que, por sua vez, exerce uma força no êmbolo. Como resultado, o êmbolo é puxado em direção ao centro da bobina de modo que o orifício se possa abrir, sendo este o princípio básico usado para abrir e fechar uma válvula solenoide.

Quanto ao tipo de válvula, pode-se caracterizar pelo número de posições e vias que para este caso é usada uma do tipo 2/2 normalmente fechada (Tabela 12). Isto é, tem 2 ligações (saída e entrada de água) e 2 posições (aberta e fechada). Já a operação pode ser direta, indireta ou semidirecta. Para este caso, irá ser o de ação direta, fazendo o líquido passar diretamente pelo orifício criado pelo êmbolo.

Tabela 12 - Características Válvula Solenoide

– Corpo da válvula: Plástico	– Tamanho da entrada: 1/2 polegada
– Temperatura de operação: 0-40°C	– Tipo: válvula solenoide direta 2/2
– Tensão nominal: 220 V	– Grau de isolamento: E
– Fluido: Água	– Fluido: Temperatura: 0°C-55°C
– Pressão: 0.02-0.8MPa	– Tempo de vida:> 100,000 Vezes

3.2 Software utilizado nos módulos do sistema hidropónico

O desenvolvimento do software tem como principal objetivo fazer com que módulos genéricos, incluídos no sistema, executem as tarefas necessárias e disponibilizando ao utilizador a supervisão e controlo das variáveis de estado do sistema em formato página web.

Com o sistema está ligado a um router Wi-Fi, é possível ao utilizador verificar o que se está a passar dentro da estufa, bem como aceder às ações possíveis de realizar através de um qualquer dispositivo, como por exemplo um PC ou smartphone.

Para isso é montado um esquema de dois sentidos: o primeiro passa por uma interface com o utilizador (através da página web), com recolha de informação (ações de controlo) que é armazenada na base de dados. Esta informação é analisada por um *script* de *python*²², presente na RPi, que depois de ser processada envia para o Arduino através de uma ligação por porta serie os comandos a serem realizados pelos atuadores de acordo com os dados lidos.

No sentido inverso, o Arduino recolhe a informação dos sensores que cria uma “frase” com os dados recolhidos de forma ordenada e envia-a para a RPi que faz a leitura da frase através do mesmo *script* de *python* acima mencionado. Este faz a gestão dos dados e regista-os de forma ordenada na base de dados que ficam disponíveis na página web.

No desenvolvimento deste processo foi usado linguagem C para a programação do Arduino, Python para o script no RPi, MyPhpAdmin para a administração e gestão de base de dados e ainda HTML, JavaScript, CSS e PHP no desenvolvimento WEB.

²² <https://www.python.org/>

3.2.1 Supervisão e Controlo de sensores e atuadores em linguagem C/C++

Foi utilizado o Arduino IDE²³ (ferramenta de desenvolvimento integrado) para o desenvolvimento do programa em C/C++ que é executado no Arduino Micro.

Através desta linguagem realiza-se um ciclo contínuo no Arduino Micro para serem realizadas as tarefas necessárias, nomeadamente, fazer a leitura dos sensores (EC, pH, Luminosidade), acrescentar nutrientes à solução existente no tanque, corrigir o pH adicionando ácido ao recipiente, ligar a lâmpada caso a luminosidade exterior não seja suficiente, renovar a solução da calha, e comunicar com a RPi.

Como será descrito mais a frente, este processo espera até receber a comunicação que o utilizador adicionou uma nova planta na estufa, através da leitura da informação disponibilizada pela RPi na porta série (UART²⁴), para isso foi usada o standard JSON²⁵. O JSON permite ao programador enviar e receber dados entre dois dispositivos permitindo definir quais as variáveis e respetivos valores serão transmitidos/trocados pelos dois dispositivos.

Foi feita uma rotina para um crescimento sustentado, que começa por acionar a válvula solenoide que faz entrar a água no recipiente, controlando a luminosidade de acordo com leitura do sensor LDR e gerindo o tempo que a lâmpada está ligada. Esta rotina é usada também para ajustar os valores de EC e pH

3.2.2 Interface entre Base de dados e sistema Arduino em Python

O *Python*²⁶ é, assim como C/C++, uma linguagem de alto nível orientada a objetos, criada na década de 80 por Guido Van Rossum, e é atualmente uma das linguagens mais usadas. Tem dois tipos fundamentais de execução, o modo interativo, em que o utilizador pode executar pequenas funções diretamente na *Python Shell*²⁷, ou o modo *script* em que o programador usa um arquivo em ficheiro de texto que será executado pelo compilador.

²³ <https://www.arduino.cc/en/software>

²⁴ UART (Universal Asynchronous Receiver / Transmitter) é uma forma de comunicação entre dois dispositivos. Este usa bytes de dados e transmite os bits individuais de forma sequencial. No destino, um segundo UART reúne os bits em bytes completos.

²⁵ <https://www.json.org/json-en.html>

²⁶ <https://www.python.org/>

²⁷ Janela de execução de Python

Neste caso, foi desenvolvido um *script* para criar uma ligação entre o RPi e Arduino e ainda gerir a informação disponível para o utilizador.

A ligação entre o RPi e Arduino é também realizada através da biblioteca JSON, em que é enviado para o microcontrolador os parâmetros e valores definidos pelo utilizador para EC, pH e controlo da luminosidade.

Toda a vez que existe uma transmissão de dados, estes são analisados e incorporados numa base de dados, para isso foi também criada, através deste script, o acesso á mesma. Para isto, foi usada a biblioteca *pymysql*²⁸, esta é uma das bibliotecas que permite ao utilizador fazer a gestão da base de dados através de *Python*. Esta é fundamentalmente usada para ler os valores de referência de EC, pH definidos pelo utilizador, registar em tempo real os dados atuais da estufa e reiniciar as tabelas da base de dados caso exista uma nova plantação.

Para a comunicação com o programador e/ou administrador do sistema é possível usar a *SHELL* (ver Figura 44 em apêndices). Esta *interface* do *script* permite ao utilizador verificar o ponto de situação sobre o decorrer dos processos.

3.2.3 Base de Dados do sistema existente na Raspberry Pi

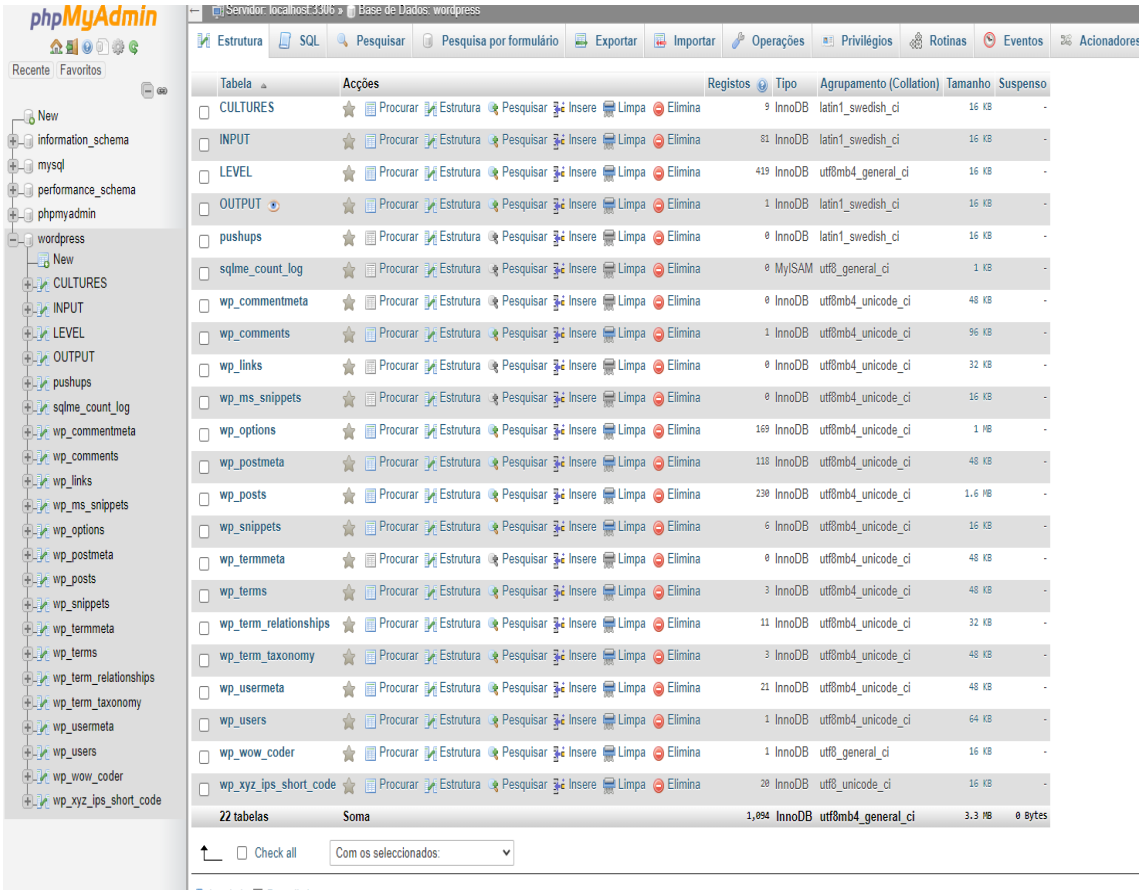
A BD do sistema foi desenvolvida para armazenar todo o histórico, dados atuais e as características base das plantações padrão ou criadas pelo utilizador.

A BD inclui 3 tabelas de maior importância: a tabela CULTURES regista os dados originais para um tipo de plantação (Tipo de plantação, EC recomendado, pH, tempo de crescimento e ainda a possibilidade de criar uma descrição relativa a mesma); a tabela INPUT - permite adicionar a data e hora exata a que foram recolhidos todos os valores obtidos pelos sensores de EC, pH, luminosidade e temperatura; por último a tabela OUTPUT serve como referência para a cultura em crescimento dentro da estufa, isto é, o valor de EC e pH o tempo de crescimento, o estado da lâmpada e ainda a opção entre modo automático e manual.

²⁸ <https://pypi.org/project/PyMySQL/>

Uma vez que esta base de dados reside na *RPi*, optou-se pelo sistema de gestão *MariaDB*²⁹ por ser gratuito e por se adequar ao sistema *Linux*.

Todo este ponto pode ser visualizado e administrado de uma forma mais *User Friendly* (amiga para o utilizador), através do aplicativo web livre *phpMyAdmin*³⁰ que como o próprio nome indica, tem como base a linguagem PHP³¹ para poder remover, criar, modificar tabelas, campos, linhas, colunas e até executar códigos SQL³². Na Figura 19 é apresentada a estrutura principal da base de dados que é mostrada ao programador onde além das tabelas acima referidas, estão também as tabelas necessárias para o bom funcionamento do sistema de gestão de conteúdo da página web. Para este efeito foi usado o *Wordpress*.



The screenshot shows the phpMyAdmin interface for a database named 'wordpress'. The left sidebar lists the database structure, including 'information_schema', 'mysql', 'performance_schema', 'phpmyadmin', and 'wordpress'. The main panel displays a table of database tables with their respective actions, record counts, types, collations, and sizes.

Tabela	Ações	Registos	Tipo	Agrupamento (Collation)	Tamanho	Suspensão
CULTURES	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	9	InnoDB	latin1_swedish_ci	16 KB	-
INPUT	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	81	InnoDB	latin1_swedish_ci	16 KB	-
LEVEL	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	419	InnoDB	utf8mb4_general_ci	16 KB	-
OUTPUT	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	1	InnoDB	latin1_swedish_ci	16 KB	-
pushups	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	0	InnoDB	latin1_swedish_ci	16 KB	-
sqlme_count_log	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	0	MyISAM	utf8_general_ci	1 KB	-
wp_commentmeta	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	0	InnoDB	utf8mb4_unicode_ci	48 KB	-
wp_comments	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	1	InnoDB	utf8mb4_unicode_ci	96 KB	-
wp_links	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	0	InnoDB	utf8mb4_unicode_ci	32 KB	-
wp_ms_snippets	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	0	InnoDB	utf8mb4_unicode_ci	16 KB	-
wp_options	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	169	InnoDB	utf8mb4_unicode_ci	1 MB	-
wp_postmeta	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	118	InnoDB	utf8mb4_unicode_ci	48 KB	-
wp_posts	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	230	InnoDB	utf8mb4_unicode_ci	1.6 MB	-
wp_snippets	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	6	InnoDB	utf8mb4_unicode_ci	16 KB	-
wp_termmeta	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	0	InnoDB	utf8mb4_unicode_ci	48 KB	-
wp_terms	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	3	InnoDB	utf8mb4_unicode_ci	48 KB	-
wp_term_relationships	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	11	InnoDB	utf8mb4_unicode_ci	32 KB	-
wp_term_taxonomy	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	3	InnoDB	utf8mb4_unicode_ci	48 KB	-
wp_usermeta	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	21	InnoDB	utf8mb4_unicode_ci	48 KB	-
wp_users	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	1	InnoDB	utf8mb4_unicode_ci	64 KB	-
wp_wow_coder	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	1	InnoDB	utf8_general_ci	16 KB	-
wp_xyz_ips_short_code	Procurar Estrutura Pesquisar Inserir Limpar Eliminar	20	InnoDB	utf8_unicode_ci	16 KB	-
22 tabelas Soma		1,094	InnoDB	utf8mb4_general_ci	3.3 MB	0 Bytes

Figura 19 - Exemplo da BD do sistema

²⁹ MariaDB – Sistema de gestão de base de dados - <https://mariadb.org/>

³⁰ <https://www.phpmyadmin.net/>

³¹ <https://www.php.net/>

³² SQL – Linguagem de consulta estruturada - <https://www.w3schools.com/sql/>

3.2.4 WordPress - Interface gráfica WEB com o utilizador

O *WordPress*³³ é um sistema de gestão de conteúdo (SGC) em código aberto que facilita a criação e a administração de um site. Um site do *WordPress* é composto por diversos tipos diferentes de dados, sendo que todas essas informações são armazenadas num banco de dados *WordPress* central. Esta base de dados torna possível que o site seja executado e que as alterações que o proprietário ou os seus visitantes façam sejam salvas.

Quando um utilizador visita um site, o seu navegador envia um pedido ao servidor do site. Consequentemente, o servidor envia os dados que são necessários para exibir o site e fazê-lo funcionar corretamente. Não obstante, uma BD requer o seu próprio software para funcionar.

O *WordPress* utiliza um sistema de gestão de BD chamado *MySQL*, que é um software de código aberto. *MySQL* é o que permite à BD armazenar informações e fornecer acesso a elas. Quando os dados precisam ser armazenados, alterados ou excluídos, o *WordPress* envia instruções sobre quais os dados que devem ser afetados e o que deve ser feito com eles.

No entanto, o proprietário do site também pode aceder o seu banco de dados diretamente e executar esses tipos de comandos manualmente, ou usar um gestor de BD para simplificar o processo.

Para que a enorme quantidade de informação seja rápida e facilmente acedida, ela é organizada numa série de “tabelas de BD”. O *WordPress* usa tabelas para comentários, *posts*, links, etc. Além disso, cada tabela contém várias colunas e campos que contêm tipos de informação ainda mais específicos. Por exemplo, a tabela *wp_comments* contém dados relacionados aos comentários deixados em páginas e *posts*.

Isso significa essencialmente que a BD armazena muitas informações sobre cada comentário específico. Isso inclui a ID exclusiva do comentário, a mensagem onde ele está localizado, detalhes sobre seu autor e muito mais. Se se quiser excluir comentários de spam, esta tabela de BD é o que se precisaria aceder usando uma consulta *MySQL*.

³³<https://kinsta.com/pt/base-de-conhecimento/base-de-dados-wordpress/>

A escolha recaiu sobre esta aplicação sobretudo para gerar um *website* que tenha uma vista uniforme em todos os tipos de navegadores, resolução e dispositivos. Também onde existe uma classe específica que facilita a comunicação com a BD intitulada de *\$wpdb*.

Esta aplicação fica alocada no servidor (Raspberry Pi) e tem como ponto de comunicação entre a interface com o utilizador e o sistema de controlo (Arduino Micro) um *script Python* que busca informação na base de dados gerada pelo *Wordpress*.

Todo o *website* tem um tema base, que é ajustado às necessidades das páginas, ficando todas com uma estrutura muito semelhante. Neste caso, usou-se o tema criado pela equipa do *WordPress* designado TWENTY SEVENTEEN³⁴. O corpo central, detém um código em HTML desenvolvido, na íntegra, de raiz onde aparecem os campos, botões e indicadores, que fazem a diferença entre cada uma das páginas. Com esta dinâmica de programação, é possível usar outras linguagens que, neste caso, foram o PHP para comunicação com a BD e o *JavaScript* para atualizar em tempo real alguns indicadores e ainda personalizar tudo isto através de CSS³⁵.

Para o desenvolvimento deste ponto, foi usado como suporte um projeto denominado de *Build a LAMP Web Server with WordPress*³⁶. Este é um projeto que dá informação de como instalar e aceder ao *Wordpress* através da Raspberry Pi. Após este processo dá-se início ao desenvolvimento do conteúdo da página web (interface com o utilizador) com base em HTML, *JavaScript* para o que é exibido no site e PHP para comunicar com a base de dados.

O HTML³⁷, (abreviação para a expressão inglesa *HyperText Markup Language*, que significa *Linguagem de Marcação de Hipertexto*) é uma linguagem de marcação usada na construção de páginas na Web. Esta tecnologia é fruto da junção entre os standards *HyTime* e SGML. Por sua vez, *HyTime* é um padrão para a representação estruturada de *hipermídia* e conteúdo baseado em tempo. Um documento é visto como um conjunto de eventos concorrentes dependentes de tempo (como áudio, vídeo, etc.), conectados por hiperligações. O padrão é independente de outros padrões de processamento de texto em

³⁴ <https://wordpress.org/themes/twentyseventeen/>

³⁵ CSS – (Cascading Style Sheets) – Linguagem para adicionar estilo a um documento web

³⁶ <https://projects.raspberrypi.org/en/projects/lamp-web-server-with-wordpress/>

³⁷ <https://developer.mozilla.org/en-US/docs/Web/HTML>

geral. Já o SGML, é um padrão de formatação de textos. Não foi desenvolvido para hipertexto, mas tornou-se conveniente para transformar documentos em hiper-objetos e para descrever as ligações. Portanto o código HTML é, basicamente, um conjunto de *tags* em que cada um deles representa elementos das páginas, entre eles, links, imagens, textos, tabelas, entre outros. No sentido de embelezar o código HTML, foi incorporado o CSS (*Cascading Style Sheet*) para melhorar a estética da página e com *tags* de `` onde se pode alterar cores, tamanhos e tipos de letra.

O *Javascript*³⁸ é uma linguagem aplicada ao lado do cliente, isto é, todo o código é processado pelo navegador em vez de ser pelo próprio servidor, como é o caso do PHP. É uma linguagem que tem a grande função de interatividade com o utilizador e alteração em tempo real de indicadores, dando maior dinamismo à página. Neste caso a maior implementação está representada pela contagem decrescente continuamente apresentada na página de Dados

O PHP³⁹ (um acrónimo recursivo para *PHP: Hypertext Preprocessor*), é usado sobretudo para obter ou enviar valores de/ou para a BD, onde o código fonte não fica disponível para o utilizador comum, ou seja, apenas quem tem acesso de administrador terá acesso a gestão de todos os serviços. É uma linguagem dedicada ao lado do servidor e, por isso, muito usada para o desenvolvimento de servidores web e páginas web podendo ser embutido em todo o tipo de documentos HTML.

.

³⁸ <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

³⁹ <https://www.php.net/>

4 Ensaios do Sistema

Para a realização deste projeto realizaram-se 2 ensaios experimentais. No primeiro ensaio, foi usado o Arduino Micro, como principal gestor do sistema. Era este que realizava todo o controlo da estufa, desde enviar dados para o utilizador, ler sensores, pedir referências a BD, enquanto a RPi funciona como *Slave* e apenas seria colocada em ação sempre que fosse solicitado pelo *Master* (Arduino Micro). Num segundo ensaio, realizou-se essa mesma gestão a ser definida pelo servidor, neste caso a *Raspberry Pi* com o auxílio do *Python*. Em ambos os casos foi implementada uma base de dados que é descrita em seguida. Na Figura 20 é possível visualizar a estrutura da estufa desenvolvida para os ensaios. É uma estrutura de pequenas dimensões (para apenas quatro plantas em simultâneo), com dois andares, onde se encontram, no andar de superior, as plantas (Figura 36 em Apêndice) e, no andar inferior, os reservatórios de solução nutritiva (Figura 37 em Apêndice), e solução ácida acompanhados pelo sistema de controlo (Figura 39 em Apêndice).

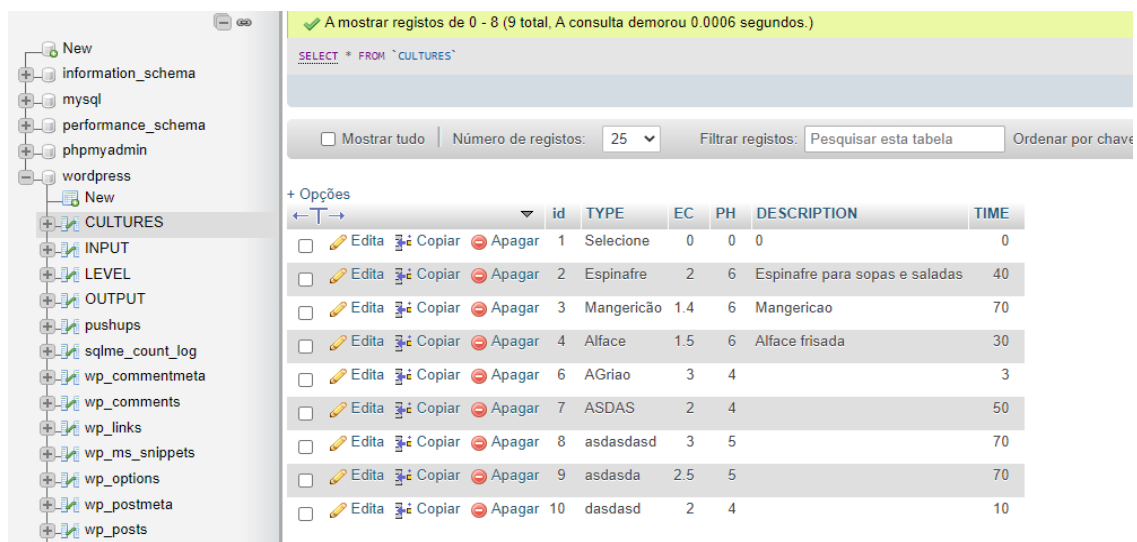


Figura 20 - Estrutura da Estufa

4.1 Tabelas da Base da Dados de Suporte ao Sistema

Como já referido, na Figura 19, são apresentadas tabelas usadas para a ligação entre o utilizador e a máquina. Além das que já se encontram pré-configuradas para a comunicação com a plataforma *Wordpress* foram ainda adicionadas 4 tabelas: CULTURES, INPUT, LEVEL e OUTPUT.

Na tabela CULTURES, apresentada na Figura 21, fica guardada a informação sobre as já existentes e novas sementes que sejam adicionadas pelo utilizador sendo que terão de conter obrigatoriamente o Tipo de sementes (TYPE), o EC inicial (EC), pH inicial (pH), e o tempo de crescimento (TIME).



A screenshot of a database management interface. On the left is a sidebar with a tree view showing the database structure: 'New' (expanded), 'information_schema', 'mysql', 'performance_schema', 'phpmyadmin', 'wordpress', and 'New' (expanded). Under the second 'New' folder, the tables 'CULTURES', 'INPUT', 'LEVEL', and 'OUTPUT' are listed. The main area displays the 'CULTURES' table. At the top, a green status bar says 'A mostrar registos de 0 - 8 (9 total, A consulta demorou 0.0006 segundos.)'. Below it is a query editor with the text 'SELECT * FROM `CULTURES`'. Further down are controls for 'Mostrar tudo', 'Número de registos: 25', 'Filtrar registos: Pesquisar esta tabela', and 'Ordenar por chave'. The table itself has columns: 'id', 'TYPE', 'EC', 'PH', 'DESCRIPTION', and 'TIME'. It contains 9 rows of data, each with a checkbox, 'Edita', 'Copiar', and 'Apagar' icons. The data rows are as follows:

	id	TYPE	EC	PH	DESCRIPTION	TIME
<input type="checkbox"/>	1	Selecione	0	0	0	0
<input type="checkbox"/>	2	Espinafre	2	6	Espinafre para sopas e saladas	40
<input type="checkbox"/>	3	Mangericão	1.4	6	Mangericao	70
<input type="checkbox"/>	4	Alface	1.5	6	Alface frisada	30
<input type="checkbox"/>	6	AGriao	3	4		3
<input type="checkbox"/>	7	ASDAS	2	4		50
<input type="checkbox"/>	8	asdasdasd	3	5		70
<input type="checkbox"/>	9	asdasda	2.5	5		70
<input type="checkbox"/>	10	dasdasd	2	4		10

Figura 21 - Tabela Demonstrativa CULTURES

Tabela INPUT, nesta tabela representada na Figura 22, adicionam-se todos os valores obtidos pelos sensores de EC, pH, LDR e ainda, automaticamente, é adicionada a hora exata em que os valores são inseridos.

Server: localhost:3306 » Base de Dados: wordpress » Tabela: INPUT

✓ A mostrar registos de 50 - 74 (81 total, A consulta demorou 0.0002 segundos.)

SELECT * FROM `INPUT`

<< < 3 > >> | ☐ Mostrar tudo | Número de registos: 25 | Filtrar registos: Pesquisar esta tabela | Ordenar por ch

+ Opções

		id	EC	PH	LUMINOSIDADE	TEMPERATURA	TIME
<input type="checkbox"/>	Edita	51	2.01	5.39	203.47	0.55	2020-06-30 06:11:23
<input type="checkbox"/>	Edita	52	2.01	5.4	303.1	0.55	2020-06-30 06:19:18
<input type="checkbox"/>	Edita	53	2.01	5.39	661.11	0.55	2020-06-30 06:49:18
<input type="checkbox"/>	Edita	54	2.04	5.39	864.29	0.55	2020-06-30 07:19:18
<input type="checkbox"/>	Edita	55	2.06	5.38	994.9	0.55	2020-06-30 07:49:19
<input type="checkbox"/>	Edita	56	2.05	5.39	1063.04	0.55	2020-06-30 08:19:19
<input type="checkbox"/>	Edita	57	2.05	5.41	1113.64	0.55	2020-06-30 08:49:19
<input type="checkbox"/>	Edita	58	2.03	5.4	1154.71	0.55	2020-06-30 09:19:19
<input type="checkbox"/>	Edita	59	2.05	5.37	1230	0.55	2020-06-30 09:49:19
<input type="checkbox"/>	Edita	60	2.03	5.4	1333.78	0.55	2020-06-30 10:19:19
<input type="checkbox"/>	Edita	61	2.04	5.4	1455.88	0.55	2020-06-30 10:49:19
<input type="checkbox"/>	Edita	62	2.01	5.42	1715.52	0.55	2020-06-30 11:19:19
<input type="checkbox"/>	Edita	63	2.03	5.39	1919.23	0.55	2020-06-30 11:49:18
<input type="checkbox"/>	Edita	64	2.02	5.38	2176.09	0.55	2020-06-30 12:19:19
<input type="checkbox"/>	Edita	65	2.01	5.38	2794.44	0.55	2020-06-30 12:49:19
<input type="checkbox"/>	Edita	66					2020-06-30 16:04:29
<input type="checkbox"/>	Edita	67	1.52	5.44	17016.67	0.57	2020-06-30 19:16:22

Figura 22 - Tabela Demonstrativa INPUT

Na tabela LEVEL (Figura 23) é possível acompanhar do nível de líquidos existentes nos depósitos de EC e pH, para que seja possível ao utilizador seguir o nível através da página web “Níveis de Líquidos”.

Server: localhost:3306 » Base de Dados: wordpress » Tabela: LEVEL

✓ A mostrar registos de 0 - 24 (419 total, A consulta demorou 0.0006 segundos.)

SELECT * FROM `LEVEL`

1 > >> | ☐ Mostrar tudo | Número de registos: 25

+ Opções

		id	EC1	EC2	pH
<input type="checkbox"/>	Edita	1	100	100	100
<input type="checkbox"/>	Edita	2	100	100	99.8
<input type="checkbox"/>	Edita	3	100	100	99.6
<input type="checkbox"/>	Edita	4	100	100	99.4
<input type="checkbox"/>	Edita	5	100	100	99.2

Figura 23 - Tabela Demonstrativa LEVEL

A tabela OUTPUT (Figura 24) é o principal elo de ligação entre o que utilizador e a estufa, pois, a partir desta é definido o tipo de planta, o EC e pH que irá estar presente na solução que as plantas irão receber. Para isto é apresentado ao utilizador, na página web

desenvolvida, um separador onde o utilizador insere os valores pretendidos e que, por sua vez, irão ser atualizados na tabela OUTPUT através da comunicação feita por PHP. Sempre que existir alguma alteração, esta é detetada pelo *script* de *python* que envia a informação ao Arduino para que sejam ajustados os novos valores desejados pelo utilizador.

The screenshot shows the phpMyAdmin interface. On the left is the database structure tree. The main area displays the 'OUTPUT' table from the 'wordpress' database. A message at the top indicates that 0 records are shown out of 1 total. Below this, a table with 8 columns is shown: id, PLANTA, LAMPADA, HORA, EC, PH, DIAS_CRESCIMENTO, and MODO_AUTOMATICO. One record is listed with id 1, planta 'alface', LAMPADA 0, HORA '2020-06-28 19:31:26', EC 1.5, PH 5.9, DIAS_CRESCIMENTO 30, and MODO_AUTOMATICO 1. Below the table are options to check all, edit, delete, or export the selected record.

Figura 24- Tabela Demonstrativa OUTPUT

4.2 Página WEB, Hidroponia de Interior

Foi desenvolvida uma página *web* onde o utilizador pode escolher o que plantar, visualizar os últimos valores obtidos pelos dos sensores e ainda ajustar os parâmetros padrão. Para isso foram criadas 4 páginas:

- **Plantar** (plantação de nova semente na estufa) – nesta página web apresentada na figura 18 existem três opções padrão na base de dados (Alface, Manjerição e Espinafre) bastando clicar nas respetivas imagens, para começar uma nova cultura. Também existem quatro campos, para se poder inserir uma planta diferente, onde se deve colocar o nome da nova semente, o EC e pH recomendado e o tempo de crescimento médio até atingir o ponto de colheita. Caso se tente criar nova cultura com alguma em crescimento, irá aparecer uma mensagem, descrevendo que a estufa se encontra em utilização.

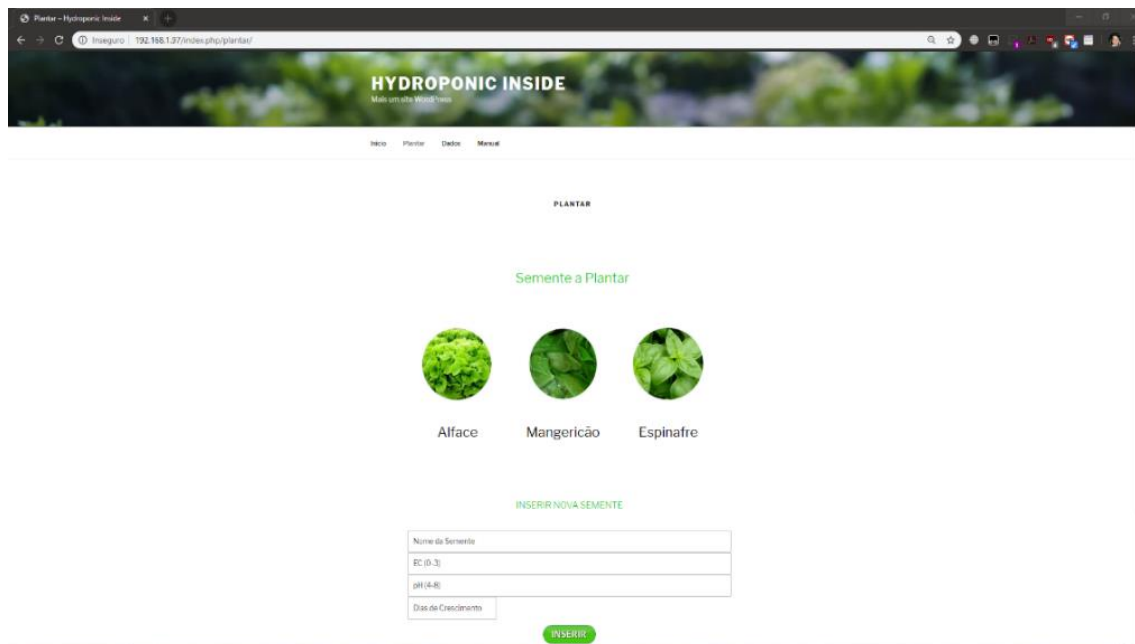


Figura 25 - Página WEB Plantar

- **Dados** – Nesta página encontram-se os dados atuais da estufa, a temperatura, o tempo restante para a colheita, a luminosidade, o EC e pH atualizados em tempo real. Também existe uma ligação para descarregar todo o histórico registado na base de dados e ainda o botão COLHER, que deverá ser usado quando se efetua a colheita das plantas, para reiniciar o sistema e assim poder receber uma nova plantação.

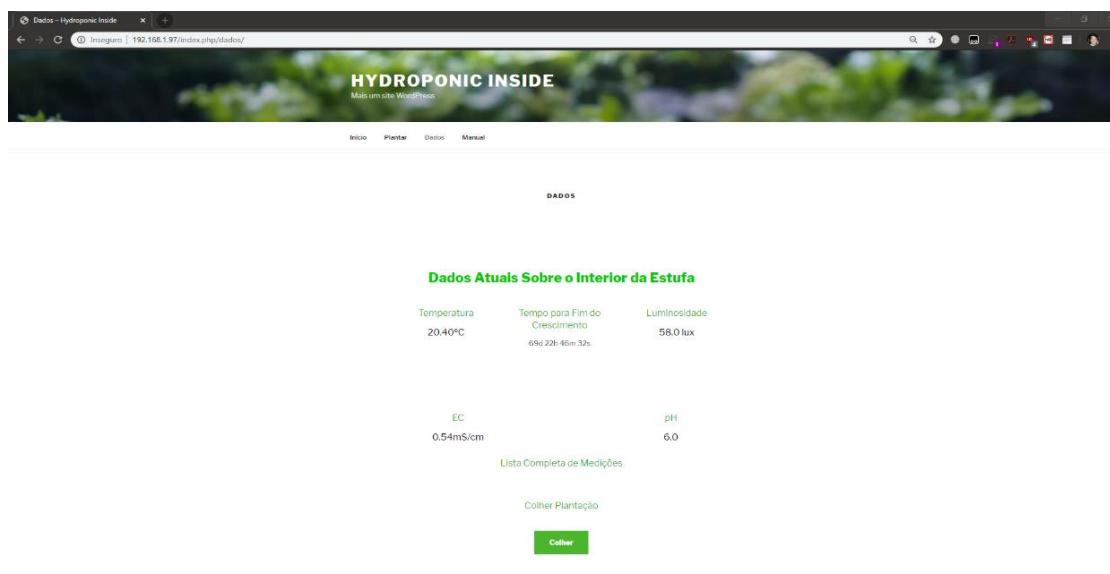


Figura 26- Página WEB Dados

- **Manual** – Local onde se pode alterar os valores iniciais de EC e pH, neste caso, só se poderá aumentar a partir dos dados iniciais, caso contrário ocorre uma mensagem de erro e os dados permanecem inalterados. Há ainda a possibilidade de desligar/ligar a lâmpada de espectro completo e a alteração entre modo Manual e Automático.

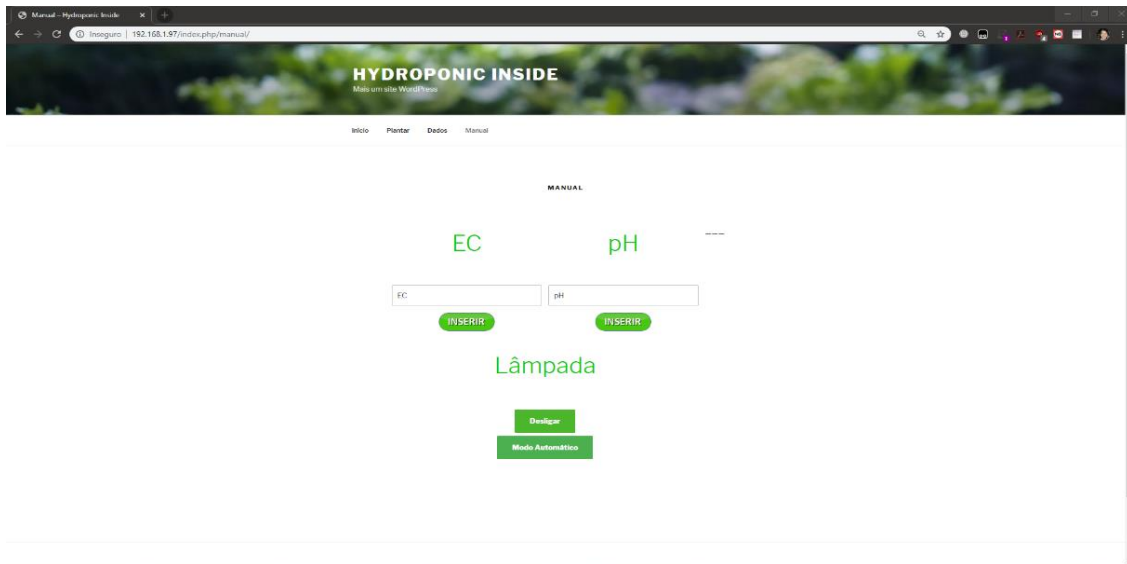


Figura 27 - Página WEB Manual

- **Nível de Líquidos** – Secção que serve para acompanhar, em tempo real, a percentagem de líquido ainda existente nos recipientes de EC e pH. Esta percentagem é obtida exclusivamente através de fórmulas matemáticas uma vez que cada rotina em que adiciona nutrientes ou acido á solução 0.5ml. Também é indispensável ao utilizador certificar-se que a cada nova planta, os 3 depósitos se encontram cheios.

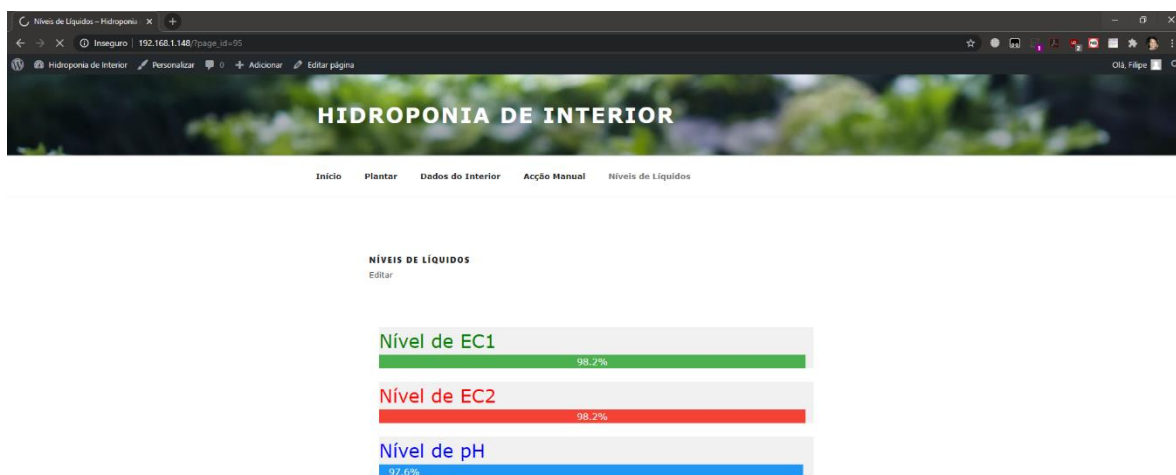


Figura 28 - Nível de Líquidos

4.3 Ensaio de funcionamento com gestão no Arduino – Ensaio 1

O princípio básico de funcionamento da estufa é exposto na Figura 29.

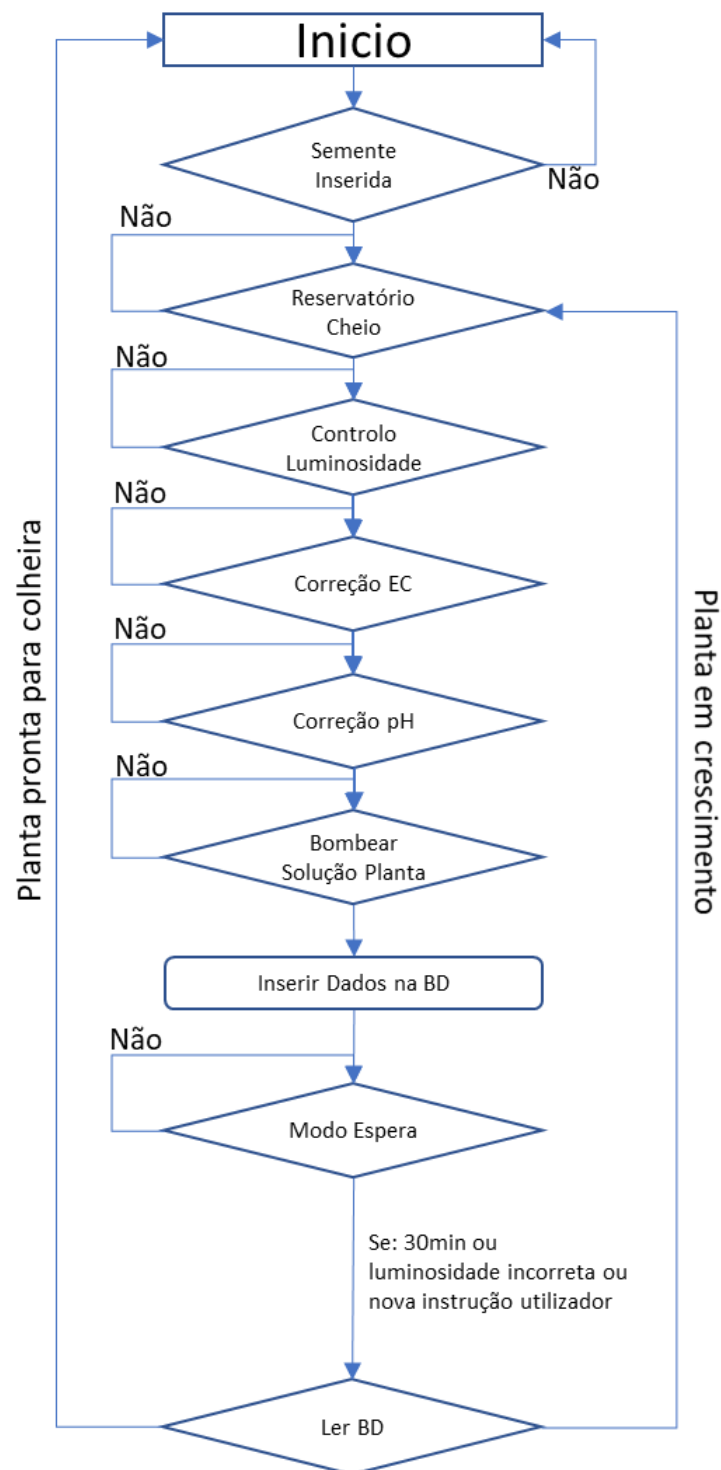


Figura 29 - Fluxograma representativo da rotina do Ensaio 1

Como se pode verificar, esta gestão está assente sobre uma rotina de funcionamento, em que cada bloco é explicado de seguida.

4.3.1 Semente Inserida

A inserção da semente é feita em dois passos fundamentais:

1. Colocar as sementes envolvidas, em algodão ou lã de rocha em cada copo do tubo.
2. Caso seja escolhida uma das plantas já existentes na base de dados, basta ir à página PLANTAR alocada no servidor e clicar na imagem referente à semente em questão (alface, manjerição e espinafre) e são automaticamente definidos os padrões base para o nível de nutrientes e de pH para um crescimento saudável da planta.

Para cada uma destas três sementes pré-definidas, existe código HTML e PHP como o representado na Figura 25. Código este que serve, para colocar na base de dados, mais concretamente na tabela OUTPUT, um *array* com os dados necessários.

Para inserir manualmente uma nova planta ter-se-á de conhecer os padrões base associados, e registá-los na página PLANTAR, nos campos destinados para esse fim (Figura 30).

INSERIR NOVA SEMENTE

Nome da Semente	
EC (0-3)	
pH (4-8)	
Dias de Crescimento	

INSERIR

Figura 30 - Página Web - Secção Plantar (Nova Semente)

Para não exceder os parâmetros nominais máximos e mínimos do sistema, foi colocado ao utilizador um bloqueio de EC e pH, que ao serem excedidos, impede a inserção de dados na tabela. Assim que os dados estão dentro dos parâmetros, inicia uma nova plantação e são colocados na tabela OUTPUT os valores inseridos (Apêndice 3.1).

Após se concretizar esse processo, surge o script em *Python* que irá, numa primeira fase, enviar, do RPI, pela porta série, para o Arduino Micro a data e hora a que foi inserida a semente, para serem criados ciclos com tempos reais. E, numa segunda fase, remeter a

linha inserida na BD pelo *array* (mensagem enviada pela porta série) para o Arduino (Apêndice 3.2).

Por fim, o Arduino, espera que esses dados sejam colocados na porta série sequenciados em quatro parâmetros, tempo, planta, EC e pH para continuar o ciclo. Para tal, é feita a leitura da hora e do vetor, separando os valores e associando as suas variáveis globais.

No primeiro fragmento, é feita a seleção da etapa, o que na programação é feita com um switch case, relativo à plantação (Apêndice 3.3). No segundo fragmento, situa-se a leitura e alojamento do vetor nas variáveis como está descrito nos comentários referentes a cada linha de código Apêndice 3.4. O terceiro fragmento é a leitura da data e horário. Este processo começa por receber o formato de hora, minuto, segundo, dia, mês e ano como se segue “hhmmssddMMaaaa” sendo colocado cada caracter numa variável. Uma vez que as variáveis são independentes, foi aplicada a álgebra para que cada número representativo de hora, minuto, segundo, dia, mês e ano, fiquem alocados numa só variável. Para isso multiplica-se cada primeira variável por 10 e adiciona-se a segunda variável. Por exemplo, 15 horas, onde 1(primeira variável) e 5 (segunda variável) será igual a $1 * 10 + 5$. Para o formato de ano, como são quatro variáveis, a primeira será multiplicada por 1000, a segunda por 100 e a terceira por 10, no fim são todas somadas (Apêndice 3.5).

Posto isto, dá-se por concluída a etapa de semente, ficando em registo os dados principais de EC, pH, tipo de planta, tempo de crescimento, data e hora de início.

4.3.2 Reservatório Cheio

Para dar continuidade ao processo é necessário ter o reservatório cheio de água. Para isso, colocou-se o sensor de EC numa posição estratégica para medir a capacidade máxima de 5L e assim facilitar cálculos associada à quantidade de nutrientes. Como tal, usa-se apenas o Arduino Micro para fazer essa gestão com a abertura da electroválvula, que permite o enchimento do depósito.

4.3.3 Controlo Luminosidade

Após o depósito estar cheio, faz-se o controlo de luminosidade. Tendo em conta que a planta deve ter um tempo de repouso sem iluminação incidente, foi considerado a

utilização de iluminação artificial, isto é, se a partir das 07h00 não existir luz solar suficiente para um crescimento sustentado, é ligada a lâmpada e assim permanece até que a luz solar atinja os 200lux. A partir das 21h00, este plano é desativado, assim o nível de luminosidade será baixo dando tempo à planta para repousar.

Toda esta gestão é realizada na sua grande maioria pelo Arduino Micro, existindo apenas um botão para ligar a lâmpada na página MANUAL para uma eventual exceção onde o utilizador pretenda ligar a luz artificial, fora das condições definidas.

Na Figura 31 é ilustrado a interface gráfica desta secção do projeto está ligado com a base de dados, uma vez que o botão da lâmpada fica ou não disponível com o facto de o modo automático estar ou não estar ativo. Isto é, para que a lâmpada possa ser acessível é necessário que o modo automático esteja desativado e consequentemente o modo Manual fica ativo. Só após esta operação é possível, ao utilizador, ligar/ desligar a lâmpada através do botão Desligar.

The image shows a web application interface for 'HIDROPONIA DE INTERIOR'. At the top, there is a navigation bar with links: 'Início', 'Plantar', 'Dados do Interior', 'Acção Manual', and 'Níveis de Líquidos'. The 'Acção Manual' section is active, displaying an 'Editar' link. Below this, there are two input fields: 'EC' and 'pH', each with a green 'INSERIR' button. A red rectangular box highlights the 'Lâmpada' control area, which contains a green 'Desligar' button and a green 'Modo Automático' button.

Figura 31 - Interface para Ligar/Desligar Lâmpada

Nesta secção indicada a vermelho existem dois botões, em que cada um tem dois estados. Na primeira é possível ligar ou desligar a lâmpada de espectro completo, no entanto assim que esse botão for acionado (o texto no interior alterna entre Desligar/Ligar), o controlo automático de luminosidade é desligado, fazendo o *update* à BD (\$wpdb->update).

O Arduino Micro faz a análise das seis variáveis para o controlo de luminosidade, nível de luz Baixa ou Alta, Dentro de Horas ou Fora de Horas e o modo manual (Apêndice 3.9) ou modo automático. Se o utilizador estiver em modo manual, é ele que controla quando a lâmpada está ou não ligada. Caso o modo automático (Apêndice 3.10) esteja ativo, compara os níveis de luminosidade no horário, caso este se encontre entre as 7 horas e as 21 horas e a luminosidade estiver baixa, este acende a lâmpada, caso contrário, a lâmpada permanece desligada. Fora desse horário, a lâmpada fica desligada para as plantas poderem realizar o processo de respiração.

4.3.4 Controlo de EC

Após colocar a semente, impõe-se a definição de um valor de EC inicial que pode ser aumentado ao longo do tempo (Figura 1). Para esse efeito, foi colocada uma secção na página Ação Manual (Apêndice 3.11).

The image shows a web application interface for 'HIDROPONIA DE INTERIOR'. The top navigation bar includes links for 'Início', 'Plantar', 'Dados do Interior', 'Acção Manual', and 'Níveis de Líquidos'. The 'Acção Manual' section is active, showing an 'Editar' link. The main content area features two input fields: 'EC' and 'pH'. The 'EC' input field is highlighted with a red border, and its corresponding 'INSERIR' button is also highlighted. Below these fields, the text 'Lâmpada' is displayed. At the bottom, there are two buttons: 'Desligar' and 'Modo Automático'.

Figura 32 - Interface para aumentar EC

Em primeiro lugar, é verificado se a tabela OUTPUT está ou não vazia, se estiver indica que não existe plantas na estufa pelo que retorna essa mesma informação ao utilizador, caso não esteja vazia, assim que o botão inserir é pressionado com o valor na caixa de texto, este faz a comparação entre o valor previamente existente na tabela. Se o novo valor estiver entre o valor anterior e 3 (valor máximo de EC permitido) este atualiza a linha da tabela. Caso contrário, avisa o utilizador para inserir um valor dentro dos parâmetros.

De seguida é feita a recolha da temperatura (Apêndice 3.12) para que posteriormente, nos cálculos da condutividade seja feita a compensação entre os 25°C (valor nominal) e o valor real. Este valor é obtido através do pino analógico T-out existente no módulo de pH.

Depois é feita a leitura dos valores de EC (Apêndice 3.13), para isso é realizada, alternadamente, a leitura nos dois lados da ponta de prova (5 vezes cada uma) com rotação dos pinos como na função *EC_read*, após a leitura é feita uma média para ser enviada para a função *checkEC*, onde são aplicados os cálculos necessários para fazer corresponder os valores do ADC à condutividade elétrica.

Por fim, pode ou não ser usada a terceira parte que corresponde ao adicionar nutrientes no reservatório (Apêndice 3.14), caso as medições assim o indiquem, para isso é acionado cada um dos motores peristálticos durante 300ms que adicionavam sensivelmente 0.5ml de nutrientes. Também é usado um contador para que, cada vez que, esses motores sejam ativados, faça uma transição do led RGB correspondente de verde (Cheio) para vermelho (Vazio) para que o utilizador tenha também uma melhor perceção da quantidade de nutrientes disponíveis em cada frasco.

4.3.5 Controlo de pH

Com uma interface representada na Figura 33 - Interface para diminuir pH percebe-se que processo é muito similar ao anterior, com a diferença que em vez de ter a possibilidade de ser aumentado, este tem apenas a possibilidade de ser reduzido.

HIDROPONIA DE INTERIOR

[Início](#)[Plantar](#)[Dados do Interior](#)[Acção Manual](#)[Níveis de Líquidos](#)

Editar

ACÇÃO MANUAL

EC

EC

INSERIR

pH

pH

INSERIR

Lâmpada

Desligar

Modo Automático

Figura 33 - Interface para diminuir pH

Em tudo muito idêntico ao caso anterior, o valor inserido pelo utilizador tem de ser igual ou inferior ao que estava na base de dados (Apêndice 3.15).

É recomendado que o valor de pH esteja sempre entre os valores 5.6 e 6.5, para que o crescimento seja saudável. Para tal, faz-se uma média de 10 medições como representado na função *checkPH* (Apêndice 3.16). Se a solução no reservatório não estiver dentro dos parâmetros definidos é adicionado uma solução ácida até atingir o pH desejado.

4.3.6 Bombear Solução

Neste ponto é apenas feita a renovação da água existente no tubo das plantas, por nova solução existente no reservatório. Caso não exista qualquer tipo de alteração na base de dados, este processo deve ser repetido a todos os minutos 15' e 45' de cada hora (Apêndice 3.17).

Devido ao facto da calha ter uma saída muito pequena, foi necessário fasear o processo de troca de solução. Deste modo, criou-se um ciclo que se repete por 10 vezes fazendo a bomba trabalhar 5 segundos e esperar 10 segundos para que a solução no interior da calha tenha tempo para sair.

4.3.7 Escrever valores na base de dados

Concluído o processo de renovação de solução, são enviados para a base de dados todos os valores que foram medidos pelos sensores. Para tal, o Arduino Micro envia (Apêndice 3.18) uma “frase” para a RPi com um carácter de iniciação que foi definido como ‘a’, de seguida o EC (*ACTUALEC*), pH (*ACTUALPH*), luminosidade e temperatura.

No *script* (Apêndice 3.19) é feita a leitura da porta série do RPi (Dados enviados pelo Arduino Micro), rejeitando toda a informação até que seja obtido o carácter ‘a’ e assim que isso se suceder inicia-se um ciclo para ler as seguintes 4 entradas do *buffer* e colocar tudo num *array* ficando este com o seguinte formato [‘a’, EC, pH, luminosidade, temperatura]. Seguidamente é separado esse *array* em variáveis independentes para serem adicionadas na tabela INPUT fechando assim o processo de inserção de dados.

4.3.8 Sleep Mode

Para melhorar o rendimento da estufa a nível energético, foi equacionado o colocar do Arduino Micro em Modo Standby para que este não estivesse a consumir energia enquanto estiver na fase de repouso. Foram então implementadas 2 interrupções (Apêndice 3.20) externas em 2 pinos. Um que está conectado ao pino digital do módulo de LDR (que foi ajustado para o seu potenciómetro para a luminosidade mínima para um crescimento sustentado) e o segundo a um pino da RPi que ativa sempre que existe alguma alteração na BD ou atinge o limite o minuto 15’ ou 45’ de cada hora. Este processo é feito pelo código *python* no Apêndice 3.21.

Em ciclo permanente, o script vai ler a única linha da tabela *OUTPUT* e se algum dado for diferente do que estava previamente definido, vai ativar o pino 21 da RPi, fazendo ativar a interrupção. No caso da colheita ter sido feita pelo utilizador, o script faz um reset ao Arduino Micro.

Após acionada uma das interrupções, vai ser ativada a respetiva função que colocará a variável global *INTSTATUS* a 1 caso a interrupção seja feita pelo pin referente ao utilizador (Utilizador alterou algum valor na página web) ou a 2 no caso do LDR ter detetado uma alteração de luminosidade. Termina o repouso do microcontrolador e desativa as interrupções até que a função seja chamada de novo.

4.3.9 Ler Base de Dados

Se a interrupção acionada foi a que está ligada ao *USER* (Apêndice 3.21) significa que houve uma alteração na BD ou chegou o minuto 15'/45'. Com isso, através do Python, é feita a comparação e enviado numa só linha, um *array* com toda a informação para a porta série (Apêndice 3.22). Este *array* irá conter por esta ordem: ['a' EC PH Automático Lâmpada] e o código referido pelo Apêndice 3.22 mostra como é decomposto pelo Arduino Micro.

Num formato muito idêntico ao inicial, de quando se coloca a semente, é colocado numa *string* (frase composta pelos dados transferidos) toda a informação e decomposta em parcelas sempre que um espaço em branco for reconhecido na *string*. Ao concluir o processo estão disponíveis novos valores nas variáveis globais *STARTEC*, *STARTPH*, *AUTOMATIC*, *LIGHT* (Apêndice 3.23).

4.4 Esquema Ensaio 1

Na Figura 34 estão representadas as ligações existentes entre o Arduino Micro e respetivos sensores e atuadores. De salguardar que, a comunicação série é feita através do cabo USB/microUSB ao mesmo tempo que alimenta o microcontrolador.

4.5 Conclusões sobre Ensaio 1

Após algum tempo de utilização foram detetadas falhas no funcionamento do sistema, como perda de sincronia entre o Arduino Micro e a RPi, os dados recolhidos com erros e inseridos na BD, pelo que foi decidido alterar a estratégia de controlo para a gestão de funcionamento na RPI como descrita no Ensaio 2.

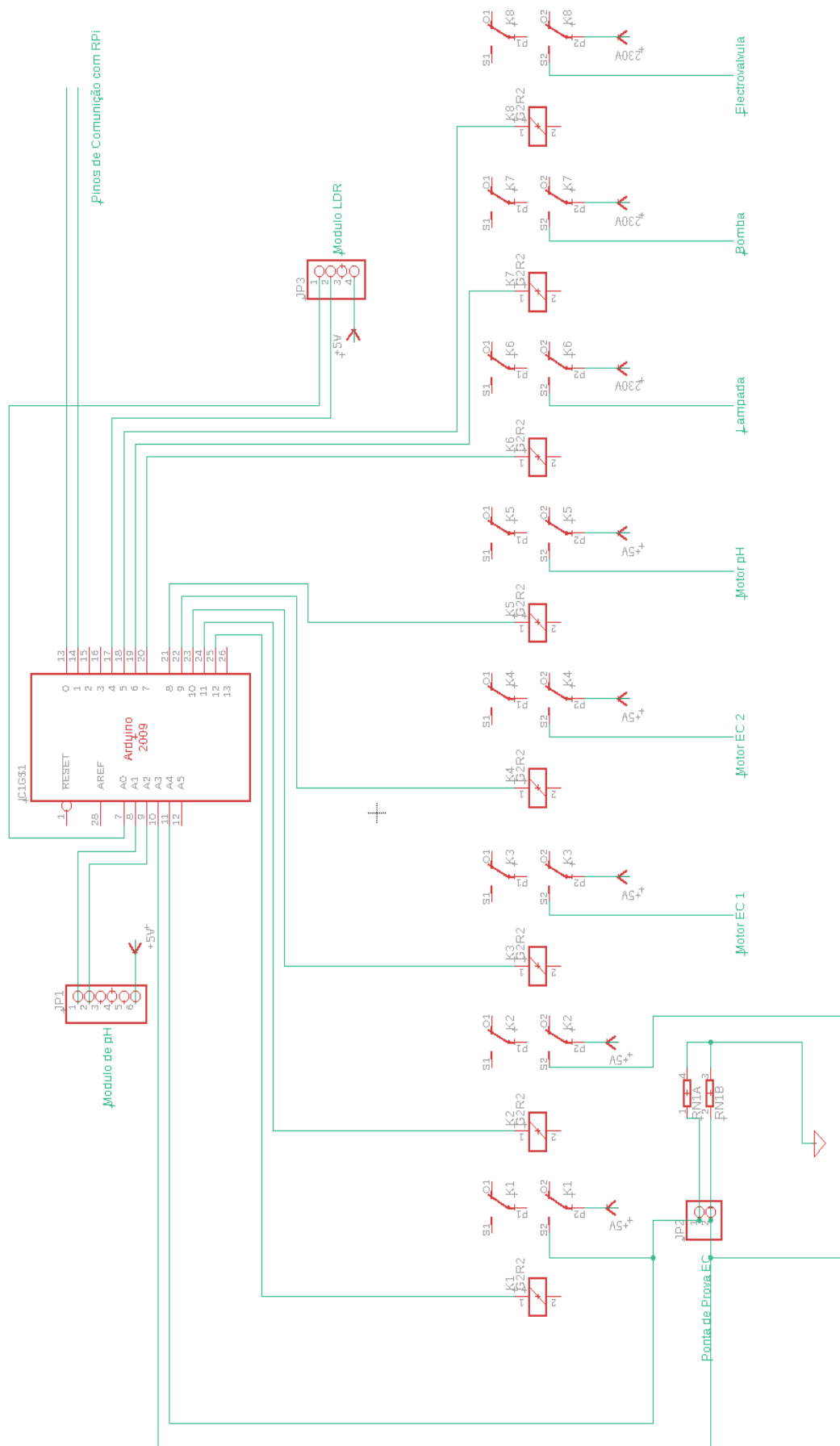


Figura 34 - Esquema de ligações ao Arduino

4.6 Ensaio funcionamento com gestão na Raspberry Pi – Ensaio 2

Este segundo Ensaio tem um modo de funcionamento muito idêntico ao anterior, sendo que aqui é o script *Python* a descrever a rotina que está representada no esquema seguinte:

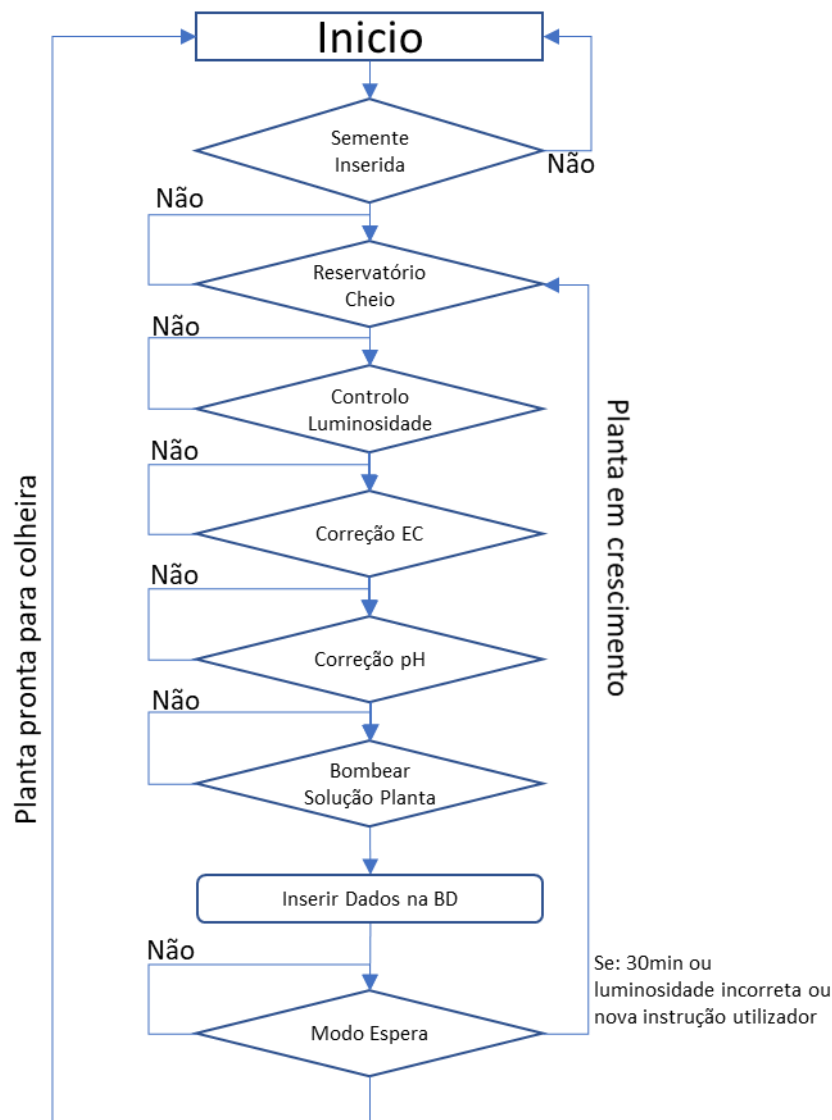


Figura 35 - Fluxograma representativo da rotina ensaio 2

Com o servidor (RPi através do *script* em *python*) a fazer a mesma gestão de todo o sistema, pode-se suprimir a etapa de leitura de dados, pois este está sempre a ser lido em cada etapa representada no fluxograma acima descrito. No que toca ao plano de WEB e base de dados, estes mantêm-se iguais ao Ensaio anterior, pelo que apenas se descreverá o processo realizado entre o Arduino Micro e o Servidor (RPi).

O código criado que representa o fluxograma anterior implementa uma máquina de estados. Em cada estado é chamada uma função que verifica se é obtida a condição de troca de estado. Em algumas destas funções é usada a biblioteca *Json*, uma ferramenta muito útil para comunicar entre dispositivos pois usa uma nomenclatura própria que permite ao utilizador recolher, enviar e criar a informação a ser transmitida no *buffer* da porta série.

Nomenclatura esta que tem este formato:

```
data = { "CMD": 2, "day": 1 }
```

É criada uma variável (neste caso com o nome “data”) e dentro das chavetas encontra-se, entre aspas o nome de uma variável que se pretende enviar ou receber. Após os dois pontos coloca-se o conteúdo da variável. Este tipo de dados pode ser *INT*, *CHAR*, *STRING* ou *ARRAY*. Mostra-se de seguida um excerto da máquina de estados (Código completo em Apêndice 3.24):

```
def StateMachine():  
    global STATE  
    if STATE is 1:  
        STATE = WaitSeed()  
  
    if STATE is 2:  
        STATE = FillRes()  
  
    if STATE is 3:  
        STATE = CheckLDR()
```

No que toca à comunicação com o Arduino Micro, sempre que existe uma transmissão, é enviada uma *string* como acima descrito na qual é incluído obrigatoriamente o comando (“CMD”: 1-8) e ainda informação complementar para o Arduino Micro ter referências aos valores de EC, pH, hora, luminosidade, etc.

Fazendo o comparativo entre a Figura 35 e o código python acima, pode-se associar cada bloco com um “if STATE is N:”, isto é, a cada etapa do fluxograma corresponde um estado na máquina de estados como é representado na Tabela 13.

Tabela 13 - Comparação entre blocos e função de estado

Bloco	Estado (Python)
Semente Inserida	if STATE is 1: STATE = WaitSeed()
Reservatório	if STATE is 2: STATE = FillRes()
Controlo Luminosidade	if STATE is 3: STATE = CheckLDR()
Correção EC	if STATE is 4: STATE = ControlEC() if STATE is 5: STATE = AddEC()
Correção pH	if STATE is 6: STATE = ControlpH() if STATE is 7: STATE = AddpH()
Bombear Solução Planta	if STATE is 8: STATE = BombSolution()
Inserir Dados na BD	if STATE is 9: STATE = InsertBD()
Modo Espera	if STATE is 10: STATE = WaitChange()

Essa leitura do comando, no Arduino Micro, é executada através de:

```
while ( !Serial.available() )
{
}
if ( Serial.available() )
    payload = Serial.readStringUntil( '\n' );

DeserializationError error = deserializeJson(doc, payload);
if (error) {
    Serial.println(error.c_str());
    return;
}
```

Caso a *string* recebida não seja erro, procede-se a comparação da variável existente em “CMD” (Código completo em Apêndice 3.25):

```
if (doc["CMD"] == 1) {

    Serial.println(payload.length());
    delay(50);
    fillRes();
}
if (doc["CMD"] == 2) {
    Serial.println(payload.length());
    delay(50);
}
```

```
day = doc["day"];
light = doc["light"];
LDR(day, light);
}
```

De cada uma destes estados é chamada uma função como será descrito mais a frente.

4.6.1 Semente inserida

Na forma inicial, o script (Apêndice 3.26) fica em ciclo a fazer a leitura da tabela *OUTPUT* da base de dados, até que ali surja alguma informação nova. Assim que existir, é sinal que o utilizador inseriu, através da página, uma nova semente.

É obtida a única linha existente na tabela *OUTPUT* e é transmitida ao Arduino Micro. Este armazena esta linha (*string*) que será decomposta num *array*, de onde se obtém então os valores de EC, pH e o estado da lâmpada assim como o modo de funcionamento (Automático/Manual).

4.6.2 Reservatório Cheio

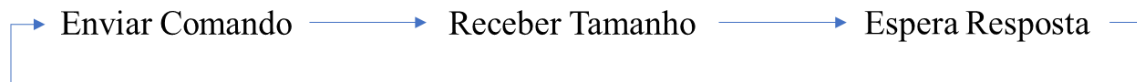
Neste ponto inicia-se a comunicação com o Arduíno. Começa por enviar o comando “CMD: 1” da RPi para o Arduino Micro (Apêndice 3.27) e espera seja retornado o tamanho da mensagem enviada. No caso de retornar um valor igual ao tamanho da mensagem, o programa prossegue. Se não existir igualdade, este vai ficar em *loop* (ciclo) até que os dois dispositivos se sincronizem.

Assim que se concluir a tarefa, o Arduíno lê o comando (neste caso “CMD: 1”, referente à etapa de enchimento do reservatório) e executa a função para enchimento do tanque (Apêndice 3.28). Enquanto o reservatório enche, o Arduino Micro escreve na porta série a mensagem com o resumo da condição, isto é, se o depósito estiver cheio “*Ready*”: 1, se não, “*Ready*”: 0. Enquanto isto, o lado do servidor fica à espera de resposta (Apêndice 3.29)

Assim que a resposta for “*Ready*”: 1 este vai dar como concluído o passo para encher o reservatório de água, seguindo para a nova etapa.

4.6.3 Controle Luminosidade

Assim como no processo de enchimento do reservatório, o código desta etapa terá o mesmo processo:



Contudo, neste caso, para além do comando “CMD” também é enviado o estado da lâmpada e a informação sobre o período diurno (7h e 21h). Então a resposta do Arduino não será apenas uma variável booleana, mas também um *array* de 50 inteiros, dos quais irá ser calculada uma mediana através da função “statistics.median()” e apresentada ao utilizador. Ficando em variável global esse mesmo valor para posteriormente ser carregada para a base de dados (Apêndice 3.30).

Em primeiro lugar, é verificado se está em modo automático e no período diurno (*day* = 1) e ligar/desligar a lâmpada consoante os valores do LDR. Caso o modo manual esteja ativo vai ler o estado de “*light*” e ligar/desligar a luminária de espectro completo.

Por fim é criado o *array* de 50 inteiros e enviado com o comando “*Ready*” = 1. Assim que os 50 valores sejam recebidos, é feita a conversão, ficando o resultado guardado em READ_LDR (Apêndice 3.31).

4.6.4 Controle de EC

Mais uma vez, o processo de transmissão segue um processo idêntico aos demais com a particularidade que neste ponto, depende dos valores recebidos para ou chamar a função de leitura de pH ou a de adicionar nutrientes ao reservatório

No Arduino Micro são feitas as medições de EC (Apêndice 3.31). Estes cálculos surgem através da recomendação do próprio construtor do sensor de EC, GravityTDS⁴⁰

Por fim, faz-se o retorno de um *array* de 30 medições de temperatura e outro com 30 medições de TDS (*Total Dissolved Solids* – Nutrientes dissolvidos na água)⁴¹ que é

⁴⁰ https://wiki.dfrobot.com/Gravity__Analog_TDS_Sensor__Meter_For_Arduino_SKU__SEN0244

⁴¹ http://hmdigital.com/knowledge_base/the-relationship-between-tds-and-electrical-conductivity/

convertido em EC pelo *python*, associado a estes irá também o comando de função concluída “*Ready*”: 1. Caso seja necessário adicionar nutrientes à solução, é feita da com o código Apêndice 3.33

Fazendo ligar cada um dos motores por 2 segundos, é adicionado 2ml de solução nutritiva. De salguardar que neste Ensaio, foram usadas bombas submersas para enviar os nutrientes para o reservatório (motores peristálticos não foram devidamente protegidos, o que causou danos no seu funcionamento).

4.6.5 Controlo de pH

Neste ponto, o processo é exatamente igual ao EC sendo apenas feitos os ajustes necessários para enviar o comando 6, ler o pino do módulo de pH e acionar (caso necessário) o ácido ao reservatório.

4.6.6 Bombear Solução para as Plantas

Este processo é uma junção de duas partes, já implementadas neste processo, no lado de *python* o enviar o comando 7 para fazer a mudança da água no tubo de crescimento. Dado o pormenor de a espera de resposta ser maior que o normal, implementou-se um tempo de espera de 150 segundos, com uma barra de progresso a ser apresentada ao utilizador (Apêndice 3.34). Esses 150 segundos devem-se ao facto da rega se dar de forma faseada (10 vezes de 15 segundos) para não exceder a capacidade do tubo e dar tempo de escoamento (Apêndice 3.35).

4.6.7 Escrever na Base de Dados

O passo de escrever na base de dados é muito idêntico ao do Ensaio 1, os dados são alocados em variáveis globais e chamados para esta função já definidos, sendo estes *Read_EC*, *Read_PH*, *Read_LDR*, *Read_Temp* que são, respetivamente o valor lido de EC, pH, luminosidade e temperatura (Apêndice 3.36)..

4.6.8 Tempo de Espera

Um novo ciclo do processo do sistema é iniciado a cada minuto 15’ ou 45’ de cada hora caso não exista nenhuma alteração na base de dados ou a luminosidade tenha uma variação maior que o normal. O tempo de espera ocorre no final do ciclo e existe para

criar uma pequena poupança nos recursos energéticos e ao mesmo tempo não criar um ruído constante. Quanto à luminosidade, é colocada na variável *NowLDR* a leitura do sensor LDR e comparada com o último valor inserido na tabela INPUT. Se ambos estiverem acima ou abaixo de 200 (valor mínimo medido para indicar a luminosidade durante o horário diurno), não existe qualquer tipo de alteração. Caso exista diferença, todo o ciclo é recommençado (Apêndice 3.37).

Seguindo o código em C para a leitura do LDR, caso a leitura seja 0 (Maior luminosidade possível) os cálculos iriam ser indefinidos pois este valor é usado como denominador numa divisão. Assim sendo, é enviado o valor de 1 que pouco altera os cálculos e salvaguarda a conclusão da conversão (Apêndice 3.38).

Para a alteração executada pelo utilizador é feita uma leitura constante da base de dados e comparada com as variáveis globais lidas no início do ciclo, se alguma delas for diferente, é executado um novo ciclo, agora com os novos valores associados (Apêndice 3.39).

Por fim, caso nenhuma destas condições anteriores ocorram, é o fim do tempo de espera que faz recommençar o ciclo (Apêndice 3.40).

a

5 Conclusões

Após a realização de pesquisas sobre Hidroponia em vários *sites* e acompanhando alguns produtores, foi concluído que a produção hidropónica é um método de cultivo muito útil devido ao facto de ser mais fácil controlar todos os fatores externos existentes garantindo uma produção mais segura. Tendo o ambiente controlado é sempre mais seguro suprimir pragas, acompanhar o crescimento das plantas e ainda conseguir reduzir a quantidade de água necessária para o crescimento das culturas.

A utilização de um servidor Web permite ajudar, em tempo real, o utilizador a ser informado caso alguma anomalia ocorra na estufa e então, de alguma forma, minimizar possíveis estragos ou demoras nas produções.

Como resultado, foi então criada uma estufa de pequenas dimensões com comprovada capacidade produção de plantas. Todo este sistema precisa apenas de uma ligação à tomada de 230V, para ligar o circuito elétrico e uma entrada de água para manter o reservatório cheio. Toda a comunicação entre o *Arduíno* e o *Raspberry* é feita através de um cabo USB2.0 – MicroUSB. A ligação à *Web* foi feita pela *Raspberry*. Toda a interface *Web* e base de dados está apenas disponível para todos os dispositivos ligados ao Router residencial.

No decorrer dos Ensaios foram surgindo alguns problemas, entre eles, de salientar as medições de EC e a comunicação série entre o *Arduíno* Micro e RPi que foram resolvidos na implementação do Ensaio 2 e com o uso de um medidor de EC comercial.

Começando pelo EC, na ponta de prova inicial existiu o problema de os sais se depositarem numa das placas, pelo que foi necessário polarizar nos dois sentidos os placas de medição para anular esse problema.

No entanto ao longo de várias medições, as leituras que o ADC estava a obter eram diferentes de cada vez que o *Arduíno* era desligado e ligado de novo, pelo que se tornou necessário usar um sensor disponível no mercado, que envia uma onda quadrada alternada e as leituras deviam ser filtradas para que fossem as mais corretas possíveis. Daí surgir então o medidor de EC 2, um módulo que permitiu obter valores mais estáveis ao longo do tempo.

Em relação à comunicação série, o Ensaio 1 fazia com que o Arduino fosse colocado em modo poupança de energia, o que no processador ATmega34U4, fazia com que os pinos de interrupções algumas vezes não ligassem e como o programa não estava concebido da melhor forma, este não se sincronizava com o *RPi*, reportando leituras sem informação. Por isso foi então aplicado o Ensaio 2, onde o Arduino não entra em modo de poupança (13.92mA em estado ativo e 6.2μA *Standby*) e com as devidas proteções em código tanto do lado do Arduino como no lado da *RPi*, conseguiu-se suprimir esses erros e reduzir o código uma vez que o servidor tem de estar forçosamente ligado 24 horas por dia, para que a página se mantenha acessível ao utilizador.

Realçando alguns dos pontos que podem ser melhorados, uma vez que este é um projeto em evolução, é aconselhável incorporar mais sensores para controlar temperatura da água, humidade e até controlo de pragas como recurso a análise de imagens. Isto é, fazer melhorias no que respeita à gestão direta do crescimento em vez da interface com o utilizador, ponto onde mais se prestou atenção, para que exista uma comunicação mais segura e atrativa.

Este é um projeto que com poucas alterações se pode incorporar em qualquer casa que queira ter a própria “horta”, mesmo em casos como apartamentos. Podendo de uma forma limpa e sem grandes complicações acompanhar o crescimento das próprias culturas.

Seria também interessante, num próximo passo, incorporar um aquário com peixes para evitar o uso de nutrientes fertilizantes e criando assim uma estufa hidropónica biológica, uma vez que as fezes dos animais têm os nutrientes precisos para um bom crescimento das plantas além de dar uma maior beleza à estufa.

Completando todo o projeto, este permanece em execução tendo ainda sido adicionado um acesso com uma *WEBCAM* que pode ser acedida pelo próprio servidor, e que permite ter um acompanhamento em tempo real de toda a estufa. Importa também salientar que todos os passos são mostrados na *SHELL* do *python* (Apêndice 4.10 - Imagem da *SHELL* do *Python* com rotina completa).

Referências Bibliográficas

- [1] L. Rebelo, “Hidroponia em Ambiente Controlado: desenvolvimento de uma metodologia de análise ao desempenho produtivo e aos consumos de produção. Retrieved,” 2015. [Online]. Available: https://repositorio.ul.pt/bitstream/10451/22292/1/ulfc116052_tm_Lu%c3%ads_Rebelo.pdf.
- [2] F. G. A. Blidariu, “Increasing the Economical Efficiency and Sustainability of Indoor Fish Farming by Means of Aquaponics,” *Animal Science and Biotechnologies*, p. 44 (2), 2011.
- [3] E. Folds, “The History of Hydroponics,” 2018. [Online]. Available: <https://medium.com/@evanfolds/the-history-of-hydroponics-99eb6628d205>.
- [4] C. (. Teodoro, “History of Hydroponics: When Was Hydroponics Invented?,” [Online]. Available: <https://www.epicgardening.com/history-of-hydroponics/>.
- [5] O. Stephens, “the hydroponics planet,” 2019. [Online]. Available: <https://thehydroponicsplanet.com/>.
- [6] F. Greenhouse, “Hydroponic Systems,” [Online]. Available: <https://www.fullbloomgreenhouse.com/hydroponic-systems-101/>.
- [7] H. e. Portugal, “A Hidroponia em Portugal,” 2016. [Online]. Available: <https://hidroponiaempotugal.blogs.sapo.pt/hidroponia-em-portugal-1009>.
- [8] GroHo. [Online]. Available: <https://www.hidroponiabrasil.com/post/o-que-e-a-condutividade-eletrica>. [Acedido em 20 06 2021].
- [9] Hidrogood, 12 02 2019. [Online]. Available: <https://hidrogood.com.br/noticias/fertilizantes/o-que-e-condutividade-eletrica-e-a-sua-importancia-no-cultivo-hidroponico>. [Acedido em 20 06 2021].
- [10] M. Z. Beckmann, “Radiação solar em ambiente protegido cultivado com tomateiro nas estações verão-outono do Rio Grande do Sul,” 24 08 2005.

Apêndices

Nesta seção são apresentados alguns apêndices que permitem enquadrar o trabalho desenvolvido.

Apêndice 1 Fotografias do sistema desenvolvido da estufa completa



Figura 36 - Foto do andar superior da estufa





Figura 40 - Raspberry Pi

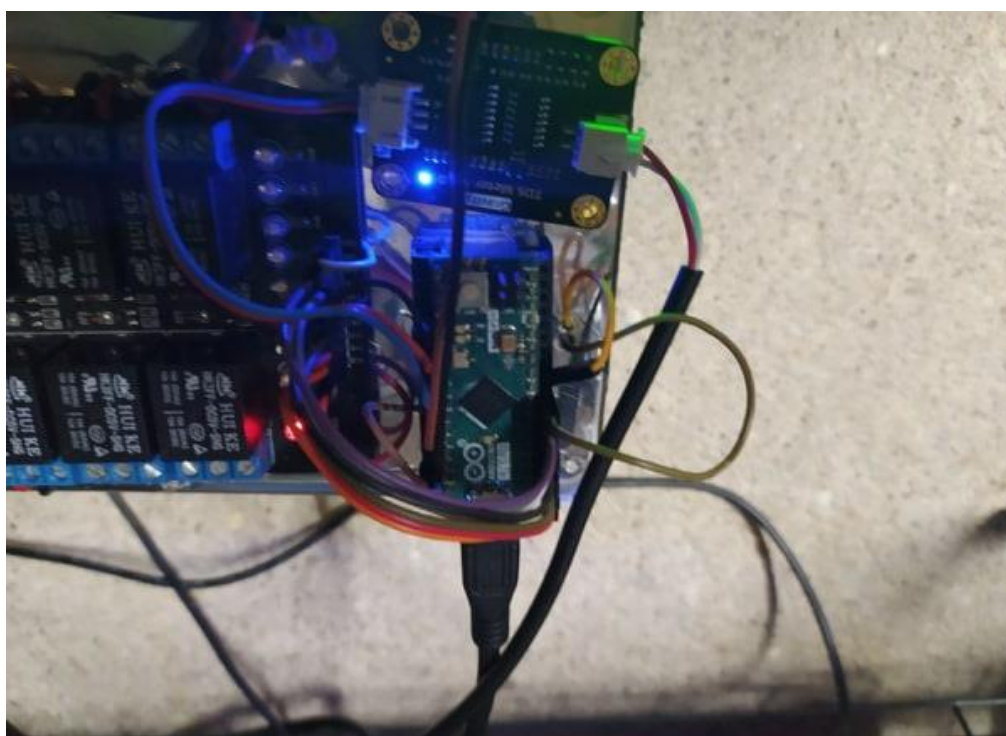


Figura 41 - Arduino Micro

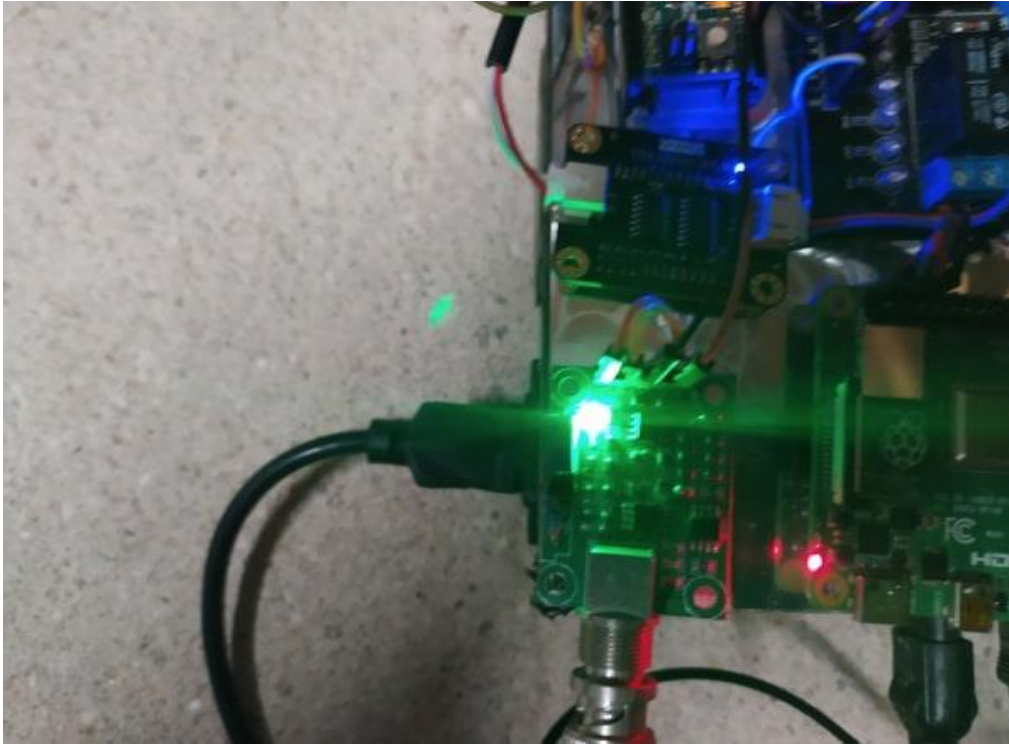


Figura 42 - Módulo de pH (Sem ponta de prova)

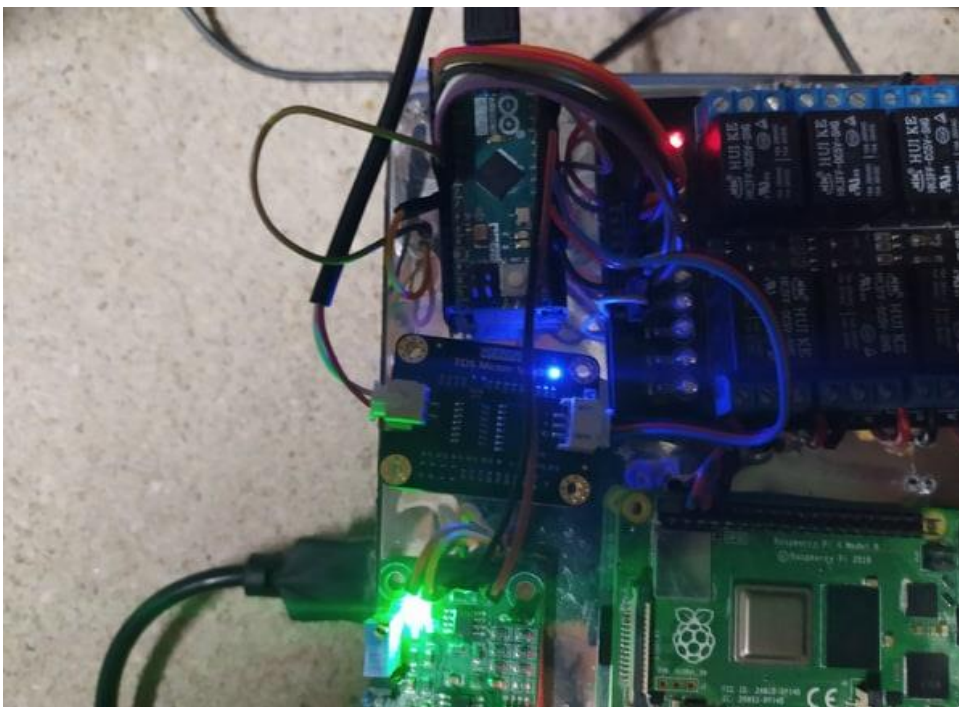


Figura 43 - Módulo de EC (Sem ponta de prova)

Apêndice 2 Imagem da SHELL do Python com rotina completa

```
Porta Usada: ACM0
-----DADOS DO UTILIZADOR-----
Planta: alface
EC: 1.5
Dias de Crescimento: 30
-----DADOS MEDIDOS-----
ENCHER RESERVATORIO: OK
Luminosidade em Modo Automatico e fora de horas
lampada Desligada|
LDR 1.05
Control Luminosidade: OK
Valor EC: 1.55
Confirmar Valor de EC: 1 / 5
Valor EC: 1.55
Confirmar Valor de EC: 2 / 5
Valor EC: 1.53
Confirmar Valor de EC: 3 / 5
Valor EC: 1.54
Confirmar Valor de EC: 4 / 5
Valor EC: 1.55
Confirmar Valor de EC: 5 / 5
Valor EC: 1.55
Confirmar Valor de EC: OK
pH: 5.64
Control pH: OK
A Repor Solucao nas plantas ...
[0%:::10%:::20%:::30%:::40%:::50%:::60%:::70%:::80%:::90%:::100%]
Repor Solucao nas plantas: OK
Dados inseridos na BD: OK
```

Figura 44 - Exemplo da SHELL Python com ciclo da rotina

Apêndice 3 Código de suporte para explicação dos blocos de rotina

Neste apêndice está presente o código de apoio para a explicação da rotina de funcionamento do sistema, em web, Python e C/C++.

Apêndice 3.1 Ensaio 1 Código web para inserir semente

```
<style>
h1 {font-size:30px;color:#00cc00;text-align:center;font-weight:bold;}
h2 {font-size:20px;color:#00cc00;text-align:center}
h3 {font-size:18px;color:#000000;text-align:center;font-weight:bold;}
p1 {color:black;}
p2 {font-size:30px;color:#000000;text-align:center;}
.SEED > div {text-align:center;}
</style>
<body>
<h1> Semente a Plantar </h1>
<br>
<br>

<div style="width:32%;padding:0 10pt 0 0;float:left;text-align:center; ">
<form method="Post">
```

```

<button type="submit" id="alface" name="alface" value="" style="cursor:pointer;
background-color:white; border:none"></button>
<?php
global $wpdb;
if(isset($_POST['alface']))
{
$ready = $wpdb->get_row( "SELECT * FROM `OUTPUT` WHERE id=1", ARRAY_A);
if($ready)
echo "Estufa em utilização";
else{
$result = $wpdb->insert('OUTPUT', array('id'=>'1','PLANTA'=>'alface',
'LAMPADA'=>'0', 'EC'=>'1.5', 'PH'=>'5.5', 'DIAS_CRESCIMENTO'=>'30',
'MODO_AUTOMATICO'=>'1'));
echo 'Planta Inserida';
$result = $wpdb->insert('LEVEL', array('id'=>'1','EC1'=>'100', 'EC2'=>'100',
'pH'=>'100'));
}
}
?>
</form>
</br>
<p2> Alface </p>
</div>

```

```

<div style="width:100%;padding:0 10pt 0 0;float:left;text-align:center;">

```

```

</br>

```

```

</br>

```

```

<h2>INSERIR NOVA SEMENTE</h2>

```

```

</br>

```

```

<form action="" method="Post" >

```

```

<input type="text" name="SEED" size="30" placeholder="Nome da Semente" /> <input
type="number" min="0" max="3" onkeypress='return event.charCode >= 48 &&
event.charCode <= 57' name="EC" size="10" placeholder="EC (0-3)" step=".1"/>
<input type="number" min="4" max="8" name="PH" size="10" placeholder="pH (4-8)"

```

```
step=".1"/>    <input type="number" min="0" max="366" name="TIMEGROW"
style="width: 200px" placeholder="Dias de Crescimento" />
```

```
<button type="submit" id="new" name="new" value="" style="cursor:pointer;
background-color:white; border:none"></button>
```

```
<?php
```

```
global $wpdb;
```

```
if(isset($_POST['new']))
```

```
{
```

```
$ready = $wpdb->get_row( "SELECT * FROM `OUTPUT` WHERE id=1",
ARRAY_A);
```

```
if($ready)
```

```
echo "Estufa em utilização";
```

```
else{
```

```
$SEED= $_POST["SEED"];
```

```
$EC= $_POST["EC"];
```

```
$PH= $_POST["PH"];
```

```
$TIMEGROW= $_POST["TIMEGROW"];
```

```
$wpdb->insert('CULTURES', array('TYPE'=>$SEED, 'EC'=>$EC,
'DESCRIPTION'=>' ', 'PH'=>$PH, 'TIME'=>$TIMEGROW));
```



```
$result2 = $wpdb->insert_id;
```

```
echo $result2;
```

```
$result2 = $wpdb->insert('OUTPUT', array('id'=>'1','PLANTA'=>$SEED,  
'LAMPADA'=>'0', 'EC'=>$EC, 'PH'=>$PH,  
'DIAS_CRESCIMENTO'=>$TIMEGROW));
```

```
$result = $wpdb->insert('LEVEL', array('id'=>'1','EC1'=>'100', 'EC2'=>'100',  
'pH'=>'100'));
```

```
}
```

```
}
```

```
?>
```

```
</form>
```

```
</br>
```

```
</div>
```

```
</body>
```

Apêndice 3.2 Ensaio 1 Python Inicia rotina com semente inserida

while row is **None** and x is **0**:

```
dbConn = pymysql.connect(host="localhost",user="root",passwd="girafa",db="wordpress")  
cursor = dbConn.cursor()  
cursor.execute("SELECT * FROM OUTPUT ORDER BY id DESC LIMIT 1 ")  
row=cursor.fetchone()  
var1 = GPIO.input(20)
```

if row is not **None** and x is **0** :

```
row_str = str(row)  
row_str_aux = row_str.replace("'", "")  
row_str_final = row_str_aux.replace(",","")  
data_splited = row_str_final.split(",")
```



```

data_aux = (data_splited[1] + data_splited[9] + data_splited[11] + data_splited [12] + data_splited
[2])
data_final =data_aux.ljust(30, ' ')
EC = data_splited [9]
AUTOMATIC = data_splited [12]
PH = data_splited [10]
LIGHT = data_splited [2]
time.sleep(0.1)
arduino.write(time.strftime("%H%M%S%d%m%Y").encode())
print(time.strftime("%H%M%S%d%m%Y"))
time.sleep(0.1)
print(data_final.encode())
arduino.write(data_final.encode())
time.sleep(0.1)
x= 1

```

Apêndice 3.3 Ensaio 1 C/C++ Plantar semente Switch Case

```

case plantedSeed:
{
    int seedAccepted;

    seedAccepted = choosedSeed ();
    if (seedAccepted == 1)
    {
        pixels.setPixelColor(0, pixels.Color(0,255,0));
        pixels.setPixelColor(1, pixels.Color(0,255,0));
        pixels.setPixelColor(2, pixels.Color(0,255,0));
        stateMachineControler = openWater;
        break;
    }
    else
    {
        stateMachineControler = plantedSeed;
        break;
    }
}

```

Apêndice 3.4 Ensaio 1 C/C++ Leitura de variáveis iniciais

```

int choosedSeed ()
{
    int seed;
    char all_data[30];
    char c;
    char type[20] = "";
    char ec [10] = "";
    char days [10] = "";
    int j = 0;
    int k = 0;
    int date;
    int growDays;

    if (Serial.available())

```

```

{
    setRealTime();

    delay(30);
    while(Serial.available()==0)
    {}
    if (Serial.available())
    {
        delay(50);
        for(int i = 0 ; i<30 ; i++)
        {
            c = Serial.read();
            all_data[i]= c ;
            if ( c == ' ')
            {
                j++;
                k = 0;
            }
            if (j == 1)
            {
                type[k] = c;
                k++;
            }
            if (j == 2)
            {
                ec[k] = c;
                k++;
            }
            if (j == 3)
            {
                days[k] = c;
                k++;
            }
        }
        Serial.print("Planta: ");
        Serial.println(type);
        delay(300);
        STARTEC = (float)atof(ec);
        Serial.print("EC inicial: ");
        Serial.println(STARTEC);
        delay(300);
        growDays = (int)atof(days);
        Serial.print("Dias de Crescimento: ");
        Serial.println(growDays);
        return(1);
    }
}
else
{
    return(0);
}
}
}

```

```

void setRealTime (void)
{
    char readchar;
    char line[14];
    int hrr1,hrr2;
    int hr;
    int mtt1, mtt2;
    int mt;
    int sds1, sds2;
    int sd;
    int day1, day2;
    int dy;
    int mnt1, mnt2;
    int mn;
    int yea1, yea2, yea3, yea4;
    int ye;
    time_t t;

    while(Serial.available()!=0)
    {}
    for (int i=0; i<14; i++)
    {
        delay(100);
        readchar = Serial.read();

        if (i==0)
        {
            hrr1=int(readchar)-48;
        }
        if (i==1)
        {
            hrr2=int(readchar)-48;
        }
        if (i==2)
        {
            mtt1=int(readchar)-48;
        }
        if (i==3)
        {
            mtt2=int(readchar)-48;
        }
        if (i==4)
        {
            sds1=int(readchar)-48;
        }
        if (i==5)
        {
            sds2=int(readchar)-48;
        }
        if (i==6)
        {
            day1=int(readchar)-48;
        }
        if (i==7)
        {

```

```

    day2=int(readchar)-48;
}
if (i==8)
{
    mnt1=int(readchar)-48;
}
if (i==9)
{
    mnt2=int(readchar)-48;
}
if (i==10)
{
    yea1=int(readchar)-48;
}
if (i==11)
{
    yea2=int(readchar)-48;
}
if (i==12)
{
    yea3=int(readchar)-48;
}
if (i==13)
{
    yea4=int(readchar)-48;
}
line[i]=int(readchar);
}
hr = hrr1 * 10 + hrr2;
mt = mtt1 * 10 + mtt2;
sd = sds1 * 10 + sds2;
dy = day1 * 10 + day2;
mn = mnt1 * 10 + mnt2;
ye = yea1 * 1000 + yea2 * 100 + yea3 * 10 + yea4;
Serial.print(dy);
Serial.print("/");
Serial.print(mn);
Serial.print("/");
Serial.print(ye);
Serial.print(" ");
Serial.print(hr);
Serial.print(":");
Serial.print(mt);
Serial.print(":");
Serial.println(sd);
delay(10);
setTime(hr,mt,sd,dy,mn,ye);
}

```

Apêndice 3.6 Ensaio 1 C/C++ Enchimento de reservatório

case openWater:

```

{ //verificar aos terminais 26 e 28 se ha condutividade entre eles, caso, Fullreservoir = 0
  // temos entao o deposito cheio para começar a trabalhar
  // pinos passam a ser entradas de alta impedancia para nao criar conflito com os sensores
  int reservoir=0;
  pinMode(ECPin1,INPUT);

```

```

digitalWrite(ECPin1, HIGH);
pinMode(ECPin2, OUTPUT);
digitalWrite(ECPin2, LOW);
reservoir = digitalRead(ECPin1);
pinMode(ECPin2, INPUT);
pinMode(ECPin1, INPUT);

if (reservoir == HIGH)
{
    digitalWrite(ValveInReservoir, HIGH);
    stateMachineControler = openWater;
    Serial.println("Deposito Cheio: NO");
}
else
{
    digitalWrite(ValveInReservoir, LOW);
    Serial.println("Deposito Cheio: OK");
    stateMachineControler = lightController;
    delay(2000);
}
break;
}

```

Apêndice 3.7 Ensaio 1 Código web para controlo lâmpada

```

<div style="width:90%;padding:0 10pt 0 0;float:left;text-align:center; ">
<h1> Lâmpada </h1>
</br>
<form action="" method="Post" >
<button type="submit" id="change3" name="change3" style=" background-color:
#4CB62C; border: none; color: white; padding: 15px 32px; text-align: center;
text-decoration: none; display: inline-block; font-size: 16px; margin: 4px
2px; cursor: pointer;"><?php global $wpdb; $ready3 = $wpdb->get_row( "SELECT
* FROM `OUTPUT` WHERE id=1", ARRAY_N); if($ready3[2]==1){echo "Ligar";}else
{echo "Desligar";}?> </button>
<?php
global $wpdb;
$ready4 = $wpdb->get_row( "SELECT * FROM `OUTPUT` WHERE id=1", ARRAY_N);
if(isset($_POST['change3']))
{
    if($ready4[2]==1)
    {
        $wpdb->update('OUTPUT', array('LAMPADA' =>"0"), array('id'=>1),
array('%s'));
        $wpdb->update('OUTPUT', array('MODO_AUTOMATICO' =>"0"), array('id'=>1),
array('%s'));
    }
    else
    {
        $wpdb->update('OUTPUT', array('LAMPADA' =>"1"), array('id'=>1),
array('%s'));
        $wpdb->update('OUTPUT', array('MODO_AUTOMATICO' =>"0"), array('id'=>1),
array('%s'));
    }
}
}

```

Para o controlo de luminosidade voltar ao modo automático o utilizador terá de usar o segundo botão, este irá ficar em tons de cinza (desativo) até que o utilizador volte a pressionar o botão Ligar/Desligar.

```
</form>
<form action="" method="Post" >
<button type="submit" id="automatic" name="automatic" <?php global $wpdb;
$automatic = $wpdb->get_row( "SELECT * FROM `OUTPUT` WHERE id=1", ARRAY_N);
if($automatic[7]==1){echo "disabled" ;} else{echo 'style=" background-color:
#4CB62C; border: none; color: white; padding: 15px 32px; text-align: center;
text-decoration: none; display: inline-block; font-size: 16px; margin: 4px
2px; cursor: pointer;"';}>> Modo Automático </button>
<?php
global $wpdb;
$ready5 = $wpdb->get_row( "SELECT * FROM `OUTPUT` WHERE id=1", ARRAY_N);
if(isset($_POST['automatic']))
{
echo "<meta http-equiv='refresh' content='0'>";
$wpdb->update('OUTPUT', array('MODO_AUTOMATICO' =>"1"), array('id'=>1),
array('%s'));
}
?>
</form>
```

Apêndice 3.8 Ensaio 1 C/C++ Controlo de luminosidade Switch...Case

```
case lightController:
{
    ACTUALLUX = LDR();
    stateMachineControler = readWaterTemperature;
    break;
}
```

Apêndice 3.9 Ensaio 1 C/C++ Controlo de luminosidade Modo Manual

```
int LDR(void)
{
    int LDRValue;
    LDRValue = analogRead(LDRSensor);
    Serial.print("LDR: ");
    Serial.println(LDRValue);
    time_t t = now();
    unsigned long hr = hour(t);
    int a = 60;

    if (AUTOMATIC == 0 && LIGHT == 0)
    {
        Serial.println("LUMINOSIDADE: MODO MANUAL DESLIGADA");
        digitalWrite(BulbON, LOW);
        digitalWrite(BulbOFF, HIGH);
        delay(500);
    }
    if (AUTOMATIC == 0 && LIGHT == 1)
```

```

{
  Serial.println("LUMINOSIDADE: MODO MANUAL LIGADA");
  digitalWrite(BulbON, HIGH);
  digitalWrite(BulbOFF, LOW);
  delay(500);
}

```

Apêndice 3.10 Ensaio 1 C/C++ Controlo de luminosidade Modo Automático

```

if (AUTOMATIC == 1 && LDRValue > 30)
{
  if (hr >= 7)
  {
    if (hr <= 21)
    {
      Serial.println("LUMINOSIDADE: MODO AUTOMATICO DENTRO DE HORAS E LUMINOSIDADE
BAIXA");
      digitalWrite(BulbON, HIGH);
      digitalWrite(BulbOFF, LOW);
      delay(500);
    }
    else
    {
      Serial.println("LUMINOSIDADE: MODO AUTOMATICO FORA DE HORAS");
      digitalWrite(BulbON, LOW);
      digitalWrite(BulbOFF, HIGH);
      delay(500);
    }
  }
  else
  {
    Serial.println("LUMINOSIDADE: MODO AUTOMATICO FORA DE HORAS");
    digitalWrite(BulbON, LOW);
    digitalWrite(BulbOFF, HIGH);
    delay(500);
  }
}
if (AUTOMATIC == 1 && LDRValue >= 30)
{
  Serial.println("LUMINOSIDADE: MODO AUTOMATICO LUMINOSIDADE OK");
  digitalWrite(BulbON, LOW);
  digitalWrite(BulbOFF, HIGH);
  delay(500);
}
return LDRValue;
}

```

Apêndice 3.11 Ensaio 1 Código web para controlo EC

```

<div style="width:45%;padding:0 10pt 0 0;float:left;text-align:center; ">
<h1> EC </h1>
</br>
<form action="" method="Post" >

```

```

<input type="number" id="EC" min="0" max="3" onkeypress='return event.charCode >= 48 &&
event.charCode <= 57' name="EC" size="10" placeholder="EC" step=".1"/>
<button type="submit" id="change1" name="change1" value="" style="cursor:pointer; background-
color:white; border:none"></button>
<?php
global $wpdb;
$ready1 = $wpdb->get_row( "SELECT * FROM `OUTPUT` WHERE id=1", ARRAY_N);
if(isset($_POST['change1']))
{
    if($ready1==NULL)
    {
        echo "Insira uma planta";
    }
    else
    {
        $EC= $_POST["EC"];
        if($ready1[4]>$EC)
        {
            echo " Insira um valor entre $ready1[4] e 3";
        }
        else
        {
            $wpdb->update('OUTPUT', array('EC' =>$EC), array('id'=>1), array('%s'));
        }
    }
}
?>
</form>
</br>
</div>

```

Apêndice 3.12 Ensaio 1 C/C++ Leitura da temperatura

```

float checkWaterTemperature()
{
    int i;
    float temp;
    float To;
    float sumTemperature = 0;
    float averageTemperature = 0;

    for (i = 0; i < 10; i++) {
        temp = analogRead(ReadWaterTemperatureP0);
        To = 10 * temp * 2.8 / 1024;
        sumTemperature = sumTemperature + To;
        delay(100);
    }
    averageTemperature = sumTemperature / 10;
    return averageTemperature;
}

```


Apêndice 3.13 Ensaio 1 C/C++ Leitura de EC

```
float checkEC()
{
    float TemperatureCoef = 0.019;
    float K = 4.48;
    float WaterTemperature = 0;
    float EC = 0;
    float EC25 = 0;
    float raw = 0;
    float Vin = 4.9;
    float Vdrop = 0;
    float Rc = 0;
    int R1 = 1000;
    int Ra = 25; //Resistance of powering Pi

    WaterTemperature = checkWaterTemperature();
    do{
        raw = EC_read();
    }while(raw<490);
    //raw=((raw1+raw2)/2);
    // Vdrop = (Vin * raw) / 1024.0;
    // Serial.print("Vdrop:");
    // Serial.println(Vdrop);
    // Rc = (Vdrop * R1) / (Vin - Vdrop); //falta ajustar a diferença de valores que está com valores
    // diferentes no que toca a agua sem nutrientes
    // Rc = Rc - Ra; //accounting for Digital Pin Resitance
    //EC = raw * 0.505204 / 19;
    EC = 0.0071 * raw - 4.7624;
    EC25 = EC / (1 + TemperatureCoef * (WaterTemperature - 25.0));
    ACTUALEC = EC25;
    Serial.print("EC:");
    Serial.println(EC25);
    delay (6000);

    return EC25;
}

case loadEC:
{ // activar motores EC por 0.5s
    //caso este seja adicionado mais de 10vezes, existe um problema

    activateECMotors();

    stateMachineControler = readECReservoir;

    break;
}
```

```

int EC_read ()
{
    int i;
    int temp2;
    int temp1;
    int j;
    float raw1 = 0;
    float raw2 = 0;

    pinMode(ECPin1, OUTPUT);
    pinMode(ECPin2, INPUT);
    digitalWrite(ECPower1, HIGH);
    digitalWrite(ECPower2, LOW);
    delay(200);

    j=0;
    temp1=0;

    for (i=0; i<5; i++)
    {
        //temp1 = analogRead(ECPin2);
        temp1 = analogRead(ECPin2);
        //Serial.println(temp1);
        delay(50);
        j++;
        raw1=raw1+temp1;
    }
    digitalWrite(ECPower1, LOW);
    digitalWrite(ECPower2, LOW);
    pinMode(ECPin1, OUTPUT);
    pinMode(ECPin2, OUTPUT);

    raw1=raw1/j;
    Serial.print("raw1:");
    Serial.println(raw1);

    // ----- situação 2

    pinMode(ECPin1, INPUT);
    pinMode(ECPin2, OUTPUT);
    digitalWrite(ECPower1, LOW);
    digitalWrite(ECPower2, HIGH);
    delay(200);

    j=0;
    temp2=0;

    for (i=0; i<5; i++)
    {
        temp2 = analogRead(ECPin1);
        temp2 = analogRead(ECPin1);
        delay(50);
        j++;
        raw2=raw2+temp2;
    }
}

```

```

digitalWrite(ECPower1, LOW);
digitalWrite(ECPower2, LOW);
pinMode(ECPin1, OUTPUT);
pinMode(ECPin2, OUTPUT);

raw2=raw2/j;
Serial.print("raw2:");
Serial.println(raw2);

return raw1;
}

```

Apêndice 3.14 Ensaio 1 C/C++ Acionar motores para adicionar nutrientes

```

void activateECMotors()
{

    if(COUNTEREC < 2500)
    {
        pixels.setPixelColor(1, pixels.Color((COUNTEREC/10),255,0)); // Moderately bright green color.
        pixels.setPixelColor(0, pixels.Color((COUNTEREC/10),255,0));
        pixels.show(); // This sends the updated pixel color to the hardware.
        delay(5); // Delay for a period of time (in milliseconds).
    }
    if(2500 <= COUNTEREC <= 5000)
    {
        pixels.setPixelColor(1, pixels.Color(255,(255-((COUNTEREC-2500)/10)),0)); // Moderately bright green
        color.
        pixels.setPixelColor(0, pixels.Color(255,(255-((COUNTEREC-2500)/10)),0));
        pixels.show(); // This sends the updated pixel color to the hardware.
        delay(5); // Delay for a period of time (in milliseconds).

        if (COUNTEREC == 5000)
        {

            while(digitalRead(Button_PH_EC) == HIGH)
            {
                pixels.setPixelColor(0, pixels.Color(255,0,0)); // Moderately bright green color.
                pixels.setPixelColor(1, pixels.Color(255,0,0));
                pixels.show(); // This sends the updated pixel color to the hardware.
                delay(1000);
                pixels.setPixelColor(0, pixels.Color(0,0,0)); // Moderately bright green color.
                pixels.setPixelColor(1, pixels.Color(0,0,0));
                pixels.show(); // This sends the updated pixel color to the hardware.
                delay(1000);
            }
            COUNTEREC = 0;
        }
    }
    else

```

```

{
  Serial.println("ERRO");
}
COUNTEREC ++;

digitalWrite(ECMotorA, HIGH); //300ms = 0.5ml a 5.0V
delay(320);
digitalWrite(ECMotorA, LOW);
delay(320);
digitalWrite(ECMotorB, HIGH); //300ms = 0.5ml a 5.0V
delay(320);
digitalWrite(ECMotorB, LOW);

}

```

Apêndice 3.15 Ensaio 1 Código web para control de pH

```

<div style="width:45%;padding:0 10pt 0 0;float:left;text-align:center; ">
<h1> pH </h1>
<br>
<form action="" method="Post" >
<input type="number" id="PH" min="5" max="6" onkeypress='return event.charCode >= 48 &&
event.charCode <= 57' name="PH" size="10" placeholder="pH" step=".1"/>

<button type="submit" id="change2" name="change2" value="" style="cursor:pointer; background-
color:white; border:none">

</button>

<?php global $wpdb; $myresult = $wpdb->get_row("SELECT * FROM `INPUT`ORDER BY id DESC LIMIT 1",
ARRAY_N);
if ($myresult)
{echo "$myresult[2]";}
else
{echo "Insira uma planta";}?>
</form>
</br>
</div>

```

Apêndice 3.16 Ensaio 1 C/C++ Controlo de pH

```

case readPHReservoir:
{ //leitura de ph
  // caso baixe suba dos 6.2, vai adicionar PHDOWN para voltar a normalizar a solução
  float PH;
  int timesAddedPHDown = 0;

  PH = checkPH();

  if (PH < 5.6 or PH > 6.5)
  {
    stateMachineControler = loadPH;

```

```

    ACTUALPH=PH;
}
else
{
    ACTUALPH=PH;
    stateMachineControler = loadPipe;
    delay (3000);
}
break;
}
case loadPH:
{ //adicionar produto para baixar ph da solução,

    activatePHMotor();
    stateMachineControler = readPHReservoir;

    break;
}

float checkPH()
{
    int i;
    float sumPH = 0;

    delay (100);
    for (i = 0; i < 10; i++)
    {
        int measure = analogRead(ReadPH);
        float Po= (1023 - measure)/73.07;
        sumPH = sumPH + Po;
        delay (10);
    }
    STARTPH = sumPH / 10;
    Serial.print("PH:");
    Serial.println(STARTPH);
    STARTPH = 6;
    return STARTPH;
}

void activatePHMotor()
{
    if(COUNTERPH < 2500)
    {
        pixels.setPixelColor(2, pixels.Color((COUNTERPH/10),255,0)); // Moderately bright green color.
        pixels.show(); // This sends the updated pixel color to the hardware.
        delay(5); // Delay for a period of time (in milliseconds).
    }
    if(2500 <= COUNTERPH <= 5000)
    {
        pixels.setPixelColor(2, pixels.Color(255,(255-((COUNTERPH-2500)/10)),0)); // Moderately bright green
        color.
        pixels.show(); // This sends the updated pixel color to the hardware.
        delay(5); // Delay for a period of time (in milliseconds).
    }
}

```

```

if (COUNTERPH == 5000)
{

    while(digitalRead(Button_PH_EC) == HIGH)
    {
        pixels.setPixelColor(2, pixels.Color(255,0,0)); // Moderately bright green color.
        pixels.show(); // This sends the updated pixel color to the hardware.
        delay(1000);
        pixels.setPixelColor(2, pixels.Color(0,0,0)); // Moderately bright green color.
        pixels.show(); // This sends the updated pixel color to the hardware.
        delay(1000);
    }
    COUNTERPH = 0;
}
}
else
{
    Serial.println("ERRO");
}
COUNTERPH ++;

digitalWrite(PHMotor, HIGH); //300ms = 0.5ml a 5.0V
delay(320);
digitalWrite(PHMotor, LOW);
delay(320);

}

```

Apêndice 3.17 Ensaio 1 C/C++ Bombear solução

```

case loadPipe:
    { //caso EC E PH estejam nos parametros, vamos encher a calha
        fillPipe();
        stateMachineControler = writeSerial;
        break;
    }

```

```

void fillPipe(void)
{
    for (int i = 0 ; i < 10 ; i++)
    {
        digitalWrite(FillMotor, HIGH);
        delay(5000);
        digitalWrite(FillMotor, LOW);
        delay(10000);
    }
}

```

Apêndice 3.18 Ensaio 1 C/C++ Envia para BD

```

case writeSerial:
{
    float lux = conversionLDR_LUX();

```

```

char t;
do
{ t = Serial.read();
} while (Serial.available() > 0);

digitalWrite(SerialSend, 1);
delay(200);

Serial.println('a');
//delay(500);
Serial.println(ACTUALEC,2);
///delay(500);
Serial.println(ACTUALPH,2);
///delay(500);
Serial.println(lux,1);
//delay(500);
Serial.println(WATERTEMPERATURE,2);
digitalWrite(SerialSend, 0);
stateMachineControler = sleepMode;
break;
}

```

Apêndice 3.19 Ensaio 1 Python Escreve na BD

```

while k is 1:
    dbConn = pymysql.connect(host="localhost",user="root",passwd="girafa",db="wordpress")
    cursor = dbConn.cursor()
    cursor.execute("SELECT * FROM OUTPUT ORDER BY id DESC LIMIT 1 ")
    row=cursor.fetchone()
    var1 = GPIO.input(20)
    if var1 is 0:
        read = arduino.readline()
        if read is not b"":
            print(read)
            if read is 'a':
                print("leu 'a' adasdadasdasdasd")
                r = 1;

    cursor.close()
    while row is None and x is 0:
        dbConn = pymysql.connect(host="localhost",user="root",passwd="girafa",db="wordpress")
        cursor = dbConn.cursor()
        cursor.execute("SELECT * FROM OUTPUT ORDER BY id DESC LIMIT 1 ")
        row=cursor.fetchone()
        var1 = GPIO.input(20)

    if row is not None and x is 0 :
        row_str = str(row)
        row_str_aux = row_str.replace("","")
        row_str_final = row_str_aux.replace(")","")
        data_splited = row_str_final.split(",")
        data_aux = (data_splited[1] + data_splited[9] + data_splited[11] + data_splited [12] + data_splited
[2])

        data_final =data_aux.ljust(30, ' ')
        EC = data_splited [9]

```

```

AUTOMATIC = data_splited [12]
PH = data_splited [10]
LIGHT = data_splited [2]
time.sleep(0.1)
arduino.write(time.strftime("%H%M%S%d%m%Y").encode())
print(time.strftime("%H%M%S%d%m%Y"))
time.sleep(0.1)
print(data_final.encode())
arduino.write(data_final.encode())
time.sleep(0.1)
x= 1

if x is 1:
    dbConn = pymysql.connect(host="localhost",user="root",passwd="girafa",db="wordpress")
    cursor = dbConn.cursor()
    cursor.execute("SELECT * FROM OUTPUT ORDER BY id DESC LIMIT 1 ")
    row_str = str(row)
    row_str_aux = row_str.replace("","")
    row_str_final = row_str_aux.replace(")","")
    data_splited = row_str_final.split(",")
    now = datetime.datetime.now()
    cursor.close()
    if now.minute is 15 or now.minute is 45:
        aux = 1
    else:
        aux = 0
    if data_splited[9] != EC or data_splited[12] != AUTOMATIC or data_splited[10] != PH or
data_splited[2] != LIGHT or aux is 1:
        if data_splited[9] == ' R':
            print("RESET")
            dbConn = pymysql.connect(host="localhost",user="root",passwd="girafa",db="wordpress")
            cursor = dbConn.cursor()
            cursor.execute("set FOREIGN_KEY_CHECKS = 0;")
            cursor.execute("TRUNCATE TABLE INPUT;")
            dbConn.commit()
            time.sleep(0.1)
            cursor.execute('TRUNCATE TABLE OUTPUT ')
            time.sleep(0.1)
            dbConn.commit()
            cursor.close()
            print("RESET FEITO")
            a = 0
            break
        else:
            GPIO.output(21, GPIO.HIGH)
            EC = data_splited[9]
            AUTOMATIC = data_splited[12]
            PH = data_splited[10]
            LIGHT = data_splited [2]
            data_aux = ( "a" + data_splited[9] + data_splited[10] + data_splited[12] + data_splited[2] )
            data_final = data_aux.ljust(40, ' ')
            GPIO.output(21, GPIO.LOW)
            print(data_final.encode(errors="repalce"))
            arduino.write(data_final.encode())
            x=1

if var1 or r:

```



```

print("-----Escreve na base de dados-----")
)
reading = []
serial = arduino.readline().decode('ascii').strip()
while serial != 'a':
    serial = arduino.readline().decode('ascii').strip()
    print("preso")
    pass
reading.append('a')
print("livre")
for _ in range(4):
    reading.append(arduino.readline().strip())
reading_id +=1
print(reading)
begin = reading[0]
ec = reading[1]
ph = reading[2]
lux = reading[3]
temp = reading[4]
print("ec",ec)
print("ph",ph)
if begin == 'a':
    print("inserido na tabela")
    dbConn = pymysql.connect(host="localhost",user="root",passwd="girafa",db="wordpress")
    cursor = dbConn.cursor()
    cursor.execute("SELECT * FROM INPUT")
    cursor.execute("""INSERT INTO INPUT (EC,PH,LUMINOSIDADE,TEMPERATURA) VALUES
(%s,%s,%s,%s)"""%(ec,ph,lux,temp))
    dbConn.commit() #commit the insert
    print('-----Escrita completa-----')
    GPIO.setup(20,GPIO.IN, GPIO.PUD_DOWN)
    arduino.close()

arduino.close()

```

Apêndice 3.20 Ensaio 1 C/C++ Sleep Mode

```

void sleepNow()
{
    attachInterrupt(INT1,WakeUpByLDR,CHANGE);
    attachInterrupt(13,WakeUpByUser,HIGH);

case sleepMode:
{
    delay(300);
    sleepNow();
    if (INTSTATUS == 1)
    {
        stateMachineControler = readSerial;
        Serial.println("INTERRUPÇÃO PELO UTILIZADOR");
        break;
    }
    if (INTSTATUS == 2)
    {
        stateMachineControler = openWater;
        Serial.println("INTERRUPÇÃO PELO LDR");
        break;
    }
}

```

```

    }

}

void WakeUpByUser()
{
    INTSTATUS = 1;
    Serial.println("Interrupt By User");
}

void WakeUpByLDR()
{
    INTSTATUS = 2;
    Serial.println("Interrupt By LDR");
}

```

Apêndice 3.21 Ensaio 1 Python ativar pino de alteração

else:

```

GPIO.output(21, GPIO.HIGH)
EC = data_splited[9]
AUTOMATIC = data_splited[12]
PH = data_splited[10]
LIGHT = data_splited [2]
data_aux = ( "a" + data_splited[9] + data_splited[10] + data_splited[12] + data_splited[2] )
data_final = data_aux.ljust(40, ' ')
GPIO.output(21, GPIO.LOW)
print(data_final.encode(errors="repalce"))
arduino.write(data_final.encode())
x=1

```

Apêndice 3.22 Ensaio 1 Python ler BD

else:

```

GPIO.output(21, GPIO.HIGH)
EC = data_splited[9]
AUTOMATIC = data_splited[12]
PH = data_splited[10]
LIGHT = data_splited [2]
data_aux = ( "a" + data_splited[9] + data_splited[10] + data_splited[12] + data_splited[2] )
data_final = data_aux.ljust(40, ' ')
GPIO.output(21, GPIO.LOW)
print(data_final.encode(errors="repalce"))
arduino.write(data_final.encode())
x=1

```

Apêndice 3.23 Ensaio 1 C/C++ Ler porta série

```

int read_Serial ()
{
    int seed;
    char all_data[30];
    char c;
    char ec [10] = "";
    char ph [10] = "";
    char aut [10] = "";
    char l [10] = "";

```

```

int j = 0;
int k = 0;

while (c!='a')
{
    c = Serial.read();
}
delay(30);
for(int i = 0 ; i<30 ; i++)
{
    c = Serial.read();
    all_data[i]= c ;
    if ( c == ' ' )
    {
        j++;
        k = 0;
    }
    if (j == 1)
    {
        ec[k] = c;
        k++;
    }
    if (j == 2)
    {
        ph[k] = c;
        k++;
    }
    if (j == 3)
    {
        aut[k] = c;
        k++;
    }
    if (j == 4)
    {
        l[k] = c;
        k++;
    }
}
Serial.println(all_data);
STARTEC = (float)atof(ec);
Serial.print("novo EC: ");
Serial.println(ec);
delay(1000);
STARTPH = (float)atof(ph);
Serial.print("novo PH: ");
Serial.println(ph);
delay(1000);
AUTOMATIC = (float)atof(aut);
Serial.print("novo aut: ");
Serial.println(aut);
delay(1000);
LIGHT = (int)atof(l);
Serial.print("luz: ");
Serial.println(l);

if (all_data[1]== 'R')
{

```

```

Serial.println("RESET MACHINE");
delay(1000);
return 1;
}
else
{
Serial.println("continua");
delay(1000);
return 2;
}
}

```

Apêndice 3.24 Ensaio 2 Python Máquina de estados

```

def StateMachine():
    global STATE
    if STATE is 1:
        STATE = WaitSeed()

    if STATE is 2:
        STATE = FillRes()

    if STATE is 3:
        STATE = CheckLDR()

    if STATE is 4:
        STATE = ControlEC()

    if STATE is 5:
        STATE = AddEC()

    if STATE is 6:
        STATE = ControlpH()

    if STATE is 7:
        STATE = AddpH()

    if STATE is 8:
        STATE = BombSolution()

    if STATE is 9:
        STATE = InsertBD()

    if STATE is 10:
        STATE = WaitChange()

```

Apêndice 3.25 Ensaio 2 C/C++ Máquina de estados

```

if (doc["CMD"] == 1) {
    Serial.println(payload.length());
    delay(50);
    fillRes();
}
if (doc["CMD"] == 2) {
    Serial.println(payload.length());
    delay(50);
    day = doc["day"];
}

```

```

light = doc["light"];
LDR(day, light);
}
if (doc["CMD"] == 3) {
    Serial.println(payload.length());
    delay(50);
    EC = doc["EC"];
    ControlEC(EC);
}
if (doc["CMD"] == 4) {
    Serial.println(payload.length());
    delay(50);
    activateECMotors();
}
if (doc["CMD"] == 5) {
    Serial.println(payload.length());
    delay(50);
    PH = doc["pH"];
    ControlpH(PH);
}
if (doc["CMD"] == 6) {
    Serial.println(payload.length());
    delay(50);
    activatepHMotors();
}
if (doc["CMD"] == 7) {
    Serial.println(payload.length());
    delay(50);
    activateBomb();
}
if (doc["CMD"] == 8) {
    Serial.println(payload.length());
    delay(500);
    checkLDRWAIT();
}
delay(20);
}

```

Apêndice 3.26 Ensaio 2 Python Semente inserida

```

def WaitSeed():
    global EC
    global PH
    global AUTOMATIC
    global LIGHT
    dbConn = pymysql.connect(host="localhost",user="root",passwd="girafa",db="wordpress")
    cursor = dbConn.cursor()
    cursor.execute("SELECT * FROM OUTPUT ORDER BY id DESC LIMIT 1 ")
    row=cursor.fetchone()
    if row is not None:
        row_str = str(row)
        row_str_aux = row_str.replace("'", "")
        row_str_final = row_str_aux.replace(")", "")
        data_splited = row_str_final.split(",")
        data_aux = (data_splited[1] + data_splited[9] + data_splited[11] + data_splited [12] + data_splited
[2])

```

```

data_final = data_aux.ljust(30, ' ')
EC = data_splited [9]
AUTOMATIC = data_splited [12]
PH = data_splited [10]
LIGHT = data_splited [2]
print("-----DADOS DO UTILIZADOR-----")
print("Planta:", data_splited [1])
print("EC:", EC)
print("Dias de Crescimento:", data_splited [11])
print("-----DADOS MEDIDOS-----")
return 2
else:
    return 1

```

Apêndice 3.27 Ensaio 2 Python Reservatorio cheio

```

def FillRes():
    Rsize = 1
    FillResComplete = 1
    data = {"CMD": 1}
    data=json.dumps(data)
    size = len(data)
    while Rsize != size:
        arduino.write(data.encode('ascii'))
        arduino.flush()
        try:
            incoming = arduino.readline().decode("utf-8").strip("\r").strip("\n").strip("")
            Rsize = int(incoming)
        except Exception as e:
            print ("erro 1")
            print (e)
            pass
        time.sleep(1)

```

Apêndice 3.28 Ensaio 2 C/C++ Enche Reservatorio

```

if (doc["CMD"] == 1) {
    Serial.println(payload.length());
    delay(50);
    fillRes();
}

```

```

void fillRes()
{
    StaticJsonDocument<512> doc;
    int res=0;

    pinMode(SensorRes1,INPUT);

```

```

pinMode(SensorRes2,OUTPUT);
digitalWrite(SensorRes2, HIGH);
delay(10);
res = digitalRead(SensorRes1);
digitalWrite(SensorRes2, LOW);
pinMode(SensorRes1, INPUT);
pinMode(SensorRes2, INPUT);

if (res)
{
    digitalWrite(ValveInReservoir, HIGH);
    digitalWrite(MixSolution, HIGH);
    doc ["Ready"] = 1;
    serializeJson(doc, Serial);
}
else
{
    digitalWrite(ValveInReservoir, LOW);

```

Apêndice 3.29 Ensaio 2 Python Ready = 0

```

while FillResComplete:
    try:
        incoming = arduino.readline()
        incoming_dict = json.loads(incoming)
        if incoming_dict["Ready"] == 1:
            print ("ENCHER RESERVATORIO: OK")
            time.sleep(1)
            FillResComplete = 0
            return 3
        else:
            FillResComplete = 1
            print ("ENCHER RESERVATORIO: ...")
            time.sleep(1)
            return 2

    except Exception as e:
        print ("erro aqui")
        print (e)
        pass
        return 2

```

Apêndice 3.30 Ensaio 2 Python Controlo Luminosidade

```

def CheckLDR():
    global Read_LDR
    Rsize = 1
    LDRComplete = 1
    Aut = int(AUTOMATIC)
    lig = int(LIGHT)
    now = datetime.datetime.now()

```

```

if Aut:
    if now.hour >= 7 and now.hour <= 20:
        data = { "CMD": 2, "day": 1}
        print("Luminosidade em Modo Automatico e dentro de horas")
    else:
        data = { "CMD": 2, "day": 0}
        print("Luminosidade em Modo Automatico e fora de horas")
else:
    if lig:
        data = { "CMD": 2, "light": 1}
        print("Luminosidade em Modo Manual: Liga Lampada")
    else:
        data = { "CMD": 2, "light": 0}
        print("Luminosidade em Modo Manual: Desliga Lampada")
data=json.dumps(data)
size = len(data)
while Rsize != size:
    arduino.write(data.encode('ascii'))
    arduino.flush()
    try:
        incoming = arduino.readline().decode("utf-8").strip("\r").strip("\n").strip("")
        Rsize = int(incoming)
    except Exception as e:
        print (e)
        pass
    time.sleep(1)

```

Apêndice 3.31 Ensaio 2 C/C++ Controlo Luminosidade

```

void LDR(int day, int light)
{
    StaticJsonDocument<512> doc;
    int LDRValue;
    int R;

    LDRValue = analogRead(LDRSensor);
    if (day)
    {
        if (LDRValue < 30)
        {
            doc ["light"] = 0;
            digitalWrite(Bulb, LOW);
        }
        else
        {
            doc ["light"] = 1;
            digitalWrite(Bulb, HIGH);
        }
    }
    else
    {
        if (light)
        {
            doc ["light"] = 1;
            digitalWrite(Bulb, HIGH);
        }
        else
    }
}

```



```

{
    doc ["light"] = 0;
    digitalWrite(Bulb, LOW);
}

}
for (R = 0; R < 50; R++)
{
    if (analogRead(LDRSensor) == 0)
        doc ["LDR"][R] = 1;
    else
        doc ["LDR"][R] = analogRead(LDRSensor);
}
doc ["Ready"] = 1;
serializeJson(doc, Serial);
}

```

Ensaio 2 Python Registo da luminosidade

```

while LDRComplete:
    try:
        incoming = arduino.readline().decode()
        incoming_dict = json.loads(incoming)
        if incoming_dict["Ready"] == 1:
            time.sleep(1)
            LDRComplete = 0
            if incoming_dict["light"] == 1:
                print("lampada ligada")
            else:
                print("lampada Desligada")
            temp = statistics.median(incoming_dict["LDR"])
            #temp = 1023 - temp
            Read_LDR = round((250/((temp)*0.0048828125))-50,2)
            #Read_LDR = int((1.25 * pow(10,7))*pow(temp,(-1.4059)))
            print("LDR", Read_LDR)
            print("Control Luminosidade: OK")
            return 4
        else:
            LDRComplete = 1
            print("Control Luminosidade: ...")
            time.sleep(1)
            return 3

    except Exception as e:
        print(e)
        pass
        return 2

```

Apêndice 3.32 Ensaio 2 Python Controlo EC

```

def ControlEC():
    global EC
    global Read_EC

```

```

global Read_Temp
global EC_Confirm
Rsize = 1
ECComplete = 1
MultiEC = float(EC)*100
MultiEC = int(MultiEC)
data = { "CMD": 3, "EC": MultiEC}
data=json.dumps(data)
size = len(data)
while Rsize != size:
    arduino.write(data.encode('ascii'))
    arduino.flush()
    try:
        incoming = arduino.readline().decode("utf-8").strip("\r").strip("\n").strip("")
        Rsize = int(incoming)
    except Exception as e:
        print (e)
        pass
    time.sleep(1)
while ECComplete:
    try:
        time.sleep(1)
        incoming = arduino.readline()
        incoming_dict = json.loads(incoming)
        if incoming_dict["Ready"] == 1:
            time.sleep(1)
            ECComplete = 0
            REC = statistics.median(incoming_dict["ReadEC"])
            temp = statistics.median(incoming_dict["Water"])
            print(temp)
            Read_Temp = round((10 * temp * 2.8 / 1024),2)
            Read_EC = round((REC / 500),2)
            if Read_EC < float(EC):
                print ("Control EC: Falta EC")
                print ("Valor EC:", Read_EC)
                EC_Confirm = 0
                return 5
            else:
                print ("Valor EC:", Read_EC)
                if EC_Confirm < 5:
                    EC_Confirm += 1
                    print ("Confirmar Valor de EC: ", EC_Confirm, " / 5")
                    time.sleep (5)
                    return 4
                else:
                    EC_Confirm = 0
                    print ("Confirmar Valor de EC: OK")
                    return 6
        else:
            ECComplete = 1
            print ("READY 0")
            time.sleep(1)
            return 3
    except Exception as e:
        print (e)
        pass
        return 2

```

Ensaio 2 C/C++ Controlo EC

```
void ControlEC(int EC)
{

    StaticJsonDocument<512> doc;
    int ReadEC;
    int Water;
    int i;
    int j=0;
    int temp = 0;
    float To;
    float sumTemperature = 0;
    float averageTemperature;

    digitalWrite(MixSolution, LOW);
    delay(2000);

    for (i = 0; i < 30; i++) {
        temp = analogRead(ReadWaterTemperatureP0);
        To = 10 * temp * 2.8 / 1024;
        sumTemperature = sumTemperature + To;
        j++;
        doc ["Water"][i] = To;
    }
    averageTemperature = sumTemperature / j;
    //sumTemperature = random(190,210)/10;
    averageTemperature = sumTemperature / j;
    gravityTds.setTemperature(averageTemperature);
    gravityTds.update();
    doc ["Ready"] = 1;
    for (i=0;i<30;i++)
    {
        doc ["ReadEC"][i] = gravityTds.getTdsValue();
    }
    //
    // for (i=0;i<30;i++)
    //   doc ["Water"][i] = analogRead(ReadWaterTemperatureP0);
    //   //doc ["Water"][i] = random(20,21);
    //
    digitalWrite(MixSolution, HIGH);
    serializeJson(doc, Serial);
}
```

Apêndice 3.33 Ensaio 2 C/C++ Adicionar nutrientes

```
void activateECMotors()
{

    StaticJsonDocument<512> doc;

    digitalWrite(ECMotorA, HIGH); //2000ms = 2ml a 5.0V
```

```

delay(2000);
digitalWrite(ECMotorA, LOW);
delay(2000);
digitalWrite(ECMotorB, HIGH); //2000ms = 2ml a 5.0V
delay(2500);
digitalWrite(ECMotorB, LOW);
delay(2000);

doc ["Ready"]= 1;
serializeJson(doc, Serial);

}

```

Apêndice 3.34 Ensaio 2 Python Bombear solução

```

def BombSolution():
    BSComplete = 1
    Rsize = 1
    data = { "CMD": 7}
    data=json.dumps(data)
    size = len(data)
    while Rsize != size:
        arduino.write(data.encode('ascii'))
        arduino.flush()
        try:
            incoming = arduino.readline().decode("utf-
8").strip("\r").strip("\n").strip("")
            Rsize = int(incoming)
        except Exception as e:
            print (e)
            pass
        time.sleep(1)
    while BSComplete:
        try:
            print ("A Repor Solucao nas plantas ...")
            nextPercent, nextPoint = 0, 0
            total = 150
            for num in range (total):
                nextPercent, nextPoint = ProgressBar (num, total, nextPercent,
nextPoint)
            incoming = arduino.readline()
            incoming_dict = json.loads(incoming)
            if incoming_dict["Ready"] == 1:
                BSComplete = 0
                print ("Repor Solucao nas plantas: OK")
                return 9
            else:
                BSComplete = 1
                print ("Repor Solucao nas plantas: ...")
                time.sleep(1)
                return 8
        except Exception as e:
            print (e)
            pass
            return 2

```

Apêndice 3.35 Ensaio 2 C/C++ Bombear Solução

```

void activateBomb(void)
{

```

```

for (int i = 0 ; i < 10 ; i++)
{
digitalWrite(FillMotor, HIGH); //300ms = 0.5ml a 5.0V
delay(5000);
digitalWrite(FillMotor, LOW);
delay(10000);
}

JsonObjectDocument<512> doc;
doc ["Ready"]= 1;
serializeJson(doc, Serial);
}

```

Apêndice 3.36 Ensaio 2 Python Escrever BD

```

def InsertBD():
    try:
        dbConn = pymysql.connect(host="localhost",user="root",passwd="girafa",db="wordpress")
        cursor = dbConn.cursor()
        cursor.execute("SELECT * FROM INPUT")
        cursor.execute("""INSERT INTO INPUT (EC,PH,LUMINOSIDADE,TEMPERATURA) VALUES
(%s,%s,%s,%s)""",(Read_EC,Read_PH,Read_LDR,Read_Temp))
        dbConn.commit() #commit the insert
        print("Dados inseridos na BD: OK")
        return 10
    except Exception as e:
        print(e)
        print("Dados inseridos na BD: ERRO")
        print("Standby até nova alteração (BD, Luminosidade, Tempo)")
        pass
        return 9

```

Apêndice 3.37 Ensaio 2 Python Tempo espera

```

def WaitChange():

#=====CHECK LDR=====
Rsize = 1
LDRComplete = 1
data = { "CMD": 8}
data=json.dumps(data)
size = len(data)
while Rsize != size:
    arduino.write(data.encode('ascii'))
    arduino.flush()
    try:
        incoming = arduino.readline().decode("utf-8").strip("\r").strip("\n").strip("")
        Rsize = int(incoming)
    except Exception as e:
        print(e)
        pass
    time.sleep(1)
while LDRComplete:
    try:
        incoming = arduino.readline().decode()

```

```

incoming_dict = json.loads(incoming)
if incoming_dict["Ready"] == 1:
    time.sleep(1)
    LDRComplete = 0
#     temp = statistics.median(incoming_dict["LDR"])
#     temp = 1023 - temp
#     NowLDR = int((1.25 * pow (10,7))*pow(temp,(-1.4059)))
temp = statistics.median(incoming_dict["LDR"])
#temp = 1023 - temp
NowLDR = round((250/((temp)*0.0048828125))-50,2)
#Read_LDR = int((1.25 * pow (10,7))*pow(temp,(-1.4059)))
dbConn = pymysql.connect(host="localhost",user="root",passwd="girafa",db="wordpress")
cursor = dbConn.cursor()
cursor.execute("SELECT * FROM INPUT ORDER BY id DESC LIMIT 1 ")
row=cursor.fetchone()
row_str = str(row)
row_str_aux = row_str.replace("","")
row_str_final = row_str_aux.replace("","")
data_splited = row_str_final.split(",")
LastLDR = float(data_splited [3])
if LastLDR > 200 and NowLDR > 200:
    pass
elif LastLDR < 200 and NowLDR < 200:
    pass
elif LastLDR > 200 and NowLDR < 200:
    print("-----LDR COM ALTERAÇÃO-----")
    print("BD LDR",LastLDR)
    print("NOW LDR",NowLDR)
    return 1
elif LastLDR < 200 and NowLDR > 200:
    print("-----LDR COM ALTERAÇÃO-----")
    print("BD LDR",LastLDR)
    print("NOW LDR",NowLDR)
    return 1
else:
    LDRComplete = 1
    time.sleep(1)
    return 10

except Exception as e:
    print (e)
    pass
    return 10

```

Apêndice 3.38 Ensaio 2 C/C++ Espera LDR

```

void checkLDRWAIT()
{
    StaticJsonDocument<512> doc;
    int LDRValue;
    int R;

    // ADCSRA = 0;
    // delay(10);
    // ADCSRA = 1;
    // delay(10);
    for (R = 0; R < 30; R++)

```

```

{
{
if (analogRead(LDRSensor) == 0)
  doc ["LDR"][R] = 1;
else
  doc ["LDR"][R] = analogRead(LDRSensor);
}
delay(10);
}
// R = (5000-((1023-LDRValue)/204.6) * 10000)/(1023-LDRValue);
// R = 30000;
doc ["Ready"] = 1;
serializeJson(doc, Serial);
}

```

Apêndice 3.39 Ensaio 2 Python Espera utilizador

```

dbConn = pymysql.connect(host="localhost",user="root",passwd="girafa",db="wordpress")
cursor = dbConn.cursor()
cursor.execute("SELECT * FROM OUTPUT ORDER BY id DESC LIMIT 1 ")
row=cursor.fetchone()
if row is not None:
  row_str = str(row)
  row_str_aux = row_str.replace("'", "")
  row_str_final = row_str_aux.replace(")", "")
  data_splited = row_str_final.split(",")
  data_aux = (data_splited[1] + data_splited[9] + data_splited[11] + data_splited [12] + data_splited
[2])
  data_final =data_aux.ljust(30, ' ')
  if data_splited[9] != EC or data_splited[12] != AUTOMATIC or data_splited[10] != PH or
data_splited[2] != LIGHT:
    if data_splited[9] == ' R':
      dbConn = pymysql.connect(host="localhost",user="root",passwd="girafa",db="wordpress")
      cursor = dbConn.cursor()
      cursor.execute("set FOREIGN_KEY_CHECKS = 0;")
      cursor.execute("TRUNCATE TABLE INPUT;")
      dbConn.commit()
      time.sleep(0.1)
      cursor.execute("set FOREIGN_KEY_CHECKS = 0;")
      cursor.execute("TRUNCATE TABLE LEVEL;")
      dbConn.commit()
      time.sleep(0.1)
      cursor.execute("TRUNCATE TABLE OUTPUT ")
      time.sleep(0.1)
      dbConn.commit()
      cursor.close()
      print("RESET FEITO")
      a = 0
      return 1
    else:
      print("----- BD COM ALTERAÇÃO-----")
      return 1
    else:
      pass
  else:

```

```
return 1
```

Apêndice 3.40 Ensaio 2 Python Espera Tempo

```
#=====TIME=====
now = datetime.datetime.now()
if now.minute is 15 or now.minute is 45:
    print("Rotina recomeça", now.minute)
    return 1
else:
    return 10
```