UNIVERSIDADE SÃO JUDAS TADEU

Bruno Vaz de Souza

Leonardo Gonçalves Pereira

Leonardo Morijo dos Santos

Matheus Dantas Cipolotti

SISTEMA AUTOMÁTICO DE HORTA HIDROPÔNICA

Engenharia Eletrônica

Ismael Mendonça Rezende

São Paulo

2022

RESUMO

A partir de análises sobre o funcionamento de hortas hidropônicas, foram encontrados diversos pontos de falhas que podem ocasionar escalonáveis problemas, visando aprimorar e aperfeiçoar a hidroponia, foi desenvolvido um sistema que tem como objetivo diminuir os riscos de falhas no desenvolvimento da horta, que podem chegar a estados extremos. Um dos pontos mais críticos percebidos a partir de estudos de caso foi a falta de energia, seja por problemas de rede, temporais, etc. Outro ponto crítico encontrado foi a dificuldade de manuseio do sistema, pois o mesmo necessita de atenção e monitoramento diário, para que continue a se desenvolver, o que até então é realizado manualmente.

Para a solução da possível falta de energia, foi desenvolvido uma fonte DC capaz de alimentar um circuito que carrega um banco de baterias, sendo este uma aplicação disponibilizada pela UFPR, que são conjugados sistemas de corrente contínua que facilitam o manuseio e eliminam a necessidade de inversores de frequência, o circuito detecta a queda de energia e através de acopladores ópticos, envia um sinal ao microcontrolador que realiza o chaveamento com o banco de baterias como circuito principal.

O monitoramento do nível de água e nutrientes é realizado dentro dos tanques de armazenamento, utilizando boias mecânicas que internamente comutam o sinal elétrico, esses sinais são analisados pelo microcontrolador que acionarão válvulas para permitir a entrada de mais água ou mais nutriente concentrado no tanque de irrigação principal.

Os dados do tipo de plantio e consequentemente a quantidade de nutrientes da mesma, são armazenados na programação do microcontrolador, que será responsável pelo envio dos dados da produção para uma Interface Homem Máquina e ao mesmo tempo ele receberá dados para configurar a plantação da maneira mais eficiente.

Palavras-chave: Hidroponia, IHM, energia

ABSTRACT

From analyzes on the functioning of hydroponic gardens, several points of failure were found that can cause scalable problems, aiming to improve and perfect hydroponics, a system was developed that aims to reduce the risks of failures in the development of the garden, which can reach extreme states. One of the most critical points perceived from case studies was the lack of energy, whether due to network problems, storms, etc. Other critical point was the difficulty of handling the system, as it needs attention and daily monitoring, so that it continues to develop, which until now has been done manually.

To solve the possible lack of energy, a DC source capable of feeding a circuit that charges a bank of batteries was developed, this being an application made available by UFPR, which are combined direct current systems that facilitate the handling and eliminate the need for frequency inverters, the circuit detects the power outage and through optical couplers, sending a signal to the microcontroller that performs switching with the battery bank as the main circuit.

The monitoring of the level of water and nutrients is carried out inside the storage tanks, using mechanical floats that internally switch the electrical signal, these signals will be analyzed by the microcontroller that will activate valves to allow the entry of more water or more concentrated nutrient in the storage tank. main irrigation.

Data on the type of planting and consequently the amount of nutrients in it will be stored in the microcontroller programming, which will be responsible for sending production data to the Human Machine Interface and at the same time it will receive data to configure the plantation in the most efficient way.

Keywords: Hydroponics, HMI, energy.

Sumário

1. Introdução e justificativa 5

1.1. Objetivos 6

2. Revisão bibliográfica 7

2.1. Hidroponia 7

2.2. Nutrição 9

2.3. Bomba hidráulica 10

2.4. Baterias 11

2.4.1. Capacidade Nominal 11

2.4.2. Capacidade Reserva 12

2.4.3. Estado de carga 12

2.4.4. Estado de Saúde 12

2.4.5. Automação 13

2.5. Microcontrolador 15

3. Metodologia 16

4. Resultados e discussão 18

4.1. Circuito Elétrico 18

4.2. Confecção da PCI (Placa de Circuito Impresso) 22

4.3. Programação do microcontrolador 24

4.4. Programação da IHM 27

5. Conclusões ou Considerações Finais 28

6. Referências 29

7. Anexos 30

7.1. Programação (Microcontrolador) 30

7.2. Programação (Aplicativo) 39

7.3. Esquema elétrico 55

1. Introdução e justificativa

O cultivo hidropônico cresce cada vez mais no Brasil e oferece vantagens tanto para o produtor quanto para o consumidor (Epamig, 2020). Um sistema hidropônico se refere a um método de cultivo onde as plantas são criadas em um ambiente diferente do solo, crescendo a partir da absorção de água e nutrientes, fornecidos por uma fonte externa, sendo um sistema de cultivo raramente visto ao ar livre (EMBRAPA, 2021).

A partir da análise de outros projetos de hortas hidropônicas e os desafios enfrentados, assim como resultados obtidos por estes projetos, atingiu-se a conclusão de que vários produtores que utilizam sistemas hidropônicos sofrem com problemas de falta de energia, devido a “blackouts” ou quedas da rede elétrica ocasionadas por temporais, defeitos da fiação, etc. Além de se tratar de um procedimento de agricultura manual, com baixos índices de monitoramento do sistema. Defeitos que podem ocasionar o atraso no desenvolvimento do plantio ou muitas vezes o fim do mesmo, por se tratar de um processo de cultivo de difícil controle manual.

Através do exposto acima, a proposta deste trabalho é o de desenvolver um sistema de irrigação que funcione em caso de falta de energia com base em uma energia reserva (nobreak) composto por banco de baterias. Além do desenvolvimento de um aplicativo para a visualização dos níveis de energia das baterias, assim como monitoramento do nível de água e vazão dos nutrientes a serem enviados para o plantio, que será definido a partir de um banco de dados com os possíveis plantios que o sistema suporta, calculando assim a quantidade de nutrientes necessários para o crescimento da horta selecionada.

O sistema desenvolvido foi planejado para utilizar a água armazenada em um reservatório próprio, que estará conectado a dois subtanques, onde serão depositados os nutrientes necessários para o plantio selecionado pelo consumidor, estes sendo depositados na forma de polvilho na água, para mistura. Para a conexão dos subtanques ao reservatório, haverá um sistema de controle de nível e controle de vazão, para garantir uma mistura adequada entre a água e o nutriente, para que o desenvolvimento da horta ocorra como planejado.

1.1. Objetivos

Conhecer os principais problemas enfrentados em um plantio que utilize hidroponia e encontrar soluções a partir da eletrônica, que aumentem a efetividade do processo, o deixando mais automatizado de modo que o sistema consiga se sustentar por um determinado período de tempo sem a necessidade de interferências externas para que a nutrição do plantio seja realizada adequadamente.

Objetivos específicos:

- Analisar as necessidades do sistema;
- Identificar as possíveis soluções para as dificuldades enfrentadas no plantio;
- Desenvolvimento de uma interface de monitoramento;
- Estruturar um sistema de autonomia própria, que nutra o plantio sozinho até a necessidade de reabastecimento dos tanques.

2. Revisão bibliográfica

2.1. Hidroponia

A hidroponia consiste no sistema de cultivo sem solo, onde as raízes se mantêm em contato com a água ou ar extremamente úmido, contendo uma solução nutritiva. Como observado na figura 1, onde as plantações se encontram acima do solo, em contato com o cano por onde percorre a água.

Figura 1 – Horta Hidropônica. Fonte: Belagro, 2019.

O cultivo hidropônico requer o conhecimento das exigências das culturas quanto a nutrição, fatores climáticos e fitossanitários, além de disponibilidade de recursos financeiros para a construção da infraestrutura e para a aquisição de equipamentos e insumos. Não é necessária, porém, a realização de práticas culturais, como rotação de cultura, correção do solo, controle de plantas-daninhas, desinfecção e preparo do solo. Conhecidas as necessidades nutricionais e as limitações dos fatores climáticos, é possível cultivar, por hidroponia, qualquer espécie de planta. (EMBRAPA, 2000)

Os sistemas hidropônicos diferem entrei si pela forma em que a solução nutritiva entra em contato com as raízes. Basicamente, para um conjunto hidropônico é necessária uma estrutura para sustentação da planta, um reservatório para solução nutritiva, um meio de contato entre as raízes e a solução nutritiva. A hidroponia deve preferencialmente ser conduzida num ambiente protegido como uma estufa, quando se tem interesse comercial e de produção em grande escala. Mas também é possível ter uma pequena horta hidropônica no quintal da sua casa devido à simplicidade da estrutura necessária para um pequeno cultivo hidropônico.

As plantas são cultivadas em perfis específicos, 80 cm acima do solo, por onde circula uma solução nutritiva composta de água pura e de nutrientes dissolvidos de forma balanceada, de acordo com a necessidade de cada espécie vegetal. Esses perfis provêm o meio de sustentação para as plantas, sem necessidade de pedrinhas ou areia. A solução nutritiva tem um controle rigoroso para manter suas características. Periodicamente é feito um monitoramento do pH e da concentração de nutrientes, assim as plantas crescem sob as melhores condições possíveis.

Essa solução fica guardada em reservatórios e é bombeada para os perfis, conforme a necessidade, retornando para o mesmo reservatório.

2.2. Nutrição

As hortas hidropônicas possuem necessidades específicas, onde, as plantas necessitam de 16 elementos, dos quais 13 são nutrientes minerais. De acordo com as quantidades requeridas, esses minerais são classificados em macro e micronutrientes. Os macros nutrientes são: o nitrogênio (N), o fósforo (P), o potássio (K), o cálcio (Ca), o magnésio (Mg) e o enxofre (S). Os micronutrientes são: o boro (8), o cloro (CI), o cobre (Cu), o ferro (Fe), o manganês (Mn), o molibdênio (Mo) e o zinco (Zn). Além dos macros e micronutrientes minerais, a planta necessita do carbono (C), do hidrogênio (H) e do oxigênio (O), que são providos pelo ar e pela água. (Embrapa, 2020)

No comércio especializado é possível encontrar diversos tipos de produtos e maneiras para realizar esse processo, que possuem todos os macros e micronutrientes necessários para o plantio, a figura 2 abaixo apresenta alguns dos tipos de nutrientes comercializados para hidroponia disponíveis nas casas especializadas em jardinagem ou através da internet. Estes nutrientes possuem todos os macros e micronutrientes necessários para o plantio, devendo ser distribuído no plantio conforme o tamanho do mesmo e o modelo escolhido.

Figura 2 - Nutrição para hidroponia. Fonte: Plantpar, 2022.

2.3. Bomba hidráulica

A bomba hidráulica é de extrema importância em um sistema hidropônico, sendo a responsável pela circulação de água e nutrientes para os dutos do plantio. Contudo é necessário escolher a bomba conforme a vazão e o tipo de plantio, sendo a vazão total a soma da vazão necessária para abastecimento das plantas mais uma vazão de retorno de 20% a 50% para a aeração da solução. A vazão para abastecimento das plantas é calculada multiplicando-se o número de canais ou calhas a serem abastecidos simultaneamente pela vazão por canal nos sistemas fechados. Na literatura, os valores são discordantes sobre essa vazão, que varia conforme a espécie a ser cultivada. De um modo geral, sugere-se 0,5 L a 1 L/min na fase inicial e 1 L/min a 2 L/min na fase final em espécies de porte pequeno, como alface. De 0,75 L/min a 1 L/min na fase inicial e de 2 L/min a 5 L/min na fase adulta, para espécies de grande porte, como o tomateiro. (Embrapa, 2020)

Observa-se a partir da figura 3, uma canaleta com plantio vertical. Nesse tipo de sistema é importante que a bomba além de possuir a vazão correta, tenha a capacidade de elevação hídrica e opere em 12V (Tensão de operação dos bancos de bateria).

Figura 3 - Hidroponia em plantio vertical. Fonte Groho, 2022.

Figura 4 - Bomba hidráulica. Fonte: Mercado Livre, 2022.

2.4. Baterias

As baterias chumbo-ácido foram inventadas por volta de 1860, através da utilização de condutores de chumbo (ânodo) e dióxido de chumbo (cátodo), separados por um material isolante e submersos em uma solução aquosa de ácido sulfúrico. A finalidade básica das baterias é transformar energia química em energia elétrica e vice-versa, servindo assim de acumulador. O processo é reversível e a operação de carga e descarga pode ser executadas centenas de vezes.

Os processos de carga e descarga da bateria estão associados ao modo como ela está sendo utilizada. Quando a bateria está fornecendo energia (descarregando), o chumbo do material ativo das placas positivas (PbO2) se combina com os íons sulfato (SO4-2) do ácido sulfúrico, formando nas placas positivas o sulfato de chumbo (PbSO4). Ao mesmo tempo, ocorre uma reação similar nas placas negativas onde o chumbo poroso (Pb) se combina com os íons 18 (SO4-2) do ácido, formando também sulfato de chumbo.

Figura 5- Bateria de Chumbo-Ácida. Fonte: Unipower, 2022.

2.4.1. Capacidade Nominal

A capacidade nominal refere-se à capacidade da bateria em fornecer corrente por um determinado tempo. Esse parâmetro é quase sempre apresentado em A.h e pode ser classificado com um padrão conhecido como C-rate. O C-rate refere-se à taxa na qual uma bateria é carregada ou descarregada. Comumente são expressos da seguinte forma: C5 e C20. Conforme MIT (2008), uma taxa de C1 significa que a corrente de descarga

descarregará toda a bateria em 1 hora. Para uma bateria com capacidade de 100 Ah, isso equivale a uma corrente de descarga de 100 A. Uma taxa de C5 para esta bateria significa que a bateria deve suprir uma corrente de 500 Amperes, e uma taxa de C/2 implicaria uma corrente de descarga de 50 Amperes.

2.4.2. Capacidade Reserva

Esse parâmetro também é conhecido como capacidade reserva, tradução do termo em inglês, “Reserve Capacity” (RC). O parâmetro demonstra o tempo, em minutos, que uma bateria com tensão nominal 12 V, totalmente carregada pode fornecer 25 A até que sua tensão terminal chegue a 10,5 V. Este parâmetro visa identificar o tempo total que a bateria poderá manter os sistemas energizados que dela dependem.

2.4.3. Estado de carga

Estado de carga, indica o percentual de energia acumulada na bateria. Existe uma boa correlação entre o SoC e o OCV, porém uma simples relação entre os dois parâmetros não é possível, pois o SoC depende também das condições da bateria como por exemplo, estado de saúde da bateria.

2.4.4. Estado de Saúde

Outra medida essencial, o estado de saúde, “State of Health” (SoH) apresentado na equação 01, é um parâmetro utilizado principalmente no caso de sistemas de energia ininterrupta ou de emergência. O SoH é uma medida da capacidade que a bateria tem de fornecer a corrente especificada quando solicitada em relação a capacidade nominal.

Equação 1:

$$SoH = \frac{Capacidade\ de\ armazenamento\ medida}{Capacidade\ de\ armazenamento\ nominal}$$

2.4.5. Automação

Quando novas tecnologias são agregadas aos processos operacionais, as metodologias de trabalho ultrapassadas são substituídas. Com isso, deixam de ser lentas e burocráticas para se tornarem ágeis e simples de manusear. O resultado é a capacidade de lidar com um grande volume de dados com facilidade.

A automação pode ser entendida como um sistema constituído por dispositivos mecânicos ou eletrônicos, utilizado em fábricas e estabelecimentos comerciais, em telecomunicações, em instituições hospitalares e bancárias etc., destinado à operacionalização e controle dos processos de produção, que dispensa a intervenção direta do homem.

A automação industrial de um sistema é um procedimento mediante o qual as tarefas de produção que são realizadas por operadores humanos são transferidas a um conjunto de elementos tecnológicos levando-se em considerações possíveis eventualidades que possam ocorrer mantendo sempre a segurança e a qualidade.

Cada vez mais os segmentos de produção industrial, geração e distribuição de energia, transportes e muitos outros requerem um número crescente de novos sistemas e máquinas automatizadas. Isto se deve ao aumento da produção, aos custos mais baixos de componentes de automação e máquinas, a qualidade e estabilidade de novos produtos e à necessidade de substituir trabalhos perigosos e monótonos dos operadores.

No passado, os sistemas automatizados eram sistemas fechados que controlavam individualmente cada processo de uma instalação, mas com o passar do tempo, estes sistemas passaram a ser abertos com capacidade de abranger mais processos de forma a otimizar o funcionamento de toda a planta. Atualmente, um sistema automatizado é composto por 2 partes principais:

• Operação

• Controle

A parte operacional na automação industrial é uma parte do sistema que atua diretamente no processo e é um conjunto de elementos que fazem com que a máquina se mova e realize a operação desejada. Estes elementos que formam a parte operacional são os dispositivos de acionamento e pré-acionamento como motores, cilindros,

compressores, válvulas, pistões e também dispositivos de detecção como sensor indutivo, sensor capacitivo, sensor de visão, sensor ultrassônico, etc.

Já a parte de controle é a parte programável do sistema que geralmente é implementada com a ajuda do CLP (Controlador Lógico Programável). No passado, esta lógica era feita com relês eletromagnéticos, temporizadores, placas eletrônicas e módulos lógicos.

2.5. Microcontrolador

Um microcontrolador é, em última análise, um computador em um único chip, como apresentado na Figura 6. Esse chip contém um processador (Unidade Lógica e Aritmética – ULA), memória, periféricos de entrada e de saída, temporizadores, dispositivos de comunicação serial, dentre outros.

Os microcontroladores surgiram como uma evolução natural dos circuitos digitais devido ao aumento da complexidade dos mesmos. Atingindo um ponto em que é mais simples, mais barato e compacto, substituir a lógica das portas digitais por um conjunto de processador e software (IFMG, 2013).

O microcontrolador selecionado é o PIC16F877 (Microchip, 2001). Que possui algumas especificações importantes para o desenvolvimento desse sistema:

- Possui 2 módulos de captura, comparação e PWM.
- Um receptor e transmissor síncrono e assíncrono universal. (USART)
- Módulo de interrupção externa.

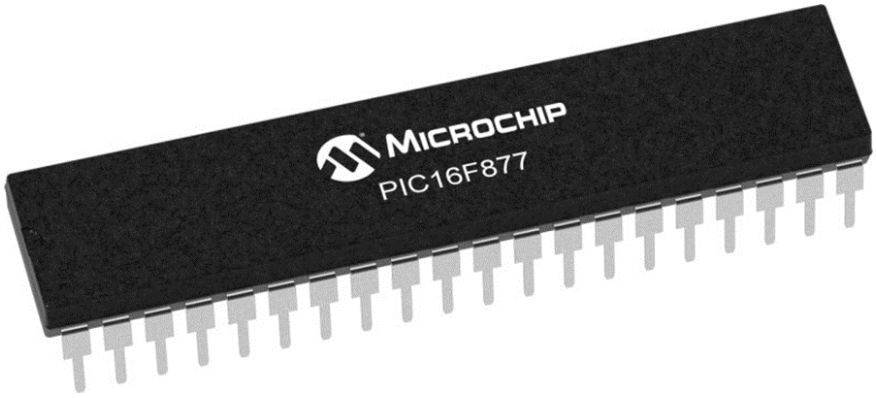

Figura 6 - MikroC PRO for PIC. Fonte: dos autores

3. Metodologia

A fim de obtermos a melhor execução para a programação e desenvolvimento do projeto, foi selecionado uma série de programas (MikroC, PyCharm, Eagle CS) que tinham a finalidade de suprir e atender a todas as funcionalidades descritas no projeto.

MikroC PRO for PIC: IDE de desenvolvimento para microcontroladores da família PIC, conforme programação apresentada no anexo 7.1.

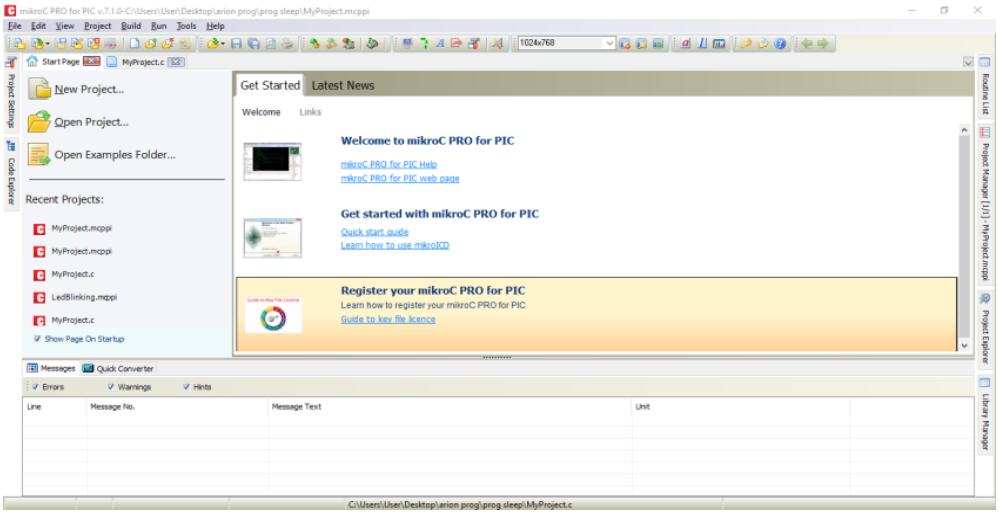

Figura 7 - MikroC PRO for PIC. Fonte: dos autores

PyCharm: Software de desenvolvimento de códigos SQL.

Figura 8 - PyCharm. Fonte: Jetbrains

Eagle CS: Software para a criação de esquemas elétricos e layouts de PCI’s para fabricação com emissão de arquivos Gerbers.

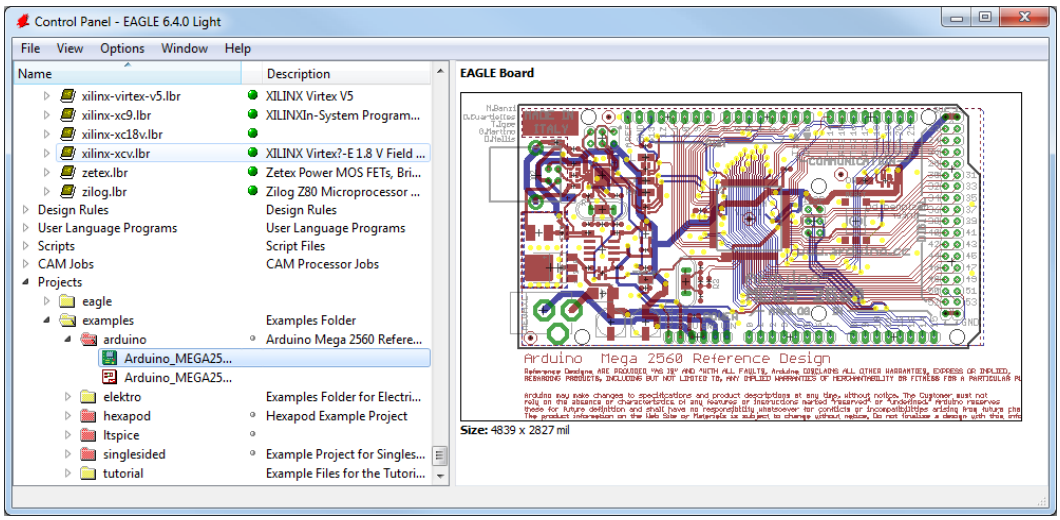

Figura 9 - Eagle CS. Fonte: dos autores

4. Resultados e discussão

4.1. Circuito Elétrico

O circuito é composto por dois retificadores capazes de alterar a corrente alternada (AC) em corrente contínua (DC). O retificador composto por D3, D4, D5 e D6, junto com o acoplador óptico OK2, são responsáveis por gerar um sinal para o microcontrolador (IC5) reconhecer a presença ou a falta de energia pela rede elétrica. O retificador B1 fica responsável pela alimentação principal do circuito e deve ter a capacidade de entregar toda a corrente necessária para alimenta-lo. Composta pela corrente da bomba, das 3 válvulas de alimentação do sistema, do circuito do carregador do banco de baterias e do circuito digital.

A partir da equação 02, é possível descrever a corrente necessária do sistema:

Equação 02:

$$It = Ib + (3 * Iv) + Ic + Is$$

$$It = 1{,}2 + (3 * 0{,}5) + 0{,}9 + 0{,}4$$

$$It = 1{,}2 + 1{,}5 + 0{,}9 + 0{,}4$$

$$It = 4\text{A}$$

Sendo Ib a corrente da bomba, Iv a corrente da válvula, Ic a corrente do carregador, Is a corrente do sistema e It a corrente total.

Como margem de segurança será considerada que o sistema completo utiliza 4,5A.

O circuito do carregador de baterias foi extraído de um documento da Universidade Federal do Paraná, onde é utilizado um regulador de tensão LM317 (IC2) para ajustar a tensão necessária para a carga da bateria. O amplificador LM741 (IC3) está sendo utilizado nessa aplicação como um comparador de tensão, onde sua saída controla a tensão de saída do regulador e aciona no final da carga o acoplador óptico OK1, que enviará um sinal para o microcontrolador.

Os componentes IC1 e IC4 são responsáveis pelo ajuste da tensão padrão para o funcionamento do circuito, respectivamente 12V (Circuito de potência) e 5V (Circuito digital).

No circuito existem três reles (K1, K2 e K3), K1 é responsável pela comutação dos 12V, definindo se essa tensão será proveniente da rede elétrica ou do banco de baterias, K2 e K3 controlam as baterias, decidindo se elas estarão no carregador ou se alimentam o circuito principal.

Para o monitoramento de descarga profunda da bateria (de acordo com o fabricante 10,3V), foi dimensionando um divisor resistivo que envia o nível de tensão da bateria para uma das portas analógicas do microcontrolador, contudo o mesmo não pode receber essa tensão diretamente (máximo 5V), as fórmulas descritas abaixo na equação 3 descrevem como foi decidido os valores dos resistores R25 e R26:

Equação 3:

$$Vout = R26/(R25 + R26) * Vin$$

$$5 = 330000/(R25 + 330000) * 15$$

$$5R25 + 1650000 = 4950000$$

$$R25 = 3300000/5$$

$$R25 = 660000\ ou\ 660K\Omega\ \approx 680K\Omega$$

Se Vin=15V, portanto Vout=4,9V e se Vin=10,3V, logo Vout=3,36V. Sendo Vin a tensão de entrada e Vout a tensão de saída.

O microcontrolador, como observado na figura 10, será responsável pelo controle do circuito e de quando devem ser acionadas as interfaces que controlam a plantação. Essas interfaces são acionadas pelos Mosfets Q7, Q8, Q9, Q10. O pino que ativa a bomba (ATVBMB) possui o controle PWM, facilitando a quantidade de vazão que o sistema necessita. Q4, Q5 e Q6 acionam os Reles.

Toda a comunicação entre o sistema e o aplicativo desenvolvido será feito por interface USB, a placa PL2303 é instalada no conector SV1, e a mesma se comunica com o microcontrolador pela porta serial.

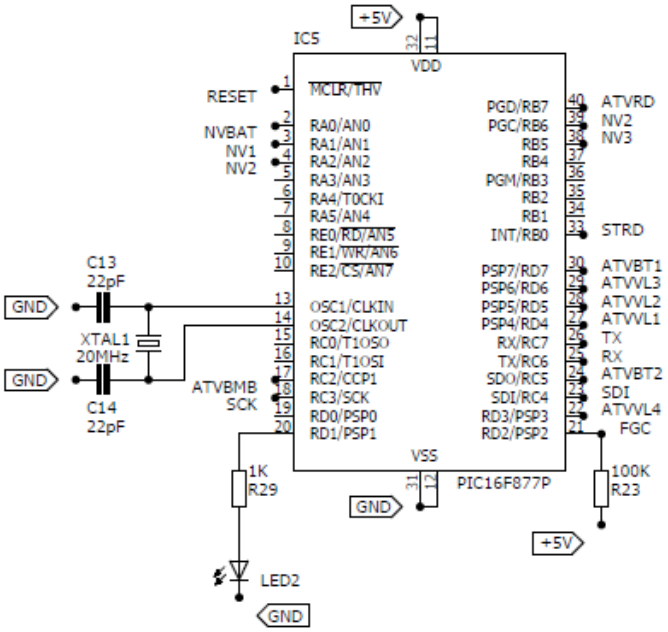

Figura 10 – Microcontrolador. Fonte: dos autores.

A figura 11 apresenta o circuito que analisará a existência de tensão na linha, gerando um sinal para análise do microcontrolador, que no caso da existência da tensão, carregará as baterias se necessário e na falta dela, realizará o chaveamento para uso das baterias como fonte de energia para o sistema.

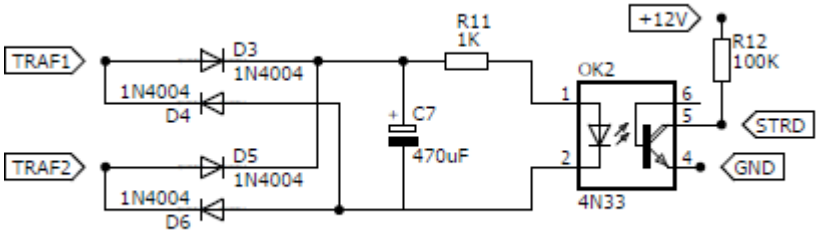

Figura 11 – Detector de falta de rede. Fonte: dos autores.

Como observado no circuito responsável pela carga das baterias, na figura 12, o regulador de tensão (LM317) ajusta a tensão do circuito de acordo com o sinal enviado pelo LM741.

O LM741 está atuando no circuito como um comparador onde, ao identificar uma corrente de 510mA, a mesma é analisada pelo resistor de potência, convertendo esse sinal em uma tensão, acionando o acoplador óptico indicando fim de carga. Esse sinal é

enviado para o microcontrolador, que realizará o chaveamento dos relés, que mostra o funcionamento lógico do microcontrolador no projeto.

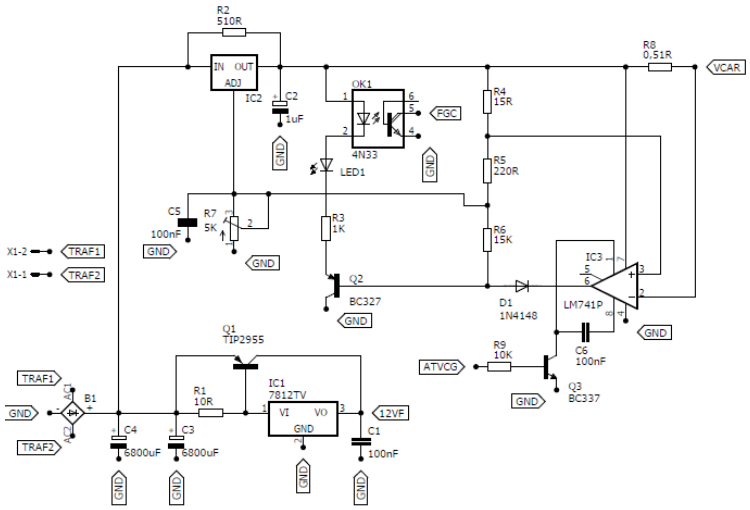

Figura 12 – Sistema de carga de baterias. Fonte: dos autores

4.2. Confecção da PCI (Placa de Circuito Impresso)

A placa foi confeccionada, como apresentada na figura 13, tendo como atenção a disposição dos componentes de alimentação e bateria localizados no mesmo local. Facilitando a dissipação de calor.

O microcontrolador, foi posicionado o mais distante possível da fonte e dos relés, a fim de se evitar possíveis ruídos, que poderiam causar mal funcionamento.

Conforme definido na equação 2, é necessário considerar durante a fabricação do protótipo trilhas mais grossas, conforme figura 14, devido ao que é tratado na primeira e segunda Lei de Ohm.

A proporcionalidade entre a corrente e a diferença de potencial observada em alguns tipos de materiais é hoje conhecida como a primeira lei de Ohm, e os componentes que apresentam essa propriedade são chamados de ôhmicos. A razão V / I, sendo V a tensão e I a corrente, denota o quanto de tensão tem de ser aplicada para passar certa corrente em um dispositivo de circuito. Assim, quanto maior for a dificuldade que o dispositivo impõe a passagem da corrente, maior deve ser a tensão aplicada para estabelece um certo valor de corrente. Logo se diz que a razão V/I é uma medida da dificuldade imposta pelo dispositivo à passagem da corrente elétrica e por isso é denominada de resistência elétrica (R). A unidade de resistência no SI foi denominada Ohm (**Ω**) em homenagem a Georg Simon Ohm. A formulação matemática dessa lei, de acordo com a equação 4 é:

Equação 4:

$$V = R * I$$

Outra observação feita por Ohm em seus experimentos foi que a resistência elétrica é proporcional ao comprimento do condutor e inversamente proporcional a área da seção transversal, conforme equação 5, o que ficou conhecido como a segunda lei de Ohm, o que pode ser escrita como:

Equação 5:

$$R = p * \frac{L}{A}$$

O coeficiente de proporcionalidade é conhecido como resistividade, e é uma característica de cada material. A tabela 1 mostra a resistividade de alguns deles. (IFCS, 2022)

Tabela 1 – Resistividade de alguns materiais

Material	P(Ω.m)	Material	P(Ω.m)
Prata	$1{,}59 * 10^{-8}$	Germânio	$4{,}6 * 10^{-1}$
Cobre	$1{,}7 * 10^{-8}$	Silício	$6{,}4 * 10^{2}$
Alumínio	$2{,}82 * 10^{-8}$	Parafina	10^{17}

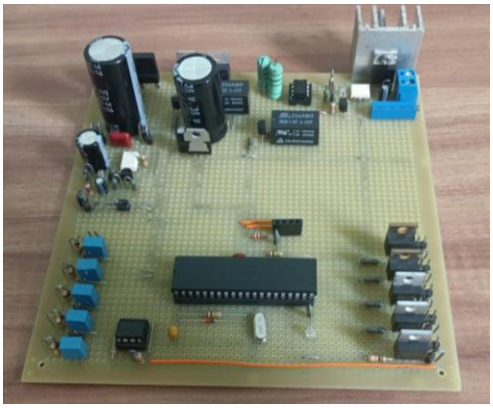

Figura 13 – Vista superior do protótipo. Fonte: dos autores

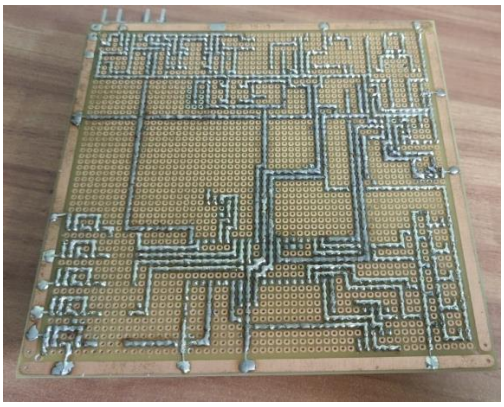

Figura 14 – Vista inferior do protótipo. Fonte: dos autores

4.3. Programação do microcontrolador

O microcontrolador, necessita que haja uma programação que defina a sequência de suas funcionalidades e consequentemente controlem o funcionamento da placa como um todo.

O fluxograma abaixo, na figura 15 apresenta o funcionamento do sistema. Onde se pode observar que após dado o início ao processo, é tomada uma decisão pelo operador. As decisões que podem ser tomadas na interação com a interface, para exibição de informações sobre o plantio e estado da bateria podem ser observadas na figura 16.

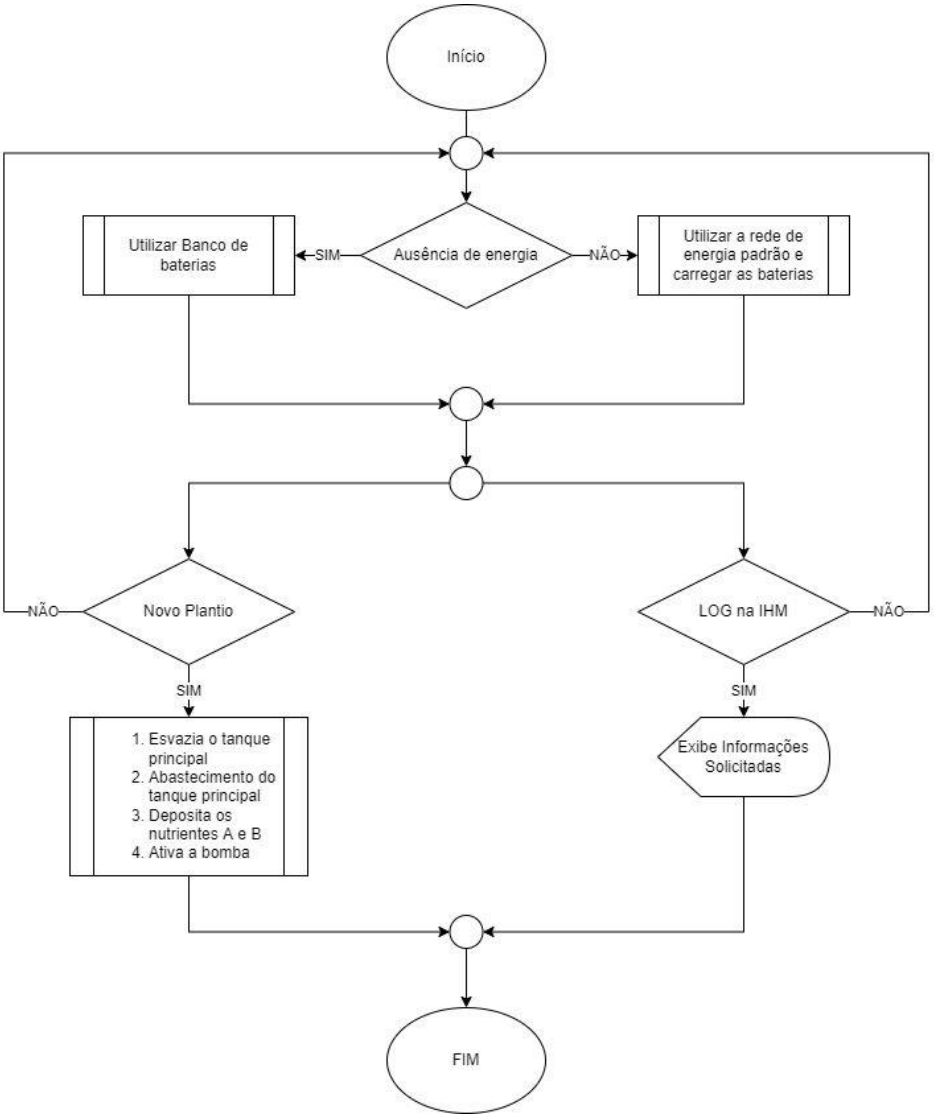

Figura 15 – Funcionamento do microcontrolador em fluxograma. Fonte: dos autores

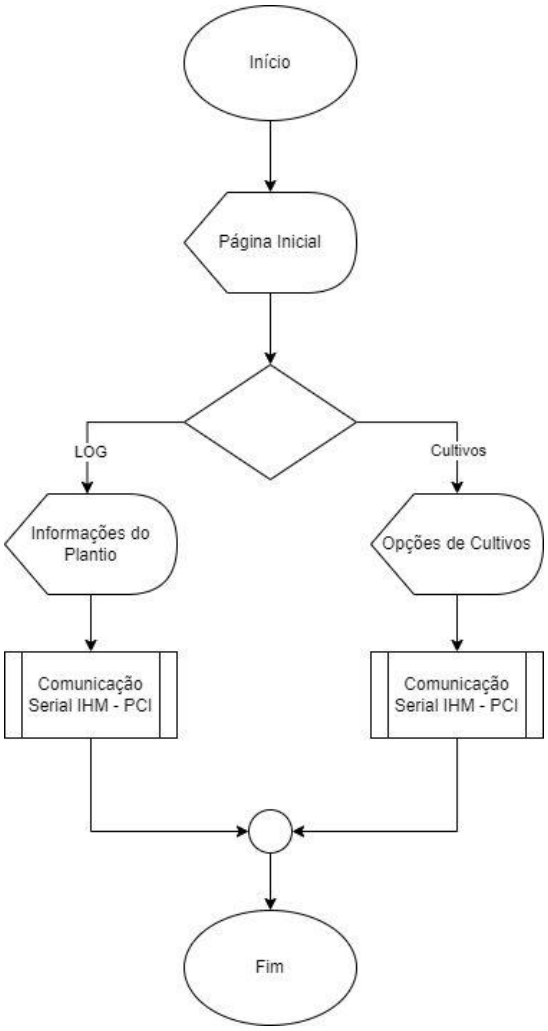

Figura 16 – Funcionamento da interface em fluxograma. Fonte: dos autores

A partir do fluxograma e das funcionalidades mapeadas, foi desenvolvido um programa em linguagem C, podendo ser analisado no anexo 7.1. O mesmo foi compilado e inserido na memória do microcontrolador. Essa programação cumpre as funções de:

- Identificar a falta de energia na rede elétrica;
- Acionar os relés nos momentos necessários;
    - Falta de energia;
    - Carga das baterias.
- Monitorar o início e fim de carga das baterias;
- Identificar descargas profundas da bateria;
- Executar a comunicação serial entre o microcontrolador e a porta USB de um computador;

- Receber os níveis lógicos apresentados pelas boias localizadas nos tanques;
- Acionar a bomba e as válvulas hidráulicas.

4.4. Programação da IHM

A plataforma de interface homem máquina (IHM) foi desenvolvida a partir das bibliotecas:

- Tkinter.
- PySerial

Tkinter é uma biblioteca da linguagem Python que acompanha a instalação padrão e permite desenvolver interfaces gráficas. Isso significa que qualquer computador que tenha o interpretador Python instalado é capaz de criar interfaces gráficas usando o Tkinter, com exceção de algumas distribuições Linux, exigindo que seja feita o download do módulo separadamente (DevMedia, 2016).

O módulo PySerial permite o acesso para a porta serial. Ele fornece funcionalidades para a linguagem Python ser executada no Windows, OSX, Linux, BSD e IronPython. O módulo chamado “serial” seleciona automaticamente a funcionalidade apropriada (PySerial, 2020).

O objetivo da IHM é permitir com que o produtor tenha uma melhor visualização das informações extraídas do plantio. Conforme a figura 17, o aplicativo disponibiliza as funcionalidades de se criar um novo plantio e verificar a situação atual da produçãoOI.

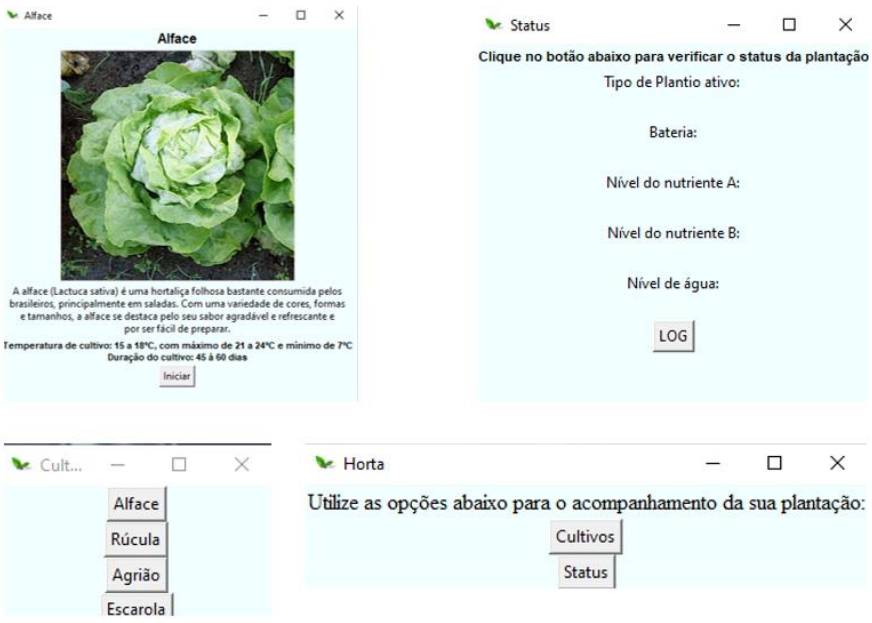

Figura 17 – Interface homem maquina. Fonte: dos autores.

5. Conclusões ou Considerações Finais

Após a confecção do protótipo e de testes, foi observado que o sistema de backup de baterias se mostrou eficiente, garantindo a carga completa das baterias. Em simulações de queda da rede elétrica, a placa realizou de forma eficaz a comutação de maneira que as baterias se tornaram a principal fonte de alimentação do sistema, de forma que não foram observados espúrios, que comprometessem o microcontrolador. A bateria, com carga completa, pode alimentar o sistema por até 4h.

Em testes de carga total, ambas as fontes tiveram capacidade de alimentar o circuito por completo. Não apresentando aquecimentos acima de 60°C nos componentes, o que resultaria na redução da eficácia do sistema e da vida útil de seus componentes.

O microcontrolador, ao se comunicar com o aplicativo, executou suas funções sem gargalos, apresentando a interface de comunicação e dados atualizados sobre o sistema de backup.

Como sugestão para novos trabalhos pode ser desenvolvida o acréscimo de novas funcionalidades ao projeto. Sendo elas o acréscimo de novos plantios no aplicativo, habilitando novas possibilidades aos usuários. A portabilidade do aplicativo para celular, este só estando disponível para notebooks até o momento. E a implementação do “sleep mode” para o microcontrolador, no caso da falta de energia.

Além dos pontos de melhoria apresentados, como medida para a redução da corrente da placa, serão necessários estudos e testes que possibilitem a redução da frequência do cristal oscilador do microcontrolador. Essa implementação pode reduzir o consumo de corrente da placa.

6. Referências

Epamig. **Cultivo hidropônico de hortaliças é oportunidade para pequenos e grandes produtores**. Disponível em: <https://epamig.wpcomstaging.com/2020/04/23/cultivo-hidroponico-de-hortalicas-e-oportunidade-para-pequenos-e-grandes-produtores/>. Acesso em: fev.2022.

Embrapa. **Hortaliças hidropônicas**. Disponível em: <https://www.embrapa.br/hortalica-nao-e-so-salada/hortalicas-hidroponicas>. Acesso em: fev.2022.

Embrapa. **Princípios de hidropônia**. Disponível em: <https://www.embrapa.br/busca-de-publicacoes/-/publicacao/769981/principios-de-hidroponia>. Acesso em: fev.2022.

UFPR. **Battery Chargers and Zappers**. Disponível em: <http://eletrica.ufpr.br/graduacao/e-boks/Electronics%20Power%20Supply%20and%20Battery%20Charger%20Circuit%20Encyclopedia.pdf>. Acesso em: fev.2022.

STA. **Vantagens e Limitações das Baterias Seladas de Chumbo-Ácido**. Disponível em: <https://www.sta-eletronica.com.br/artigos/baterias-recarregaveis/baterias-de-chumbo/vantagens-e-limitacoes-das-baterias-seladas-de-chumbo-acido>. Acesso em: fev.2022.

GROHO. **Hydro 40 Vertical Fixed**. Disponível em: <https://www.groho.pt/product/horta-caseira-vertical-40plantas>. Acesso em: fev/2022.

Microchip. **PIC16F87X Data Sheet**. Disponível em: <https://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf>. Acesso em: abr/2022.

PENIDO, E; TRINDADE, R. Microcontroladores. **Instituto Federal de Educação, Ciência e Tecnologia.** Disponível em: <https://www2.ifmg.edu.br/ceadop3/apostilas/microcontroladores>

DevMidia. **Tkinter: Interfaces gráficas em Python**. Disponível em: <https://www.devmedia.com.br/tkinter-interfaces-graficas-em-python/33956>. Acesso em: Mar/2022.

PySerial. **PySerial Overview**. Disponível em: <https://pyserial.readthedocs.io/en/latest/pyserial.html >. Acesso em: Jun/2022.

IFSC. **Resistência e corrente elétrica**. Disponível em: <https://www.ifsc.usp.br/~strontium/Teaching/Material2010-2%20FFI0106%20LabFisicaIII/04-ResistenciaCorrenteEletrica.pdf>. Acesso em: Jun/2022.

7. Anexos

7.1. Programação (Microcontrolador)

```
#include <string.h>
#define NVBAT   RA0_bit
#define NV1     RA1_bit
#define NV2     RA2_bit
#define STRD    RB0_bit
#define NV3     RB5_bit
#define NV4     RB6_bit
#define ATVRD   RB7_bit
#define ATVBMB  RC2_bit
#define ATVVL4  RD0_bit
#define TESTE   RD1_bit
#define FGC     RD2_bit
#define ATVVL1  RD4_bit
#define ATVVL2  RD5_bit
#define ATVVL3  RD6_bit
#define ATVBT1  RD7_bit

int bc=0, a = 0, b = 0, trava = 0, i = 0;
unsigned int leitura = 0;
char uart_rd, envia[7] = {'#','0','0','0','0','0','0'}, log_envia;

void main() {
    ADCON0 = 0x01;
```

```
ADCON1 = 0X0E;


UART1_Init(9600);
Delay_ms(100);


      //76543210
TRISA = 0b00000111;
TRISB = 0b11100001;
TRISC = 0b10000000;
TRISD = 0b00000100;
PORTA = 0b00000000;
PORTB = 0b00000000;
PORTC = 0b00000000;
PORTD = 0b00000000;


 while(1){
       //leitura = ADC_Read(0);


       if(UART1_Data_Ready()) {
             uart_rd = UART1_READ();
       }
       switch(uart_rd)
       {
             case 'a':
                do{
                  if(NV1 == 0 && NV2 == 0 && trava == 0){
                       ATVVL1 = 1;
```

```
            ATVVL2 = 0;
        }
        else if(NV1 == 0 && NV2 == 1 && trava == 0){
            ATVVL1 = 1;
            ATVVL2 = 0;
            trava = 1;
        }
        else if(NV1 == 1 && NV2 == 1){
            ATVVL1 = 0;
            ATVVL2 = 1;
        }
        else if(NV1 == 0 && NV2 == 0 && trava == 1){
            ATVVL1 = 0;
            ATVVL2 = 0;
            ATVVL3 = 1;
            delay_ms(10000);
            ATVVL3 = 0;
            ATVVL4 = 1;
            delay_ms(10000);
            ATVVL4 = 0;
            envia[1] = 'A';
            uart_rd = 0;
        }
    }while(uart_rd != 0);
    break;
case 'b':
    do{
```

```
    if(NV1 == 0 && NV2 == 0 && trava == 0){
        ATVVL1 = 1;
        ATVVL2 = 0;
    }
    else if(NV1 == 0 && NV2 == 1 && trava == 0){
        ATVVL1 = 1;
        ATVVL2 = 0;
        trava = 1;
    }
    else if(NV1 == 1 && NV2 == 1){
        ATVVL1 = 0;
        ATVVL2 = 1;
    }
    else if(NV1 == 0 && NV2 == 0 && trava == 1){
        ATVVL1 = 0;
        ATVVL2 = 0;
        ATVVL3 = 1;
        delay_ms(10000);
        ATVVL3 = 0;
        ATVVL4 = 1;
        delay_ms(10000);
        ATVVL4 = 0;
        envia[1] = 'B';
        uart_rd = 0;
    }
  }while(uart_rd != 0);
  break;
```

```
case 'c':
    do{
      if(NV1 == 0 && NV2 == 0 && trava == 0){
            ATVVL1 = 1;
            ATVVL2 = 0;
      }
      else if(NV1 == 0 && NV2 == 1 && trava == 0){
            ATVVL1 = 1;
            ATVVL2 = 0;
            trava = 1;
      }
      else if(NV1 == 1 && NV2 == 1){
            ATVVL1 = 0;
            ATVVL2 = 1;
      }
      else if(NV1 == 0 && NV2 == 0 && trava == 1){
            ATVVL1 = 0;
            ATVVL2 = 0;
            ATVVL3 = 1;
            delay_ms(10000);
            ATVVL3 = 0;
            ATVVL4 = 1;
            delay_ms(10000);
            ATVVL4 = 0;
            envia[1] = 'C';
            uart_rd = 0;
      }
```

```
    }while(uart_rd != 0);
    break;
case 'd':
    do{
      if(NV1 == 0 && NV2 == 0 && trava == 0){
            ATVVL1 = 1;
            ATVVL2 = 0;
      }
      else if(NV1 == 0 && NV2 == 1 && trava == 0){
            ATVVL1 = 1;
            ATVVL2 = 0;
            trava = 1;
      }
      else if(NV1 == 1 && NV2 == 1){
            ATVVL1 = 0;
            ATVVL2 = 1;
      }
      else if(NV1 == 0 && NV2 == 0 && trava == 1){
            ATVVL1 = 0;
            ATVVL2 = 0;
            ATVVL3 = 1;
            delay_ms(10000);
            ATVVL3 = 0;
            ATVVL4 = 1;
            delay_ms(10000);
            ATVVL4 = 0;
            envia[1] = 'D';
```

```
                uart_rd = 0;
              }
            }while(uart_rd != 0);
            break;
        case 'l':
            log_envia = envia;
            UART1_Write_Text(log_envia);
            uart_rd = 0;
            break;
}
if(leitura > 700 && a == 0){
        a = 1;
        delay_ms(100);
}
if(leitura <= 700 && a == 1){
        a = 0;
        b = b + 1;
        delay_ms(100);
}
if(b >= 200){
         //envia[1] = 'R';
}
if(b < 199){
         //envia[1] = 'B';
}
if(STRD == 1){
        ATVRD = 0;
```

```
        envia[2] = 'D';
        delay_ms(10);
    }
    if(STRD == 0){
        ATVRD = 1;
        envia[2] = 'L';
        delay_ms(10);
    }
    if(STRD == 0 && FGC == 0 && bc == 0){
        ATVBT1 = 1;
        bc = 1;
        delay_ms(10);
    }
    if(STRD == 0 && FGC == 1 && bc == 1){
        ATVBT1 = 1;
        bc = 2;
        delay_ms(10);
    }
    if(STRD == 0 && FGC == 0 && bc == 2){

        ATVBT1 = 0;
    }
    if(NV4 == 0){
       envia[3] = 'A';
    }
    if(NV4 == 1){
       envia[3] = 'B';
```

```
            }
            if(NV3 == 0){
                envia[4] = 'A';
            }
            if(NV3 == 1){
                envia[4] = 'B';
            }
            if(NV2 == 0){
                envia[5] = 'A';
            }
            if(NV2 == 1){
                envia[5] = 'B';
            }
            if(NV1 == 0){
                envia[6] = 'A';
                TESTE = 0;
            }
            if(NV1 == 1){
                envia[6] = 'B';
                TESTE = 1;
            }
        }
    }
```

7.2. Programação (Aplicativo)

```
from tkinter import *

from turtle import textinput


import self as self

import serial


menu_inicial = Tk()


plantio = StringVar()

energia = StringVar()

nv4 = StringVar()

nv3 = StringVar()

nva = StringVar()


menu_inicial.title("Horta")


# menu_inicial.geometry("500x200")

menu_inicial.resizable(True, True)

menu_inicial.iconbitmap("imagens/logo.ico")

menu_inicial['bg'] = '#F0FFFF'
```

```
microcontrolador = serial.Serial('COM3', 9600)

# Texto orientação


texto_orientaçao = Label(menu_inicial,

                    text="Utilize as opções abaixo para o acompanhamento da sua plantação:",

                    bg='#F0FFFF',

                    fg='#000000',

                    font="Times")

texto_orientaçao.grid(column=0, row=0)


def btnplt():

    # top level (Cultivos)

    cultivos = Toplevel()

    cultivos.title("Cultivos")

    cultivos.geometry("200x100")

    cultivos.iconbitmap("imagens/logo.ico")

    cultivos['bg'] = '#F0FFFF'


    def btnalf():

        # top level (Alface)
```

```
alface = Toplevel()

alface.title("Alface")

alface.geometry("430x500")

alface.iconbitmap("imagens/logo.ico")

alface['bg'] = '#F0FFFF'



texto_alface = Label(alface, text="Alface", bg='#F0FFFF', fg='#000000', font="Arial 12 bold")

texto_alface.grid(column=0, row=0)



self.img = PhotoImage(file="imagens/alface.png")

label_imagem = Label(alface, image=self.img)

label_imagem.grid(column=0, row=1)



texto_alface = Label(alface,

                     text="A alface (Lactuca sativa) é uma hortaliça folhosa bastante consumida pelos\nbrasileiros, principalmente em saladas. Com uma variedade de cores, formas\ne tamanhos, a alface se destaca pelo seu sabor agradável e refrescante e\npor ser fácil de preparar.",

                     bg='#F0FFFF',

                     fg='#000000')

texto_alface.grid(column=0, row=2)
```

```
    texto_alface = Label(alface,

                         text="Temperatura de cultivo: 15 a 18ºC, com máximo de 21 a 24ºC e mínimo de 7ºC\nDuração do cultivo: 45 à 60 dias",

                         bg='#F0FFFF',

                         fg='#000000',

                         font="Arial 8 bold")

    texto_alface.grid(column=0, row=6)



    def btnipa():

        while True:  # Loop principal

            microcontrolador.write('a'.encode())

            microcontrolador.flush()

            self.serial.Serial.close()



    btnipa = Button(alface, text="Iniciar", command=btnipa)

    btnipa.grid(column=0, row=8)



  def btnrcl():

    # top level (Rúcula)

    rúcula = Toplevel()

    rúcula.title("Rúcula")

    rúcula.geometry("400x500")
```

```
rúcula.iconbitmap("imagens/logo.ico")

rúcula['bg'] = '#F0FFFF'



texto_rúcula = Label(rúcula, text="Rúcula", bg='#F0FFFF', fg='#000000', font="Arial 12 bold")

texto_rúcula.grid(column=0, row=0)



self.img = PhotoImage(file="imagens/rúcula.png")

label_imagem = Label(rúcula, image=self.img)

label_imagem.grid(column=0, row=1)



texto_rúcula = Label(rúcula,

                     text="A rúcula é um vegetal verde-escuro rico em vitaminas A e C e compostos\nfenólicos, como luteína e zeaxantina, com propriedades antioxidantes e\nanti-inflamatórias, que ajudam a melhorar a saúde dos olhos, controlar os\nníveis de açúcar no sangue e a prevenir o desenvolvimento de doenças\ncardiovasculares.",

                     bg='#F0FFFF',

                     fg='#000000')

texto_rúcula.grid(column=0, row=2)



texto_rúcula = Label(rúcula, text="Temperatura de cultivo: 16 à 22°C\nDuração do cultivo: 20 à 65 dias",

                     bg='#F0FFFF',

                     fg='#000000',
```

```
                    font="Arial 8 bold")
    texto_rúcula.grid(column=0, row=7)

    def btnipr():
        while True:  # Loop principal
            microcontrolador.write('b'.encode())
            microcontrolador.flush()  # Limpa a comunicação1
            self.serial.Serial.close()

    btnipr = Button(rúcula, text="Iniciar", command=btnipr)
    btnipr.grid(column=0, row=8)

def btnagr():
    # top level (Agrião)
    agrião = Toplevel()
    agrião.title("Agrião")
    agrião.geometry("400x500")
    agrião.iconbitmap("imagens/logo.ico")
    agrião['bg'] = '#F0FFFF'

    texto_agriao = Label(agrião, text="Agrião", bg='#F0FFFF', fg='#000000', font="Arial 12
bold")
```

```
texto_agriao.grid(column=0, row=0)



self.img = PhotoImage(file="imagens/agrião.png")

label_imagem = Label(agrião, image=self.img)

label_imagem.grid(column=0, row=1)



texto_agriao = Label(agrião,

                text="O Agrião é uma verdura rica em vitamina C, Vitamina A e flavonoides,\n compostos com propriedades e antioxidantes anti-inflamatórias que\n combatem os radicais livres e fortalecem o sistema imunológico,\n ajudando na prevenção de situações como diabetes, pressão alta e gripes,\n por exemplo.",

                bg='#F0FFFF',

                fg='#000000')

texto_agriao.grid(column=0, row=2)



texto_agriao = Label(agrião, text="Temperatura de cultivo: 12°C à 20°C\nDuração do cultivo: 40 à 50 dias",

                bg='#F0FFFF',

                fg='#000000',

                font="Arial 8 bold")

texto_agriao.grid(column=0, row=7)



def btnipag():
```

```
        while True:  # Loop principal
            microcontrolador.write('c'.encode())
            microcontrolador.flush()
            self.serial.Serial.close()

    btnipag = Button(agrião, text="Iniciar", command=btnipag)
    btnipag.grid(column=0, row=8)

def btnesc():
    # top level (Escarola)
    escarola = Toplevel()
    escarola.title("Escarola")
    escarola.geometry("400x500")
    escarola.iconbitmap("imagens/logo.ico")
    escarola['bg'] = '#F0FFFF'

    texto_escarola = Label(escarola, text="Escarola", bg='#F0FFFF', fg='#000000', font="Arial 12 bold")
    texto_escarola.grid(column=0, row=0)

    self.img = PhotoImage(file="imagens/escarola.png")
    label_imagem = Label(escarola, image=self.img)
```

```
label_imagem.grid(column=0, row=1)



texto_escarola = Label(escarola,

                       text="A escarola é uma verdura com ótimas quantidades de vitamina A,\n vitamina C e ácido cafeico, nutrientes com propriedades antioxidantes\n que combatem os radicais livres no organismo, ajudando a prevenir\n doenças, como pressão alta, diabetes e derrame.",

                       bg='#F0FFFF',

                       fg='#000000')

texto_escarola.grid(column=0, row=2)



texto_escarola = Label(escarola, text="Temperatura de cultivo: 15ºC à 25ºC\nDuração do cultivo:  80 à 100 dias",

                       bg='#F0FFFF',

                       fg='#000000',

                       font="Arial 8 bold")

texto_escarola.grid(column=0, row=7)



def btnipe():

    while True:  # Loop principal

        microcontrolador.write('d'.encode())

        microcontrolador.flush()

        self.serial.Serial.close()
```

```
    btnipe = Button(escarola, text="Iniciar", command=btnipe)
    btnipe.grid(column=0, row=8)


# Botão Alface
btnalf = Button(cultivos, text="Alface", command=btnalf)
btnalf.pack()


# Botão Rúcula
btnrcl = Button(cultivos, text="Rúcula", command=btnrcl)
btnrcl.pack()


# Botão Agrião
btnagr = Button(cultivos, text="Agrião", command=btnagr)
btnagr.pack()


# Botão Escarola
btnesc = Button(cultivos, text="Escarola", command=btnesc)
btnesc.pack()
```

```
def btnsts():

    # top level (Status)

    status = Toplevel()

    status.title("Status")

    status.geometry("325x300")

    status.iconbitmap("imagens/logo.ico")

    status['bg'] = '#F0FFFF'



    texto_status = Label(status, text="Clique no botão abaixo para verificar o status da plantação", bg='#F0FFFF',

                         fg='#000000',

                         font="Arial 8 bold")

    texto_status.grid(column=0, row=0)

    texto_status = Label(status, text="Tipo de Plantio ativo: ", bg='#F0FFFF', fg='#000000')

    texto_status.grid(column=0, row=1)

    texto_status = Label(status, textvariable=plantio, bg='#F0FFFF', fg='#000000',font="Arial 8 bold")

    texto_status.grid(column=0, row=2)



    texto_status = Label(status, text="Bateria:", bg='#F0FFFF', fg='#000000')

    texto_status.grid(column=0, row=3)

    texto_status = Label(status, textvariable=energia, bg='#F0FFFF', fg='#000000',font="Arial 8 bold")
```

```
    texto_status.grid(column=0, row=4)



    texto_status = Label(status, text="Nível do nutriente A:", bg='#F0FFFF', fg='#000000')

    texto_status.grid(column=0, row=5)

    texto_status = Label(status, textvariable=nv4, bg='#F0FFFF', fg='#000000',font="Arial 8
bold")

    texto_status.grid(column=0, row=6)



    texto_status = Label(status, text="Nível do nutriente B:", bg='#F0FFFF', fg='#000000')

    texto_status.grid(column=0, row=7)

    texto_status = Label(status, textvariable=nv3, bg='#F0FFFF', fg='#000000',font="Arial 8
bold")

    texto_status.grid(column=0, row=8)



    texto_status = Label(status, text="Nível de água:", bg='#F0FFFF', fg='#000000')

    texto_status.grid(column=0, row=9)

    texto_status = Label(status, textvariable=nva, bg='#F0FFFF', fg='#000000',font="Arial 8
bold")

    texto_status.grid(column=0, row=10)



    def btnlog():

        while True:  # Loop principal

            microcontrolador.write('l'.encode())
```

```
microcontrolador.flush()


log = microcontrolador.read(size=7)
print(log)


if(log[1]==65):
    plantio.set ("Alface")


if(log[1]==66):
    plantio.set("Rúcula")


if(log[1]==67):
    plantio.set("Agrião")


if(log[1]==68):
    plantio.set("Escarola")


if(log[1]==48):
    plantio.set("Nenhuma plantação selecionada")


if(log[2]==68):
```

```
    energia.set("Bateria")


if(log[2]==76):
    energia.set("Rede elétrica")


if(log[3]==65):
    nv4.set("Alto")


if(log[3]==66):
    nv4.set("Baixo")


if(log[4]==65):
    nv3.set("Alto")


if(log[4]==66):
    nv3.set("Baixo")


if(log[5]==65 and log[6]==65):
    nva.set("Normal")


if(log[5]==66 and log[6]==65):
```

```
            nva.set("Normal")


        if(log[5]==66 and log[6]==66):
            nva.set("Baixo")


        #if(log[6]==65):
            #nv1.set("Alto")


        #if(log[6]==66):
            #nv1.set("Baixo")


        self.serial.Serial.close()


    btnlog = Button(status, text="LOG", command=btnlog)
    btnlog.grid(column=0, row=11)


# Botão Plantações


btnplt = Button(menu_inicial, text="Cultivos", command=btnplt)
btnplt.grid(column=0, row=1)
```

```
# Botão status


btnsts = Button(menu_inicial, text="Status", command=btnsts)

btnsts.grid(column=0, row=2)


menu_inicial.mainloop()
```

7.3. Esquema elétrico