UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE BACHARELADO EM ENGENHARIA ELETRÔNICA

LOUISIE ARISTIDES STANISZEWSKI

DESENVOLVIMENTO DE UMA PLATAFORMA UTILIZANDO A INTERFACE GRÁFICA DO MATLAB® PARA MONITORAMENTO DE UM AMBIENTE DE CULTIVO CONTROLADO

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO
2016

LOUISIE ARISTIDES STANISZEWSKI

DESENVOLVIMENTO DE UMA PLATAFORMA UTILIZANDO A INTERFACE GRÁFICA DO MATLAB® PARA MONITORAMENTO DE UM AMBIENTE DE CULTIVO CONTROLADO

Trabalho de conclusão de curso apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso de Bacharelado em Engenharia Eletrônica do Departamento Acadêmico de Eletrônica – DAELN – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito para obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador: Prof. Flávio Luiz Rossini

CAMPO MOURÃO
2016

Ministério da Educação

Universidade Tecnológica Federal do Paraná

TERMO DE APROVAÇÃO

DO TRABALHO DE CONCLUSÃO DE CURSO INTITULADO

Desenvolvimento de uma plataforma utilizando a interface gráfica do MATLAB® para monitoramento de um ambiente de cultivo controlado.

por

Louisie Aristides Staniszewski

Trabalho de Conclusão de Curso apresentado no dia 22 de Novembro de 2016 ao Curso Superior de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná, Campus Campo Mourão. O Candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____

Prof. Marcio Rodrigues da Cunha

(UTFPR)

_____

Prof. Roberto Ribeiro Neli

(UTFPR)

_____

Prof. Flávio Luiz Rossini

(UTFPR)

Orientador

A Folha de Aprovação assinada encontra-se na Coordenação do Curso

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus, pela minha vida, por ter me dado forças em momentos difíceis, e me dado tantas alegrias nessa trajetória.

Agradeço de forma especial aos meus pais Edson Staniszewski e Rosélia Aristides Staniszewski por todo apoio e amor dedicado a mim durante toda minha vida. Ao meu pai por me ensinar os maiores valores que se pode ter na vida, sempre me apoiando e me ajudando para que eu completasse esse ciclo, e em todo momento sendo um pai presente e carinhoso. A minha mãe por ser meu exemplo de mulher e uma mãe extremamente companheira e amorosa, por toda compreensão, principalmente nos momentos em que eu mais precisava. Se eu tivesse a oportunidade de escolher meus pais, escolheria vocês sem dúvida, os melhores pais que alguém pode ter... Amo vocês!

Agradeço meu namorado Boris Hugo Westphal pelo companheirismo, pelos abraços, pela paciência em minhas ausências, por todo incentivo e carinho.

Agradeço a minha família que sempre se fez presente e me incentivaram, me dando conforto e palavras de amor, me fazendo sentir amparada em qualquer situação. Minhas avós Iraci e Nádia que sempre fazem os melhores almoços de domingo. Minhas tias madrinhas Claudete e Claudia que são como mães para mim. Meus tios Clovis, Marcio e Nelson que me tratam como a princesinha na família. Meus compadres Marlus e Thays que me deram o melhor presente no mundo, minha afilhada Antônia. A todos vocês sempre serei grata.

Agradeço também aos meus amigos. Da faculdade que trilharam esse caminho comigo, passando finais de semana e feriados estudando, mas também com momentos de descontração que ajudaram muito nesses anos. Da Agência Luar que são as meninas mais lindas e queridas. Do Cursilho que se tornaram uma família. E amigos de infância que não importa o tempo que passe sempre estarão comigo.

Por fim agradeço aos professores que contribuíram para a minha formação, em especial a Prof.ª Luciane Agnoletti e ao Profº Flávio Rossini que me orientaram e ajudaram durante a realização desse trabalho.

A todos, muito obrigada!

RESUMO

STANISZEWSKI, Louisie Aristides. DESENVOLVIMENTO DE UMA PLATAFORMA UTILIZANDO A INTERFACE GRÁFICA DO MATLAB® PARA MONITORAMENTO DE UM AMBIENTE DE CULTIVO CONTROLADO. Trabalho de Conclusão de Curso – Bacharelado em Engenharia Eletrônica, Universidade Tecnológica Federal do Paraná. Campo Mourão 2016.

Este trabalho consiste no desenvolvimento de uma plataforma de monitoramento e controle de luminosidade, temperatura e umidade, através do uso da interface gráfica do MATLAB, sendo direcionada a um ambiente de cultivo. O mesmo visa apresentar uma metodologia de monitoramento e controle otimizada, com uma interface amigável para o usuário, a qual possibilita a gravação e posterior análise dos dados obtidos e assim podendo melhorar a produtividade em uma estufa. Portanto é apresentada uma análise dos sensores a serem utilizados, a mesma justifica as respectivas escolhas, e um estudo sobre o *hardware* Arduino® e o *software* MATLAB®, que em conjunto realizam a leitura e análise dos dados. A partir do estudo realizado pôde-se programar o *software* da plataforma para o usuário. O mesmo foi desenvolvido para o usuário final, é uma plataforma autoexplicativa com funções que permitem que o usuário possa monitorar a luminosidade, a temperatura e a umidade do ambiente e realizam o controle das mesmas quando o *software* está em execução. Após testes realizados chegou-se à conclusão que a plataforma é eficiente e que o presente projeto pode ser utilizado em diversas aplicações que tenham por objetivo realizar o monitoramento e controle das variáveis citadas.

Palavras-chave: MATLAB®, Arduino®, Monitoramento, Controle, Sensoriamento.

ABSTRACT

STANISZEWSKI, Louisie Aristides. DEVELOPMENT OF A PLATFORM USING THE GRAPHIC INTERFACE OF MATLAB® FOR MONITORING AND CONTROL OF AN CULTIVATION ENVIRONMENT. Trabalho de Conclusão de Curso – Bacharelado em Engenharia Eletrônica, Universidade Tecnológica Federal do Paraná. Campo Mourão 2016.

This work is a development of a monitoring and control platform for lighting, temperature and humidity, through the use of the graphic interface of MATLAB, being directed to a cultivation environment. It aims to present a monitoring and control optimized methodology, with a friendly interface for the user, allowing the recording and later analysis of the data and thus may improve productivity in a greenhouse. Therefore is presented an analysis of the sensors to be used, justifying the choice of the same, and a study on the Arduino® hardware and the MATLAB® software, which together perform the reading and analysis of data. From the study could be programmed the software platform for the user. The same was designed for the end user, it is a self-explanatory platform with functions that allow the user to monitor the luminosity, temperature and humidity of the environment and control them when the software is running. After tests performed, it was concluded that the platform is efficient and this project can be used in other applications that aim to carry out the monitoring and control of the mentioned variables.

Key-words: MATLAB®, Arduino®, Monitoring, Control, Sensing

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ASCII – American Standard Code for Information Interchange.

BMP – Bitmap.

BPS – Bits Per Second.

CI – Circuito Integrado.

GND – Graduated Neutral Density Filter.

GUI – Graphical User Interface.

IBM – International Business Machines.

LED - Light emitter diode.

LDR – Light Dependent Resistor.

MATLAB – Matrix Laboratory.

PSIM – Software for Power Electronics Simulation.

RTD – Resistance Temperature Detector.

TTL – Transitor-Transistor Logic.

UART – Universal Asynchronous Receiver Transmitter.

USB – Universal Serial Bus.

UTI – Unidade de Terapia Intensiva, Unidade de Tratamento Intenso.

WIMP – Window, Icon, Menu, Pointing device.

SUMÁRIO

1 INTRODUÇÃO

O primeiro tópico do presente trabalho consiste em uma breve apresentação do mesmo, assim possibilita ao leitor uma compreensão geral do mesmo.

1.1. TEMA

Desenvolveu-se uma plataforma através da interface gráfica do MATLAB® (*Matrix Laboratory*), para aquisição e visualização de valores de luminosidade, temperatura e umidade de solo, para aplicação em uma estufa.

1.1.1. Delimitação do Tema

Desde os tempos mais antigos, do Império Romano, existe a necessidade de um ambiente controlado para o cultivo de plantas. Afinal os jardineiros do Imperador Tibério, precisavam usar a imaginação para que houvesse a produção de pepino durante todo o ano, já que seu Imperador exigia este alimento à mesa todos os dias. Dessa forma iniciaram-se os estudos de produção de plantas em ambientes controlados, e assim surgiram as estufas (GUEDES, 2009).

Segundo Litjens (2009), a utilização de estufas foi se espalhando pela Europa durante o século XIII, onde havia um interesse em proteger as plantas durante o inverno rigoroso. Logo o uso de estufas foi adaptado para o uso nos quatro cantos do mundo, assim sendo evoluída e aprimorada conforme a necessidade, até chegarem às estufas mais modernas, automatizadas e controladas.

Atualmente com a escassez de recursos naturais e o aumento da população mundial, o uso de estufas é de grande importância no cultivo de alimentos. Tal tecnologia empenha esforços no sentido de suprir a demanda de determinada região, sem contar que nem todas as regiões são apropriadas para o cultivo de certas culturas, assim torna-se pertinente a construção de ambientes de cultivo controlado. A estufa por sua vez, exige investimentos

financeiros, os quais com planejamento possibilitam a rentabilidade para o investidor. Além disso, o avanço da tecnologia na área da agricultura possibilita que esta atividade cresça de forma equilibrada e sustentável (FIGUEIREDO, 2011).

As estufas, ou ambientes protegidos, são aquelas que propiciam um clima adequado, e até mesmo perto do ideal para o cultivo de plantas independente da estação do ano, e com o controle ideal podem ser utilizadas para diversos tipos de culturas (KAMPF, 2005).

Esse controle é feito através de sensores, que realizam a leitura de dados, como luminosidade, temperatura e umidade de solo, assim sendo possível monitorar as mudanças das referidas variáveis e, dessa forma, realizar um controle para obtenção das melhores condições no cultivo de determinada cultura. Deve-se levar em consideração todas as características naturais do ambiente e as desejadas, para então realizar a escolha do sensor mais apropriado e confiável para o sistema.

Com o desenvolvimento da microeletrônica, a partir da década de 70, os sensores foram sendo aperfeiçoados e diversificados, assim sendo produzidos sensores integrados, e então, sensores inteligentes. No entanto, nem sempre os dados de saída disponibilizados por um sensor são os desejados, dessa forma há a necessidade do processamento dos mesmos, para assim obter uma leitura adequada, o que pode ser realizado através de microcontrolador e *software* de apoio.

De acordo com Melo (1985) o controle de algum processo que antigamente era realizado de forma analógica, atualmente pode ser substituído por programas de controle em computador, e com o avanço desta tecnologia, é possível obter sinais precisos e em tempo real, os quais permitem uma proteção ao processo que está sendo controlado. Com a leitura dos dados provenientes de sensores, já manipulados, se torna simplificado o processamento desses dados para a construção de gráficos e tabelas, os quais potencializam a compreensão e interpretação das referidas informações pelo operador.

Como as aplicações de sensoriamento virtual por meio de programas de controle ganharam espaço no mercado, *softwares* como o MATLAB® vêm sendo utilizados em ambientes industriais e acadêmicos, uma vez que este possui um alto desempenho e possibilita a análise de dados, processamento de sinais e construção de gráficos e tabelas (GASPAR, et al., 2002). Tal *software* conta com uma interface gráfica, GUIDE, que proporciona um melhor suporte ao usuário, sendo de fácil acesso e interpretação.

1.2. PROBLEMAS E PREMISSAS

O significativo crescimento da tecnologia na área da agricultura, nos últimos 10 anos, tem possibilitado o aumento da atividade agrícola de forma equilibrada e sustentável. Ou seja, com o monitoramento e controle de estufas é possível que o produtor possa ter um aumento da sua renda, devido ao aumento da produtividade (FIGUEIREDO, 2011).

No entanto a implementação de um sistema de monitoramento e controle pode ser inviável, quando considerando pequenos produtores. Neste contexto, verifica-se a possibilidade do desenvolvimento de uma plataforma de monitoramento e controle de fácil entendimento pelo usuário, sendo aplicável a diversos modelos de estufas, assim como para diferentes culturas.

1.3. JUSTIFICATIVA

Atualmente, uma palavra muito utilizada quando se fala de desenvolvimento tecnológico é sustentabilidade, ou seja, o uso consciente de recursos naturais. E quando se trata de cultivo em estufas, a qual desempenha um papel fundamental na produção agrícola, o monitoramento e controle do ambiente contribui para a otimização de recursos.

De acordo com Coutinho (2010), o produtor que não faz uso de tecnologias perde competitividade no mercado global, pois com a grande variação climática, ou clima inadequado à cultura proposta em determinadas regiões de cultivo, torna-se necessário o uso de certas tecnologias para se obter qualidade e produtividade. Pode-se considerar hoje que o uso de ambientes controlados é mais importante que se ter um solo fértil.

Dessa forma, uma estufa automatizada contribui para a produtividade e utilização racional de recursos, afinal alguns elementos atuadores podem ser ligados e desligados com a programação de *software*, os quais não sofrem intervenção humana, o que conduz a um menor número de falhas (LITJENS, 2009).

Para a elaboração da plataforma escolheu-se a extensão GUIDE, pertencente ao *software* MATLAB®, por apresentar uma interface amigável ao programador e ao usuário. Além disso, é possível a utilização do próprio MATLAB® para o processamento dos dados obtidos pelos sensores. Outro fator importante para a escolha do *software* foi a possibilidade

de geração de um executável, que pode ser usado em outros computadores, quando a plataforma estiver finalizada, sem a necessidade de ter o software MATLAB® instalado.

1.4. OBJETIVOS

A seguir serão apresentados os objetivos do trabalho.

1.4.1. Objetivo Geral

Realizar o estudo e desenvolvimento de uma plataforma através do ambiente gráfico do MATLAB®, GUIDE, para visualização de valores de luminosidade, temperatura e umidade de solo, com o intuito de aplicação em uma estufa.

1.4.2. Objetivos Específicos

A seguir foram definidos os objetivos específicos do trabalho.

- Pesquisar trabalhos e artigos relacionados ao tema proposto;
- Avaliar quais os melhores sensores a serem utilizados para a obtenção dos dados de luminosidade, temperatura e umidade de solo da estufa;
- Aprender a utilizar o GUIDE do MATLAB® para desenvolver a interface gráfica desejada;
- Estudar e desenvolver a aquisição do sinal, com o Arduino®;
- Estudar a comunicação entre o Arduino® e o MATLAB®;
- Realizar a leitura de luminosidade, temperatura e umidade de solo da estufa;
- Analisar a leitura obtida de luminosidade, temperatura e umidade de solo da estufa;
- Enviar sinais de acionamento para possíveis atuadores na estufa;

- Salvar os dados para possível analise e realizações de estatísticas pelos usuários;
- Caracterizar o comportamento das condições de luminosidade, temperatura e umidade de solo da estufa;
- Gerar um executável da plataforma, para que usuários sem o software MATLAB® possam utilizá-la.

2 METODOLOGIA

O presente trabalho foi desenvolvido em sete etapas, sendo que estas foram baseadas em pesquisas, simulações, desenvolvimento da plataforma, confecção do protótipo e testes. As pesquisas foram feitas através do levantamento bibliográfico referente ao tema. As simulações e desenvolvimento da plataforma foram feitos, a princípio, com os *softwares* PSIM® (*Software for Power Electronics Simulation*) e MATLAB®.

Já a confecção do protótipo e testes foram desenvolvidos com diversos equipamentos disponibilizados em laboratório do Departamento Acadêmico de Eletrônica – DAELN da Universidade Tecnológica Federal do Paraná – UTFPR, campus Campo Mourão, além de equipamentos próprios que são detalhados nas etapas de desenvolvimento e resultados.

Na Figura 1, ilustra-se o diagrama de funcionamento do projeto. Posteriormente são apresentadas as etapas de elaboração e confecção do projeto.

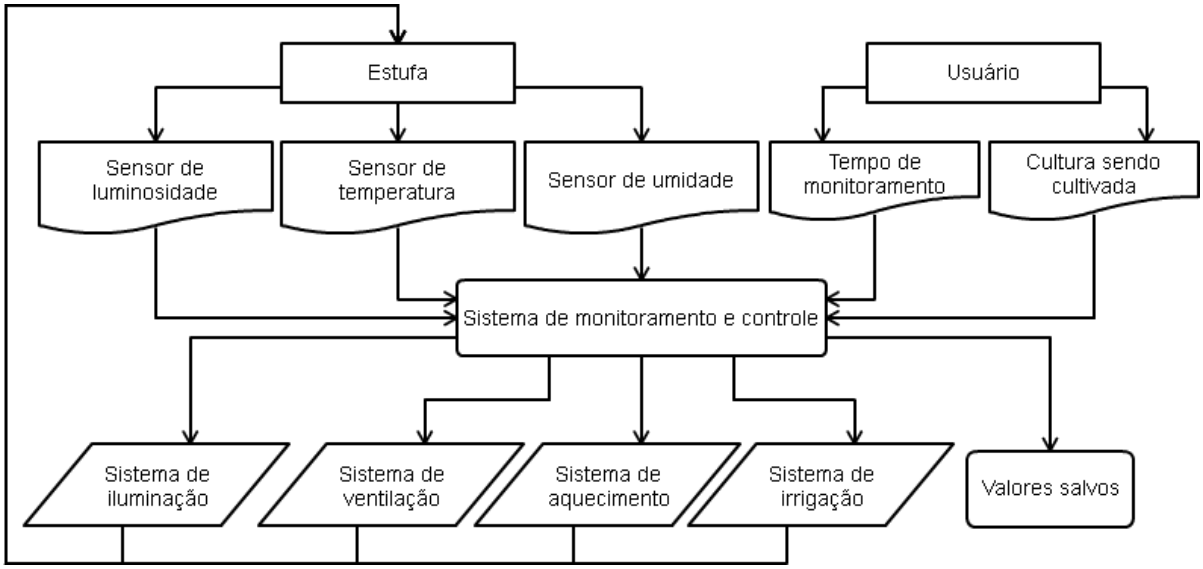

Figura 1: Diagrama de funcionamento do monitoramento e controle de uma estufa.

Fonte: Autoria própria.

A primeira etapa é composta pela revisão bibliográfica sobre o tema proposto, realizada a partir de fontes literárias, artigos e projetos referentes ao tema. Estudaram-se materiais relacionados, desde funcionamento de estufas e meios de controle, até as plataformas de monitoramento e controle já desenvolvidas.

Na segunda etapa foi feita a fundamentação do projeto, sendo dividida ainda em quatro fases, sendo elas:

1. Estudo dos sensores mais adequados para obtenção dos dados;
2. Estudo para elaboração do *hardware* de obtenção dos dados de luminosidade, temperatura e umidade de solo de uma estufa;
3. Elaboração da forma de comunicação entre o *hardware* de obtenção de dados, e o dispositivo de leitura dos mesmos, sendo esta feita através do Arduino®;
4. Elaboração do tratamento de dados e construção da plataforma de controle para o usuário;

Na terceira etapa o protótipo foi confeccionado, assim se obtendo o *hardware* para a aquisição das variáveis a serem controladas, e a plataforma para leitura e tratamento das mesmas.

Para a quarta etapa foi realizada a plataforma com o auxílio do *software* MATLAB®, sendo testada por meio de simulações, até chegar em uma interface otimizada e de fácil entendimento para o usuário.

Na quinta etapa foram realizados os testes, onde se verificou o funcionamento do projeto.

A sexta etapa constituiu-se da obtenção dos resultados finais, sendo analisados os dados e o desempenho da plataforma realizada. Para assim, partir para a sétima e última etapa que foi a conclusão, onde verificou-se que o projeto é viável e pode ser implantado em estufas.

3 FUNDAMENTAÇÃO TEÓRICA

No tópico a seguir foi abortada a teoria estudada para o desenvolvimento do trabalho, sendo essa baseada em autores conceituados em sua área de atuação.

3.1. ESTUFAS

Estufas são estruturas construídas com o objetivo de acumular calor em seu interior, e no caso, estufas de plantas são ambientes de cultivo controlado, assim propiciando um clima ideal para o cultivo de determinada espécie. Estas podem ser de diversos tamanhos e materiais, e são construídas de acordo com a necessidade do produtor. Um exemplo de estufa pode ser visualizado na Figura 2.

Figura 2: Modelo de estufa.
Fonte: (VAN DER HOEVEN, 2015).

Quando a fonte de calor é o sol o aquecimento dá-se porque a convecção é suprimida, e as estufas são feitas de materiais semitransparentes, não havendo troca de ar entre os meios interno e externo, assim mantém o calor que é absorvido e não há resfriamento causado por correntes de ar (GUEDES, 2009).

No entanto, alguns modelos de estufas têm na lateral plásticos que podem ser removidos dependendo das condições climáticas e necessidades do cultivo, assim a temperatura pode ser mantida mais elevada ou pode haver o resfriamento.

3.2. SENSORES

O termo sensor é aplicado a dispositivos que tem por finalidade serem sensíveis a alguma forma de energia em um sistema, tais como luminosidade, temperatura e umidade de solo. Através de sensores, pode-se fazer a leitura de determinadas características do ambiente e responder de acordo com elas, ou seja, criar um sistema capaz de interagir com o ambiente. No entanto um sensor nem sempre possui as características elétricas necessárias para ser utilizado em um sistema de monitoramento e controle, assim sendo necessário o uso de manipulações do sinal para possível leitura (PATSKO, 2006).

Segundo Thomazini e Albuquerque (2012), os sensores podem ser classificados como analógicos e digitais, sendo esta divisão feita de acordo com a resposta do sensor diante de variações em suas medidas. A diferença nas respostas pode ser vista na Figura 3.

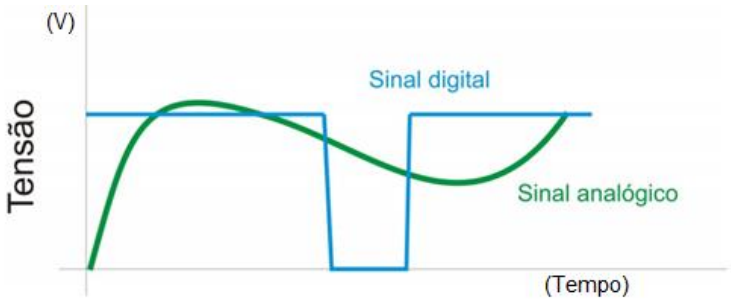

Figura 3: Respostas dos sensores analógicos e digitais.
Fonte: (PATSKO, 2006).

Os analógicos são mais antigos, portanto são mais comuns. Estes são assim designados por poderem assumir valores dentro de um intervalo previamente estabelecido. A sua utilização em circuitos analógicos é realizada sem problemas, entretanto quando é necessário monitorá-lo através de um circuito digital, como um computador, o sinal deve ser convertido em um sinal digital equivalente, os quais podem ser gravados e processados mais facilmente que os analógicos (THOMAZINI E ALBUQUERQUE, 2012).

Por outro lado, sensores digitais são baseados em níveis de tensão bem definidos, e podem ser descritos com alto ou baixo, ou simplesmente “1” e “0”, ou seja, seu funcionamento é baseado na lógica binária, dessa forma não sendo possível a existência de valores intermediários entre eles. Porém hoje em dia, os sensores digitais mais complexos podem alternar entre várias respostas distintas, e sua adaptação pode ser visualizada na Figura 4, a qual corresponde a uma discretização de um sinal analógico (PATSKO, 2006).

Figura 4: Sinal de saída de um sensor digital.
Fonte: (PATSKO, 2006).

Para este projeto foram utilizados sensores de luminosidade, temperatura e umidade de solo, que serão explicados posteriormente.

3.2.1. Sensor de Luminosidade

Atualmente um sensor utilizado em diversas aplicações é o sensor fotoelétrico, o qual tem por finalidade converter um sinal luminoso em um sinal elétrico que será processado por um circuito eletrônico assim pode realizar acionamentos de dispositivos como lâmpadas. Entretanto, para a correta utilização de sensores de luminosidade deve-se conhecer bem as suas características, para que esses sejam adequados ao tipo de projeto, e não prejudique seu funcionamento, afinal um sensor pouco sensível ou com resposta lenta pode arruinar um projeto.

O sensor pode ser um transdutor ou um sensor propriamente dito, os quais convertem luz em uma variação de uma grandeza como corrente ou resistência, sendo o caso de LDRs (*Light Dependent Resistor*) e fotodiodos, que serão brevemente exemplificados abaixo.

3.2.1.1. LDR

Os sensores LDRs também são conhecidos como foto-resistores, pois variam sua resistência de acordo com a intensidade luminosa incidente nele, sendo assim, quanto maior o diâmetro do sensor maior será sua capacidade de controlar correntes mais intensas que passam por ele (WENDLING, 2010).

A utilização mais comum deste é na iluminação pública, onde é utilizado para acionar ou desligar as lâmpadas de acordo com a luminosidade do ambiente, assim ocorrendo um processo automatizado, sem necessidade de alguém para realizar o controle.

Um modelo de LDR pode ser visto na Figura 5.

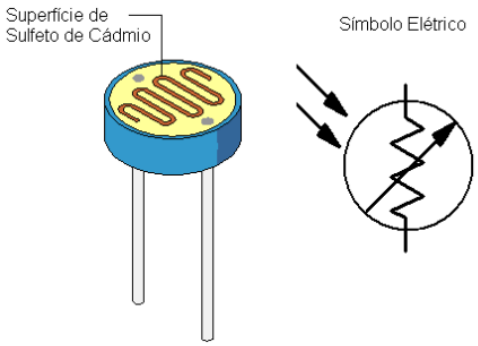

Figura 5: Exemplo de LDR.
Fonte: (WENDLING, 2010).

Como pode ser visto na Figura 18, os LDRs possuem uma camada de sulfeto de cádmio (CdS), que é um material semicondutor com a propriedade de diminuir sua resistência à passagem da corrente elétrica, quando há maior incidência de luminosidade (PATSKO, 2006).

A sua resistência é inversamente proporcional à luminosidade, dessa forma quanto mais luz no ambiente menor sua resistência, e conforme a intensidade luminosa diminui sua resistência aumenta, o que é exemplificado na Figura 6. O mesmo possui a vantagem de ser um sensor bidirecional, assim pode atuar em circuitos de corrente alternada (MENDES E STEVAN, 2013).

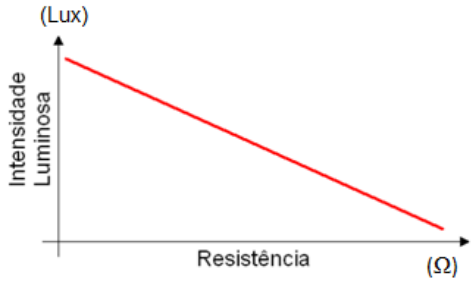

Figura 6: Gráfico da relação entre luminosidade e resistência.
Fonte: (WENDLING, 2010).

3.2.1.2. Fotodiodos

Os fotodiodos podem ser utilizados nos modos foto-condutivo ou foto-voltaico. No modo foto-condutivo o sensor aproveita a variação da resistência inversa com a luz no modo de operação,assim há uma variação de uma corrente que circula através do dispositivo, que pode ser visto na Figura 7.

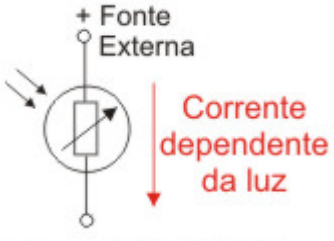

Figura 7: Foto-condutivo.
Fonte: (BRAGA, 2010).

Já no modo foto-voltaico é aproveitada a tensão gerada com a luz, ou seja, há a produção de uma corrente, o que pode ser visto na Figura 8.

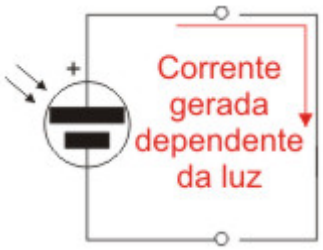

Figura 8: Foto-voltaico.
Fonte: (BRAGA, 2014).

O sensor tem um modo de funcionamento de fácil compreensão, ou seja, quando a luz incide numa junção semicondutora, portadores de carga são liberados. Sendo assim ambos os modos de funcionamento são muito utilizados (BRAGA, 2014).

3.2.2. Sensor de Temperatura

Medir a temperatura de um ambiente não é algo tão simples, devido a essa medida ser constituída de um sinal analógico que pode ser convertido para a forma digital, a sua precisão pode ser afetada. Dessa forma a escolha de um sensor para determinada aplicação envolve diversos fatores, que se mal avaliados levam a resultados inadequados em um projeto (BRAGA, 2014).

Os transdutores são conversores de grandezas, no caso, uma temperatura é convertida em sinal elétrico, e estes podem ser encontrados em três principais tipos, RTDs (*Resistance Temperature Detector*), Termistores e CIs (Circuito Integrado) Sensores.

3.2.2.1. RTD

São dispositivos que alteram a sua resistividade de acordo com a temperatura do ambiente, em sua maioria utilizam platina como material sensor, a qual apresenta medidas estáveis para temperaturas de até 500ºC. Apesar disso os tipos mais baratos e encontrados com maior facilidade no mercado são fabricados com níquel, como material sensor. A sua curva característica pode ser vista na Figura 9.

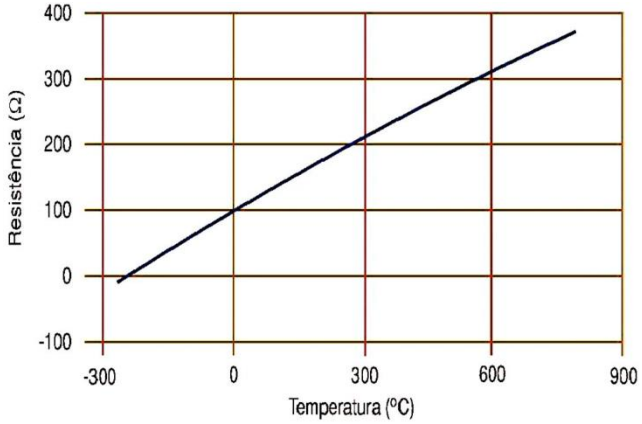

Figura 9: Curva característica RTD.

Fonte: (SILVA, 2006).

A maior desvantagem do sensor é o fato dele funcionar com uma corrente de medida, a qual pode provocar o auto aquecimento do sensor, o que pode dar uma falsa indicação da temperatura a ser medida.

3.2.2.2. Termistores

Como os RTDs, estes sensores têm a sua resistência dependente da temperatura. No entanto são fabricados com materiais cerâmicos semicondutores, e apresentam uma resistência elevada. Além disso, possuem um volume muito pequeno, com isso são mais precisos devido ao fato do auto aquecimento do sensor não afetar o ambiente cuja temperatura está sendo medida (BRAGA, 2014).

Este sensor possui a desvantagem da sua baixa linearidade, que exige algoritmos que façam a correção da leitura de temperatura.

A sua curva característica pode ser vista na Figura 10.

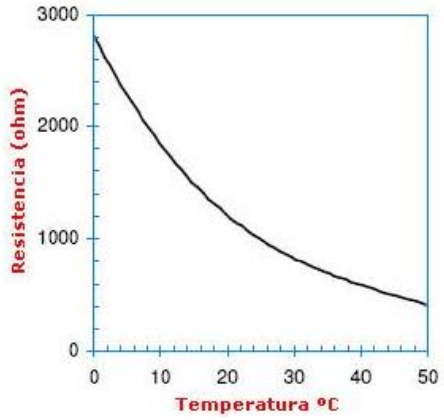

Figura 10: Curva caracerística termistor.

Fonte: (SILVA, 2006).

3.2.2.3. CIs Sensores

São sensores que já possuem recursos que permitem obter uma resposta linear com características que os circuitos normalmente utilizados podem operar. Entretanto, a faixa de temperatura que os circuitos podem operar é restrita, de acordo com o CI utilizado, e esses tipos de circuitos necessitam de uma fonte de alimentação externa, o que os torna sensíveis ao próprio aquecimento, o que pode interferir na leitura dos dados de temperatura do ambiente.

A curva característica de um CI sensor pode ser vista na Figura 11.

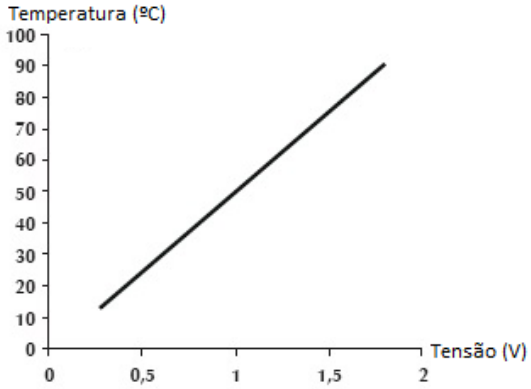

Figura 11: Curva característica CI sensor.

Fonte: (SILVA, 2006).

3.2.3. Sensor de Umidade

Sensores de umidade são sensores capazes de identificar a quantidade de água em um determinado objeto ou local, dentre as necessidades de medição de umidade, esta em se monitorar a umidade do solo em ambientes de cultivo. Isso, pois na agricultura se percebe a importância de medir a quantidade de água disponível para as plantas, e assim ser possível desenvolver sistemas automatizados de irrigação (PORTA, 2016).

3.3. LEITURA DOS SINAIS DOS SENSORES

A leitura dos sinais foi feita através da placa de desenvolvimento Arduino®, o qual se comunica com o computador pela serial, e os dados transmitidos foram lidos com o auxílio do *software* MATLAB®.

3.3.1. Arduino®

O projeto da placa de desenvolvimento Arduino® se iniciou em 2005 na cidade de Ivrea, Itália, e o mesmo pode ser visto na Figura 12, como uma alternativa ao *hardware* que era utilizado nas universidades da época, o qual possuía um custo elevado aos estudantes. Entretanto seu reconhecimento só veio em 2006 com o recebimento de uma menção honrosa na categoria Comunidades Digitais pela Prix Ars Electronica (THOMPSON, 2008).

Figura 12: Arduino® UNO
Fonte: (SOUZA, 2014).

A escolha do Arduino® UNO se deu pois com este é possível realizar a leitura de sensores e realizar o acionamento de atuadores, assim é possível a implementação do foco do projeto, que é o desenvolvimento da plataforma de monitoramento e controle de uma estufa com o *software* GUIDE do MATLAB®. Com o mesmo é possível enviar ou receber informações de praticamente todos os tipos de sistemas eletrônicos, sendo assim está presente em diversos campos de atuação, como por exemplo, impressão 3D, robótica e engenharia agronômica.

O Arduino® é uma plataforma física de computação, para prototipagem eletrônica, sendo uma placa microcontrolada com um ambiente de desenvolvimento para a escrita do código para a mesma. É utilizado o microcontrolador Atmel® AVR de 8 bits com suporte de entrada e saída, e sua linguagem de programação padrão tem origem em Wiring, e é essencialmente C/C++ (O'BRIEN, 2008).

De acordo com Grego (2009), seu objetivo é fornecer aos usuários a possibilidade da criação de ferramentas acessíveis, com baixo custo, flexíveis e fácil de serem usadas. Além de ser utilizada para projetos interativos independentes, pode ser conectada, através de uma interface serial ou USB (Universal Serial Bus), a um computador hospedeiro, o qual pode receber e enviar dados para o Arduino®. No entanto a plataforma não possui recurso de rede, porém é possível o uso de extensões e a combinação de mais placas Arduino®, assim sendo possível a comunicação entre eles.

3.3.2. Comunicação serial

Comunicação serial é o processo de enviar dados em um canal de comunicação ou barramento, sendo que esse envio é feito de um bit de cada vez e de forma sequencial, sendo usada em toda a comunicação de longo alcance e na maioria das redes de computadores. Atualmente possui grande importância na área de automação, pois grande parte dos equipamentos utilizam o padrão RS-232 de comunicação, que é a conexão encontrada em computadores compatíveis com a IBM® (*International Business Machines*), sendo assim possui diversas aplicações e o uso mais frequente para estudantes é através de microcontroladores. (CUGNASCA e HIRAKAWA , 2006).

A comunicação pode ser feita de forma síncrona, onde o tempo de transmissão e recebimento é bem definido e conhecido pelo transmissor e receptor, ou seja, estes devem estar sincronizados, e para manter a sincronia entre eles um bloco de informações é transmitido periodicamente. E assíncrona, onde cada bloco de dados possui uma *flag*, para que saiba exatamente onde começa e termina o bloco de dados e qual a sua posição na sequência de informação transmitida.

Uma das características da comunicação assíncrona é o envio sequencial dos bits, seguindo a sequência: start bit, dados, bit de paridade e stop bit, como mostrado na Figura 13. Os dados, geralmente, são de 8 bits com codificação no padrão ASCII (*American Standard Code for Information Interchange*), o bit de paridade é um método simples de se verificar se houve algum erro na transmissão do pacote de dados, dessa forma auxiliam a comunicação serial em busca de erros, pois se o bit de paridade não estiver de acordo o dado é considerado inconsistente (MARTINO, 2004).

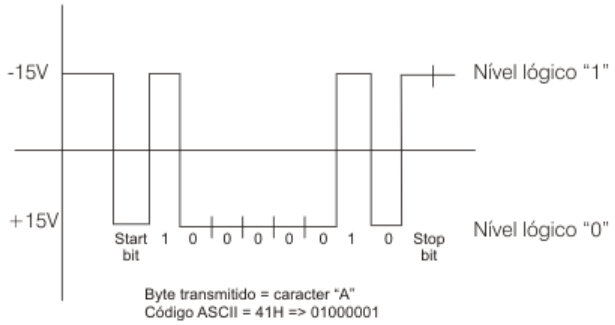

Figura 13: Exemplo de comunicação serial.

Fonte: (SOARES, 2014).

A velocidade da comunicação serial é dada por BPS (*bits per second*) ou *baud rate*, e isso informa a quantidade de bits que podem ser transmitidos por segundo. Os dispositivos que forem se comunicar devem estar configurados exatamente iguais, ou seja, com a mesma velocidade, paridade e configuração de stop bit, só assim será possível a comunicação.

O Arduino® utiliza para a comunicação serial o UART (*Universal Asynchronous Receiver Transmitter*), no qual é possível a comunicação nos dois sentidos, assim sendo Full Duplex, ou seja, permite que cada nó da rede envie e receba dados simultaneamente (TANENBAUM, 2003).

3.3.3. MATLAB®

O MATLAB® é um *software* que foi desenvolvido no fim dos anos 70, por Cleve Moler na Universidade do Novo México, mas logo se expandiu para as demais universidades, sendo amplamente utilizado no âmbito da comunidade da matemática aplicada, o mesmo foi adotado pela primeira vez por engenheiros de projeto de controle.

O *software*, de alto desempenho, é destinado a fazer cálculos com matrizes. Assim seus comandos são próximos a maneira que escrevemos expressões algébricas, assim tornando mais simples seu uso. Todavia, atualmente o MATLAB® é definido como um sistema interativo e uma linguagem de programação para computação técnica e científica em geral, integrando a capacidade de fazer cálculos, visualização gráfica e programação (TONINI e COUTO, 1999).

Entre os usos típicos do MATLAB® estão desenvolvimento de algoritmos; modelagem, simulação e confecção de protótipos; análise, simulação e confecção de dados; gráficos científicos e de engenharia; desenvolvimento de aplicações, inclusive a elaboração de interfaces gráficas com o usuário. Tendo estas aplicações em vista o *software* se torna o mais indicado para a realização do projeto, pois através dele é possível realizar o monitoramento e controle da estufa, sendo assim possível realizar a leitura dos dados dos sensores e análises dos mesmos.

3.4. INTERFACE GRÁFICA

Para o este projeto o desenvolvimento da interface gráfica foi realizado com o auxílio do GUIDE, o qual, conforme dito anteriormente, possui uma interface amigável e de uso simples. A ferramenta GUIDE é integrante do *software* MATLAB®, pode ser visualizada na Figura 14, sendo assim seu GUI (*Graphical User Interface*), ou sua interface gráfica para o usuário, que consiste em uma janela do tipo figura contendo *menus*, textos, botões, entre outros componentes visuais, os quais podem ser manipulados pelo usuário.

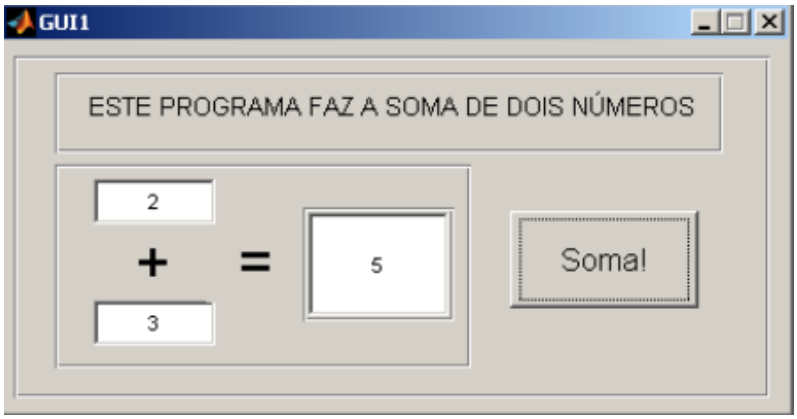

Figura 14: Exemplo de interface gráfica GUI no MATLAB®.

Fonte: (BEZERRA, et al., 2014).

Interface gráfica é um tipo de interface que permite a interação entre o usuário e o dispositivo digital. Foi desenvolvida nos anos 60, no entanto, aperfeiçoada em 1970 por pesquisadores da Xerox PARC®, os quais foram além da interface de texto, ou seja, utilizaram a interface gráfica como principal interface do computador Xerox Alto, e isto influenciou a maior parte dos ambientes gráficos em sistemas operacionais conhecidos atualmente.

Entretanto, apenas em 1980, com Steve Jobs e Jef Raskin, o processo de desenvolvimento e modernização da interface gráfica teve continuidade, e assim o primeiro produto de sucesso que utiliza interface gráfica foi lançado, o Macintosh, o qual utilizava uma metáfora onde os arquivos se pareciam com folhas de papel e os diretórios, pastas de arquivos.

Já em 1990, com a popularização do uso de computadores pessoais, houve a criação de um mercado, que poderia explorar a oportunidade de interfaces de uso fácil, além do

desenvolvimento de tecnologias que proporcionaram o aparecimento de sistemas mais sofisticados.

Uma interface gráfica combina tecnologias e dispositivos, fornecendo uma plataforma de interação, essa combinação é conhecida como WIMP (*window, icon, menu, pointing device*), que é composta por janelas, ícones, *menus* e ponteiros. Os comandos são compilados através de *menus* que são acionados pelos ponteiros, além de existir um gerenciamento para janelas e aplicações.

4 DESENVOLVIMENTO DA PLATAFORMA

Neste capítulo foi explicado o desenvolvimento do presente trabalho, sendo dividido cada parte em um tópico para melhor entendimento dos procedimentos realizados.

4.1 AMBIENTE DE CRIAÇÃO DE INTERFACE GRÁFICA DO MATLAB®

Para dar início ao projeto é necessário conhecer a interface gráfica GUIDE fornecida pelo *software* MATLAB®. Primeiramente deve-se digitar na janela de comando do próprio MATLAB® o comando “guide”, e então uma nova janela se abrirá, a qual é vista na Figura 15.

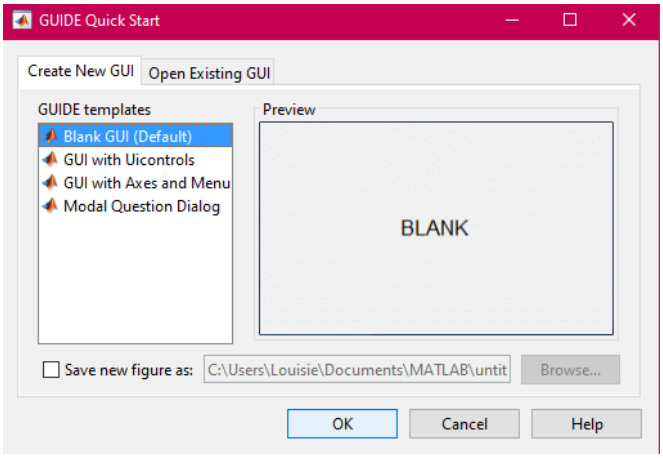

Figura 15: Iniciando o GUIDE.

Fonte: Autoria própria.

Logo selecionou-se um projeto em branco, como pode ser visto na Figura 16, para dar início a construção da interface gráfica desejada.

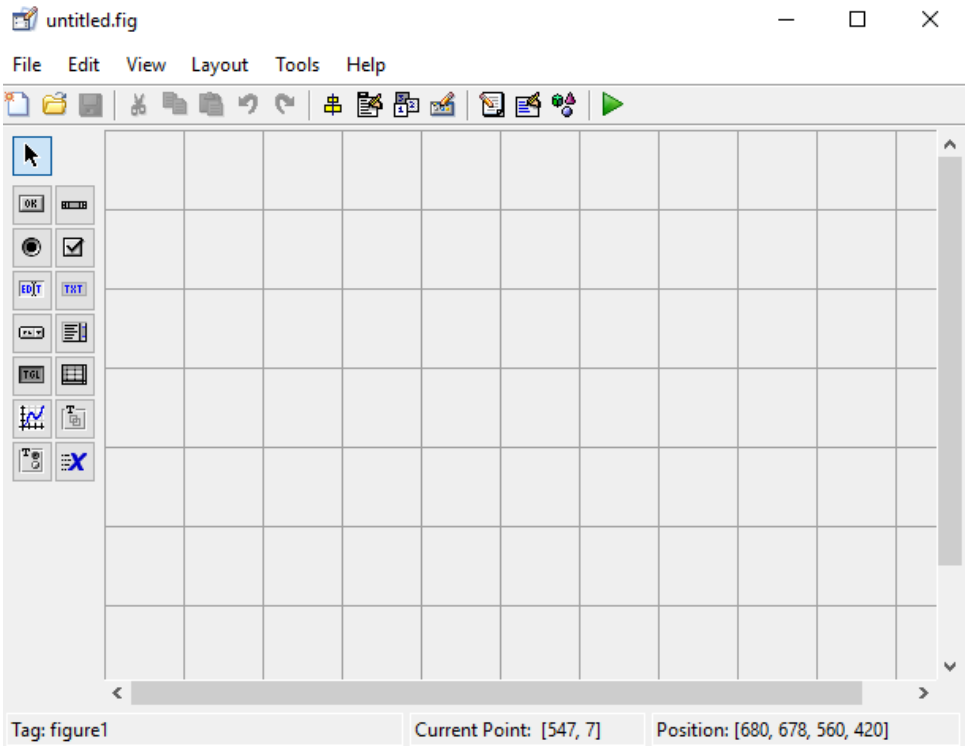

Figura 16: Janela em branco do guide

Fonte: Autoria própria.

A interface disponibiliza elementos como janelas gráficas, botões, *menus*, painéis e editores de texto. A montagem da interface gráfica utilizada é melhor explicada no decorrer do trabalho, no entanto a montagem é simples, basta clicar e arrastar os itens ao local

desejado, e se necessário podem ser feitas alterações nas configurações dos itens para melhor identificação no momento de elaboração do código, para isso basta clicar duas vezes sobre o item desejado e alterar o que se deseja no *menu* que irá se abrir, o qual pode ser visto na Figura 17.

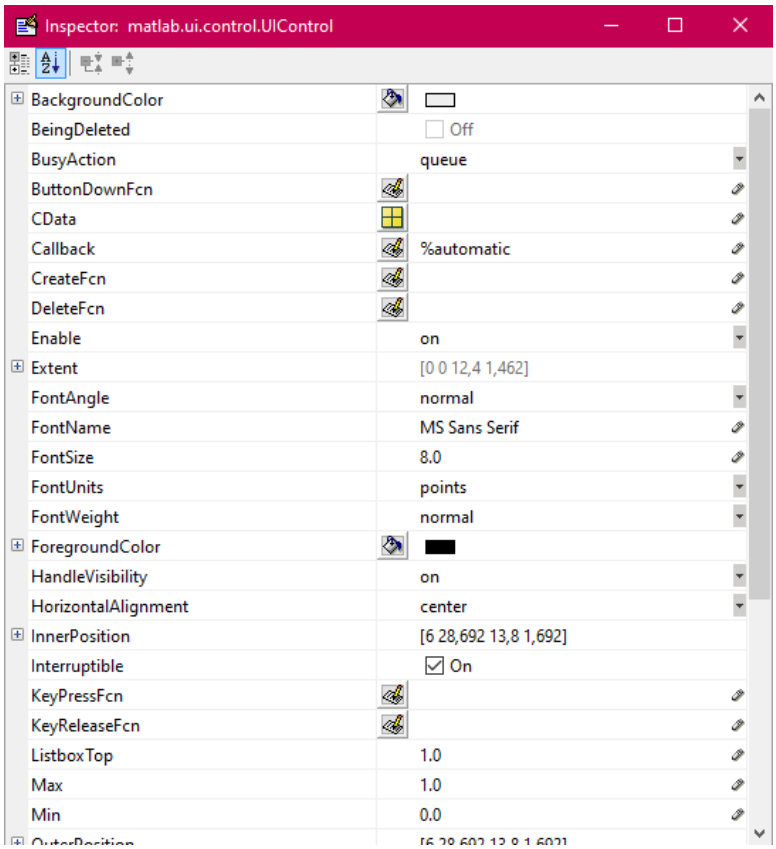

Figura 17: *Menu* de configurações de um item do GUIDE.

Fonte: Autoria própria.

O código é gerado automaticamente quando a interface é salva e então pode ser editado para receber funções específicas.

4.2 INTERFACE GRÁFICA

A primeira etapa do desenvolvimento da plataforma de monitoramento e controle de um ambiente de cultivo controlado consiste na elaboração da interface gráfica a ser apresentada para o usuário. Esta foi desenvolvida com o uso da ferramenta GUIDE do *software* MATLAB®, que é um programa que permite a análise e controle dos dados obtidos

pelas leituras dos sensores, a plotagem de gráficos em tempo real, o salvamento tanto das imagens gráficas quanto de seus pontos de medições, além de, por meio de bibliotecas, realizar a comunicação com o Arduino®.

A interface gráfica construída pode ser visualizada na Figura 18.

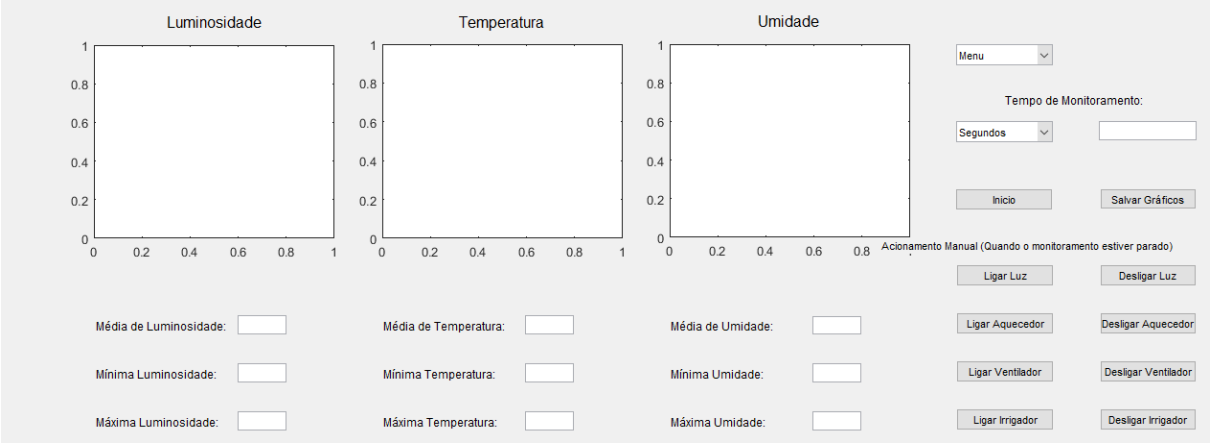

Figura 18: Interface gráfica construída.

Fonte: Autoria própria.

A mesma é composta por três janelas, mostradas na Figura 19, onde os gráficos de luminosidade, temperatura e umidade são exibidos em tempo real, dessa forma o eixo x é alterado conforme o tempo de monitoramento estipulado pelo usuário.

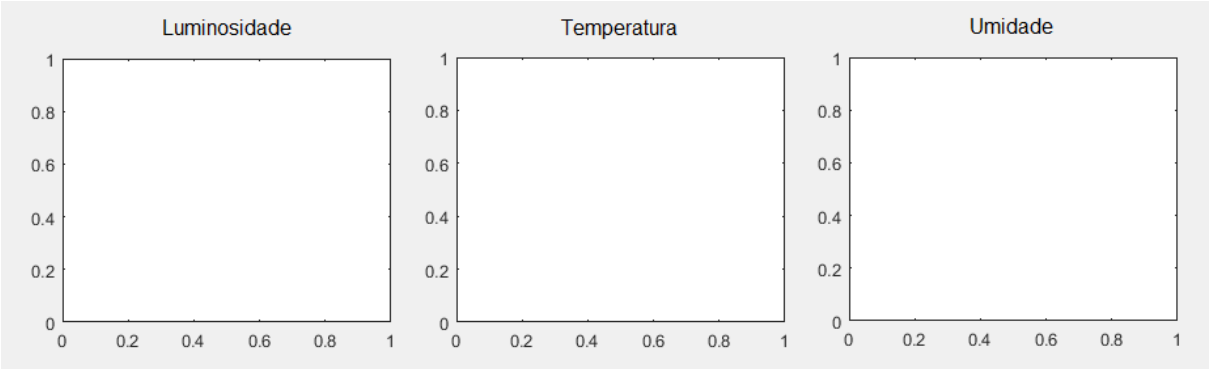

Figura 19: Janelas gráficas utilizadas na interface gráfica.

Fonte: Autoria própria.

Um *menu* onde o usuário deve selecionar o que está sendo cultivado no momento, demonstrado genericamente pela Figura 20.

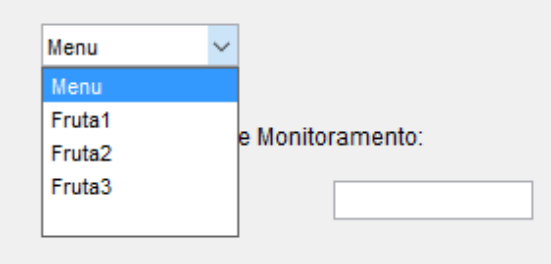

Figura 20: *Menu* de opções de cultivo.

Fonte: Autoria própria.

E também um campo de texto editável onde o usuário deve inserir o tempo que deseja monitorar o ambiente de cultivo, que é visualizado através da Figura 21.

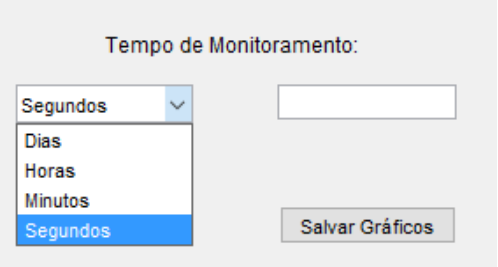

Figura 21: *Menu* com opções de tempo de monitoramento.

Fonte: Autoria própria.

Além disso, a interface conta com botão de iniciar, para que o monitoramento seja realizado. Botão de salvar gráficos para que as imagens gráficas sejam salvas onde o usuário desejar. E botões de ligar e desligar para os atuadores da estufa, sendo esses uma luz, um aquecedor, um ventilador e um irrigador. Todos os botões são mostrados na Figura 22.

Figura 22: Botões utilizados na interface gráfica.

Fonte: Autoria própria.

Por fim, a interface apresenta campos de texto, onde ao final do monitoramento, são exibidos dados como média dos valores lidos no período de monitoramento e valores de máximos e mínimos atingidos no mesmo espaço de tempo, o que pode ser visto na Figura 23.

Figura 23: Painel de exibição de valores médios, mínimos e máximos durante o monitoramento.

Fonte: Autoria própria.

4.3 ALIMENTAÇÃO DOS COMPONENTES

Inicialmente, foi projetada a forma de alimentação dos sensores e do Arduino®, pensou-se que em um ambiente de cultivo controlado deve ser utilizado o menor número de equipamentos possíveis, para que os mesmos não ocupem muito espaço e não interfiram na produção local.

Dessa forma, o Arduino® é alimentado pela própria porta USB que está conectada ao computador, e este gera sinais de 5[V] e GND (*Graduated Neutral Density Filter*), o que pode ser verificado na Figura 24, a qual mostra a placa de desenvolvimento Arduino® UNO e suas saídas utilizadas para alimentação dos sensores e dos leds sinalizadores dos atuadores dentro da estufa.

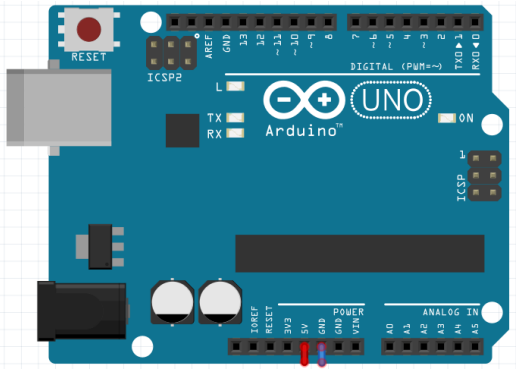

Figura 24: Saídas de tensão proporcionadas pelo Arduino®.

Fonte: Autoria própria.

4.4 SENSORIAMENTO

4.4.1 Sensor de Luminosidade

O sensor de luminosidade que foi utilizado foi o LDR de 10mm, o qual tem a variação de sua resistência dependendo da variação de luminosidade incidente nele. O sensor de luminosidade LDR não possui polarização, assim podendo ser ligados de ambos os lados no sistema.

A escolha do tamanho do LDR se deu após a realização de testes com o LDR de 5mm e o LDR de 10mm, sendo que o segundo apresentou uma maior sensibilidade à variação de luminosidade.

Para fazer a medida de luminosidade no ambiente é necessário converter a variação da resistência em uma variação de tensão. Isso pode ser realizado através de um esquema simples de um divisor de tensão, mostrado na Figura 25, onde o LDR é colocado no lugar de

um dos resistores, e desta maneira a tensão de saída irá aumentar conforme a luminosidade aumenta.

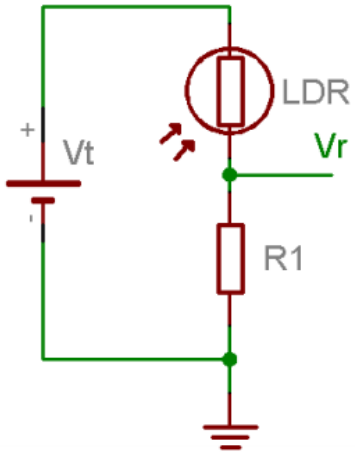

Figura 25: Circuito para medir a tensão proporcionada pela leitura do LDR.
Fonte: (PATSKO, 2006).

A vantagem do uso do LDR é que este pode trabalhar com correntes elevadas, sendo muito sensíveis, o que simplifica o projeto de seu circuito. Porém uma desvantagem é o tempo de resposta, sendo sensores lentos que não operam acima de alguns quilo hertz. (WENDLING, 2010).

4.4.2 Sensor de Temperatura

O sensor de temperatura escolhido para o projeto foi o LM35, visto na Figura 26, que é um sensor de precisão, fabricado pela National Semiconductor®, tendo uma saída analógica e com faixa de medição de -55ºC à +150ºC.

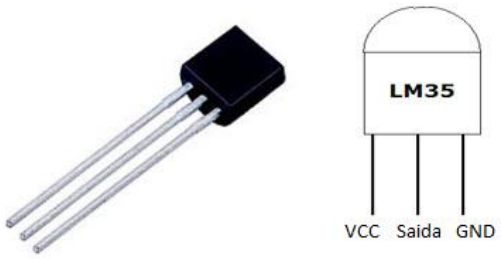

Figura 26: Lm35.

Fonte: Autoria própria.

A escolha do sensor foi realizada devido por realizar uma leitura analógica de temperatura, e poder ser alimentado com 5[V], provenientes do próprio Arduino®. Além disso, é um sensor de baixo custo, tornando-se viável para a execução do projeto.

Para obter o valor de temperatura através da variação de tensão é simples, pois cada 10[mV] na saída representa 1ºC, como pode ser visto na Figura 27.

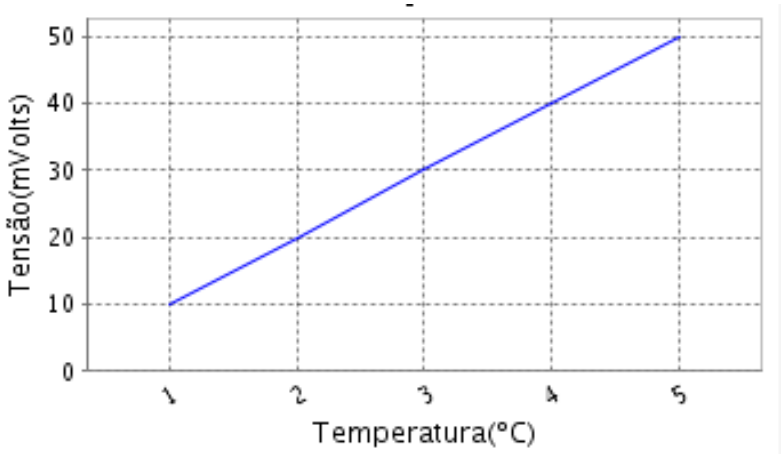

Figura 27: Curva de temperatura x tensão.

Fonte: Autoria própria.

4.4.3 Sensor de Umidade de Solo

O sensor utilizado foi o P23 da GBK Robotics®, que pode ser visto na Figura 28, o qual, atuando em conjunto com placas microcontroladas é capaz de medir a umidade do solo em determinado local, é uma tecnologia de baixo custo e consumo energético. Com o auxilio de microcontroladores é possível que em determinado índice acione-se um alarme sonoro, luminoso, ou até mesmo um equipamento para irrigação do solo.

Figura 28: Sensor de umidade de solo P23.

Fonte: Autoria própria.

A escolha do mesmo se deu por ser de fácil utilização, podendo ser aplicado por estudantes ou profissionais das áreas tecnológicas.

O P23 é composto por um sensor, que através de duas placas metálicas vistas na Figura 29, realiza a medição de umidade auferindo a corrente que percorre as mesmas, sendo assim um sensor compacto e muito aplicado em projetos eletrônicos de automação residencial.

Figura 29: Sensor de umidade de solo P23, verso.

Fonte: Autoria própria.

4.4.4 Montagem do Esquemático dos Sensores

Após pesquisas e definições de quais sensores seriam utilizados para a montagem do protótipo realizou-se o esquemático de alimentação e envio de dados dos sensores, que pode ser visualizado na Figura 30, sendo esses o sensor de luminosidade LDR de 10mm, o sensor de temperatura LM35 e o sensor de umidade P23.

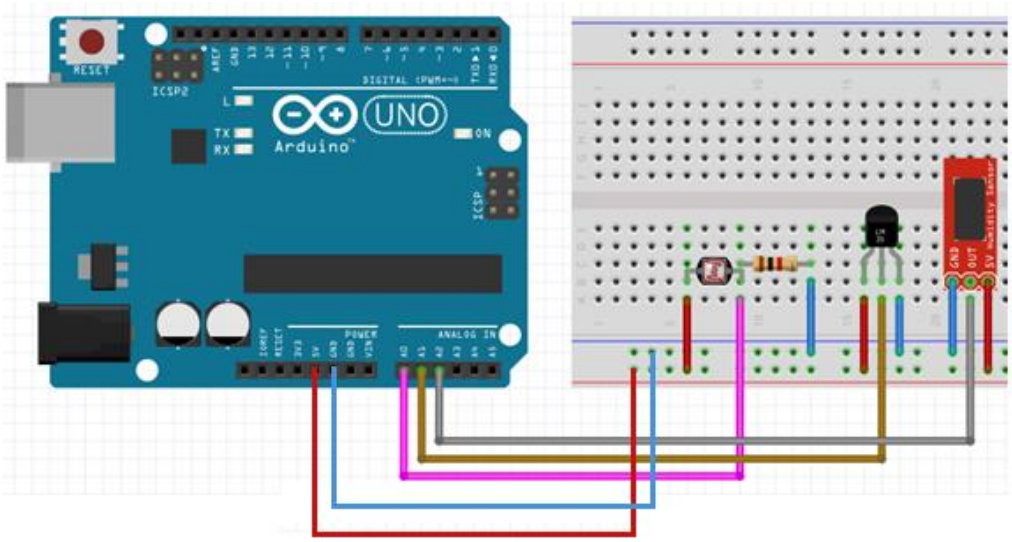

Figura 30: Esquemático de ligação dos sensores nos pinos do Arduino®.

Fonte: Autoria própria.

Como pode ser visto, uma extremidade do sensor LDR deve ser conectado à alimentação de 5[V] e a outra à extremidade do resistor de 1KΩ, que forma assim um divisor de tensão para que possa ser medida a variação da resistência do LDR, e à entrada A0 do Arduino®, onde realiza-se a leitura do sinal, logo a outra extremidade do resistor deve ser ligado ao GND. O resistor de 1KΩ foi determinado por praticidade nos cálculos posteriores.

Para o sensor de temperatura deve-se ligar o pino VCC na alimentação de 5[V], o pino Saida à entrada A1 do Arduino®, para a realização da leitura do sinal do sensor, e o pino GND deve ser ligado ao GND.

Por fim, o sensor de umidade de solo, para esse sensor deve-se sempre verificar a posição de seus pinos, pois estes podem mudar de acordo com o fabricante, no caso do sensor utilizado conecta-se o pino 1 ao GND, o pino 2 à entrada A2 do Arduino® e o pino 3 à alimentação de 5[V].

O esquemático elétrico pode ser visto na Figura 31.

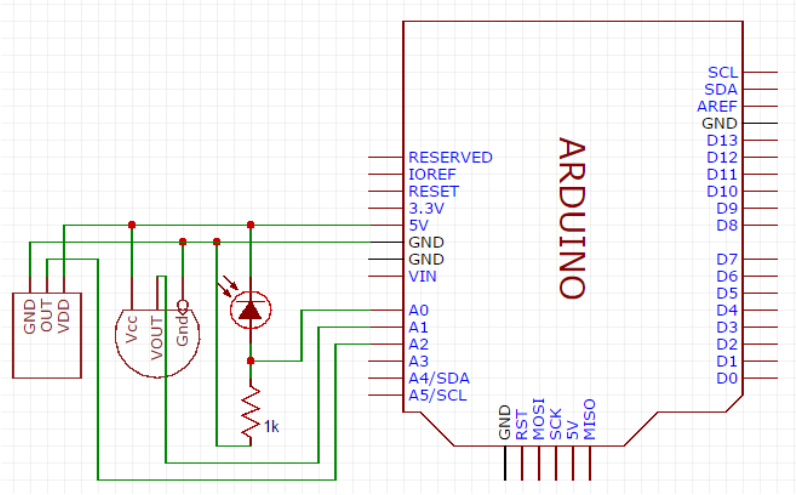

Figura 31: Esquemático elétrico sensores.

Fonte: Autoria própria.

4.5 LEITURA DOS SINAIS MONITORADOS

Depois de conectados os sensores, podem ser realizadas as leituras dos sinais provenientes de cada um, com o auxílio do *software* desenvolvido e a interface gráfica elaborada no ambiente de desenvolvimento do MATLAB®.

Para isso deve-se verificar se o programa de comunicação está gravado no Arduino® UNO, caso não esteja gravado é necessário que se grave com o código que se encontra no Apêndice A. Após isso se deve iniciar o programa executável gerado, e escolher as ações a serem realizadas.

A leitura é realizada através do comando analogRead, que é reconhecido pelo MATLAB®, o mesmo realiza a leitura das portas analógicas do Arduino®, assim obtém os valores dos sensores.

4.5.1 Luminosidade

O sensor de luminosidade tem como saída um valor analógico que varia de 0 à 1023, correspondentes à 0[V] e 5[V].O valor lido é uma variação de tensão proveniente do divisor

de tensão entre o LDR e o resistor de 1KΩ e a essa variação é vista na equação 1, onde V_{A1}corresponde ao valor lido na entrada analógica do Arduino®.

$$V_{Luz} = V_{A1} * \frac{5}{1023} \quad (1)$$

A equação da corrente que circula pelo circuito pode ser vista na equação 2, com o uso de um resistor de 1KΩ.

$$C = \frac{V_O}{1K} \quad (2)$$

Se o LDR é alimentado com 5[V], a resistência de saída proveniente do LDR pode ser expressada pela equação 3.

$$R_{LDR} = \frac{5 - V_O}{C} \quad (3)$$

Logo se obtém a equação 4 que é a relação entre a tensão e a medida da intensidade luminosa em Lux. De acordo com Mendes e Stevan (2013), a mesma é uma equação padrão.

$$Lux = 255{,}84 * R_{LDR}^{\frac{-10}{9}} \quad (4)$$

4.5.2 Temperatura

Como no sensor de luminosidade, o sensor de temperatura tem como saída um valor analógico que varia de 0 à 1023, baseado nisso pode-se determinar o valor de tensão na entrada analógica do Arduino® através da equação 5, onde V_{A2}corresponde ao valor lido na entrada analógica do Arduino®.

$$V_{temp} = V_{A2} * \frac{5}{1023} \quad (5)$$

Sabe-se que cada 10[mV] de variação corresponde a 1ºC, segundo o *datasheet* do fornecedor, o valor de temperatura é ajustado com o auxílio do cálculo presente na equação 6, para que a leitura do sinal seja representada em graus Celsius.

$$Temperatura = \frac{V_{temp}}{0.01} \quad (6)$$

No projeto que está sendo aplicado, o sensor deve compreender uma faixa de -15ºC à 55ºC, que são valores mínimo e máximo de temperaturas que compreende as temperaturas atingidas em nosso país.

4.5.3 Umidade do Solo

O sensor de umidade de solo tem um comportamento semelhante aos sensores de luminosidade e temperatura, pois o sinal da sua leitura analógica varia de 0 à 1023. Logo é necessário que se ajuste o valor da tensão de leitura, para isso utiliza-se a equação 7.

$$V_{solo} = V_{A3} * \frac{5}{1023} \quad (7)$$

Posteriormente, deve-se utilizar a equação 8 para que a umidade seja exibida em porcentagem, e neste caso a cada 19,452[mV] de alteração, de acordo com o *datasheet* do fornecedor, é 1% que altera-se na medida da umidade do solo.

$$Umidade = \frac{V_{solo}}{0.019452} \quad (8)$$

4.6 INDICADORES

A plataforma conta com indicadores de funcionamento, os quais facilitam a identificação do que está acontece no processo, e quais atuadores estão em funcionamento, os mesmos são indicadores visuais e um indicador sonoro.

4.6.1 Indicadores visuais

Os indicadores visuais utilizados são *LEDs (Light Emitter Diode)*, o qual é um componente eletrônico semicondutor que faz com que energia elétrica se transforme em luz, desde que seja polarizado corretamente, permitem assim a passagem de corrente e consequente emissão de luz.

Para o projeto foram utilizados quatro *LEDs*, sendo esses de cores diferentes para poder representar cada sinal de saída do Arduino®, e também cada atuador presente na estufa. As cores utilizadas foram amarelo, vermelho, azul e verde, que representam respectivamente os acionamentos de luz, aquecedor, ventilador e irrigador, que são os atuadores propostos para utilização em uma estufa.

4.6.2 Indicador sonoro

Como indicador sonoro foi utilizado um *buzzer*, que é um dispositivo eletrônico composto por duas camadas de metal e uma terceira camada interna de cristal piezoelétrico, que ao receber energia emite um sinal sonoro.

No projeto o *buzzer* está presente para indicar a conexão do Arduíno® com o *software*, através de três breves apitos, e também indicar o final do processo de monitoramento através de um apito mais longo.

4.6.3 Montagem do esquemático dos indicadores

Na Figura 32, pode ser vista a montagem dos indicadores, sendo que todos são componentes com polaridade, sendo assim os catodos devem ser ligados no GND e os anodos na saída correspondente do Arduino®. Pode-se observar que os anodos dos LEDs são ligados à resistores de 1KΩ, para que seu brilho seja menos intenso, pois foram utilizados LEDs de alto brilho.

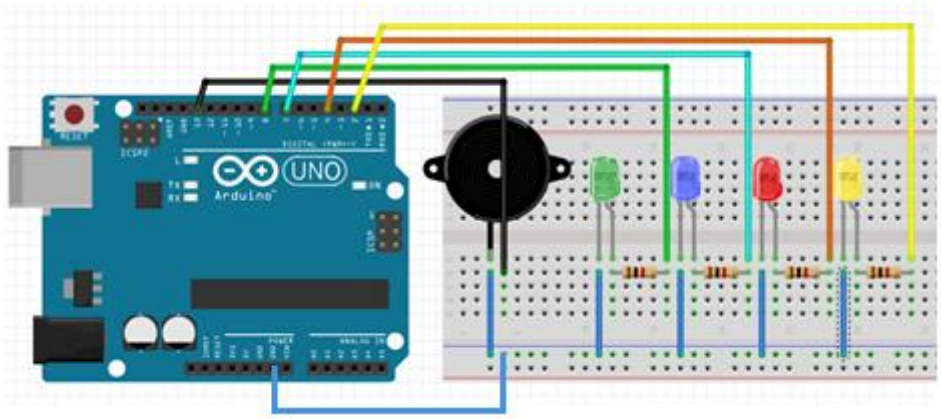

Figura 32: Esquemático de ligação dos indicadores nos pinos do Arduino®.

Fonte: Autoria própria.

O esquemático elétrico realizado pode ser visto na Figura 33.

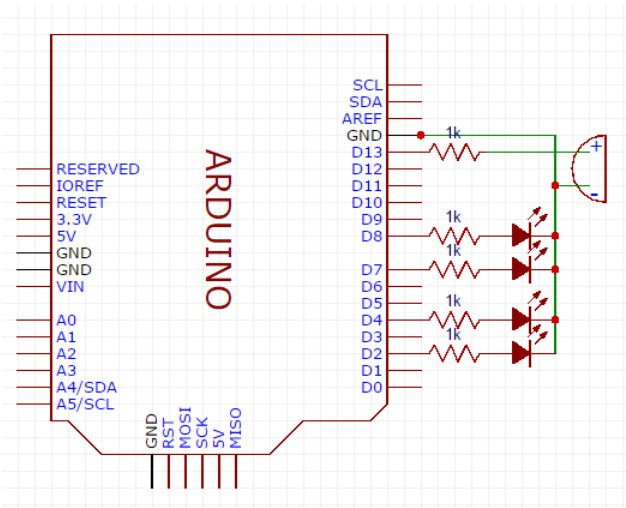

Figura 33: Esquemático elétrico dos indicadores.

Fonte: Autoria própria.

4.7 DESENVOLVIMENTO DO PROGRAMA

Para o desenvolvimento do *software* foi utilizado o MATLAB®, o qual permite a realização de scripts, onde o código fica salvo. Posteriormente pode-se gerar um executável para usuários que não possuem a licença para o mesmo programa, para que o *software* não seja modificado por usuários, e para que não seja necessário que se compile o código cada vez que se deseja executar o *software*.

O funcionamento geral do *software* pode ser visto na Figura 34, e o código elaborado pode ser visto no Apêndice B.

Ao criar a interface gráfica um código genérico já é criado, o qual tem funções de inicialização pré-definidas e as chamadas das funções de cada item utilizado. Com a base do código pronta, foi necessário programar as funções para o correto funcionamento da plataforma.

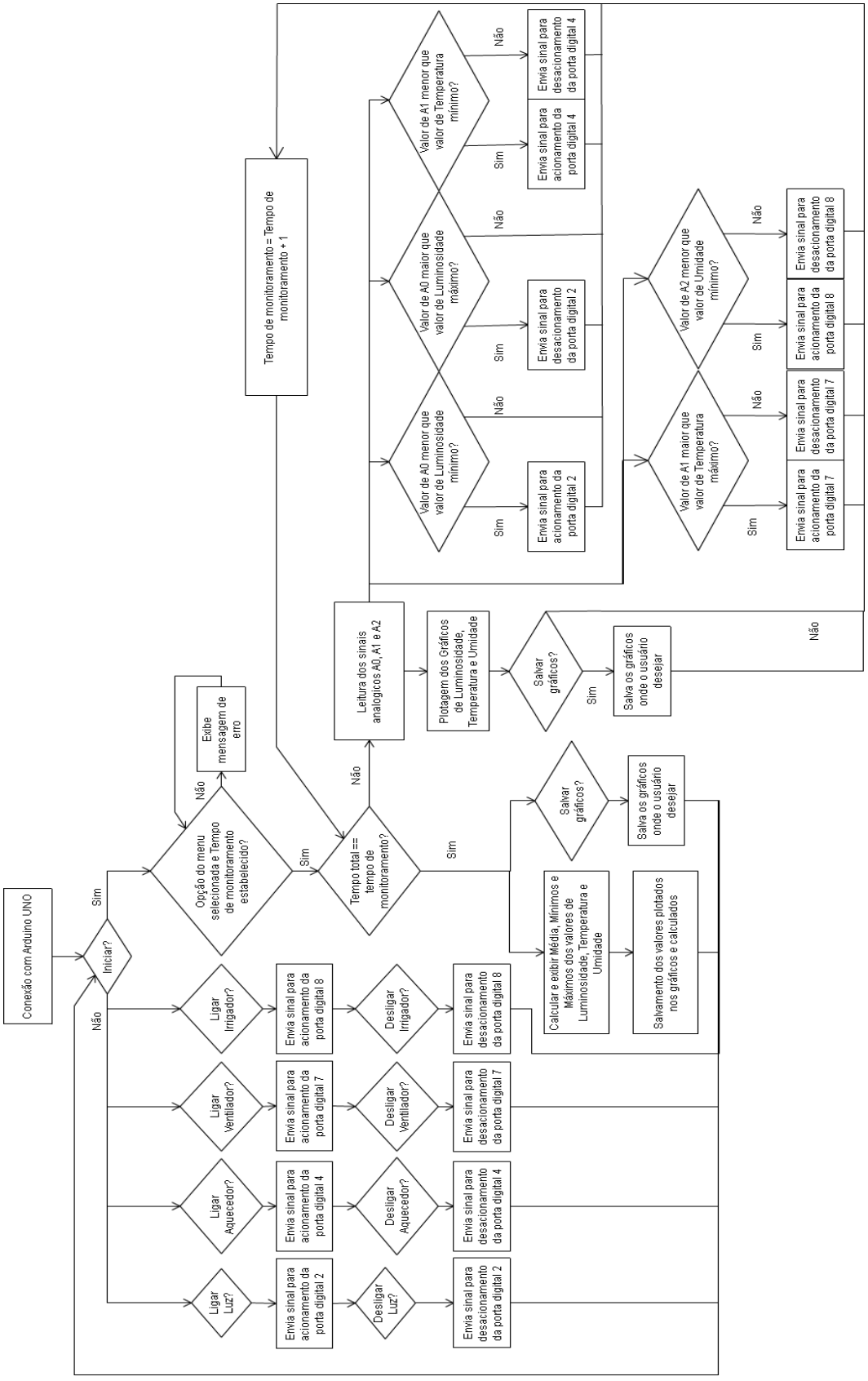

Figura 34: Fluxograma de funcionamento da plataforma.

Fonte: Autoria própria.

4.7.1 *Menu*

A primeira função a ser elaborada é o *menu* que contém opções de espécies para cultivo, no caso foi colocado de forma generalizada, sendo as opções representadas por “Fruta1”, “Fruta2” e “Fruta3”.

A função tem por objetivo verificar qual opção foi selecionada pelo usuário, e grava a mesma em uma variável que será utilizada posteriormente na função início. A verificação é feita de modo simples, primeiramente armazena-se todas as opções do *menu* como *string*, logo armazena-se qual o valor numérico da opção escolhida. Em seguida é armazenado em uma variável a *string* específica da posição escolhida, e essa variável pode ser utilizada em demais funções.

4.7.2 Tempo de Monitoramento

Primeiramente a função de tempo de monitoramento verifica a opção escolhida no *pop-up menu* que tem as opções de tempo de monitoramento, em “Dias”, “Horas”, “Minutos” e “Segundos”, e a armazena em uma variável qual a escolha do usuário.

Posteriormente a função realiza a leitura do campo editável e armazena a *string* lida, em seguida é feita a conversão de *string* para *double*. O parâmetro fornecido pelo usuário deve ser um número que posteriormente será utilizado para determinar o tempo que a plataforma irá realizar a leitura dos sensores, a variável onde o valor em formato *double* é armazenado será utilizado pela função inicio.

Nessa mesma função, após verificar a opção no *menu* de tempo e armazenar o valor digitado pelo usuário, são feitas multiplicações para se obter o tempo em minutos, horas e dias, sendo que para a opção de segundos não é necessária nenhuma modificação. Então o novo valor obtido será enviado para a função inicio.

4.7.3 Inicio

Essa função é a mais importante no desenvolvimento do *software*, pois ela é responsável pelo correto funcionamento da plataforma. Ou seja, a função faz a verificação dos parâmetros passados pelo usuário, realiza a leitura dos sensores, plota os gráficos, armazena os dados necessários para os cálculos e realiza o controle para o acionamento automático dos atuadores presentes na estufa.

4.7.3.1 Verificação de parâmetros

Para que isso tudo seja possível, primeiramente a função recebe as variáveis armazenadas pelas funções "*Menu*" e "Tempo de Monitoramento", verifica se a opção do *menu* é válida e se o usuário estabeleceu um tempo para que ocorra o monitoramento, o que pode ser visualizado no fluxograma da Figura 35.

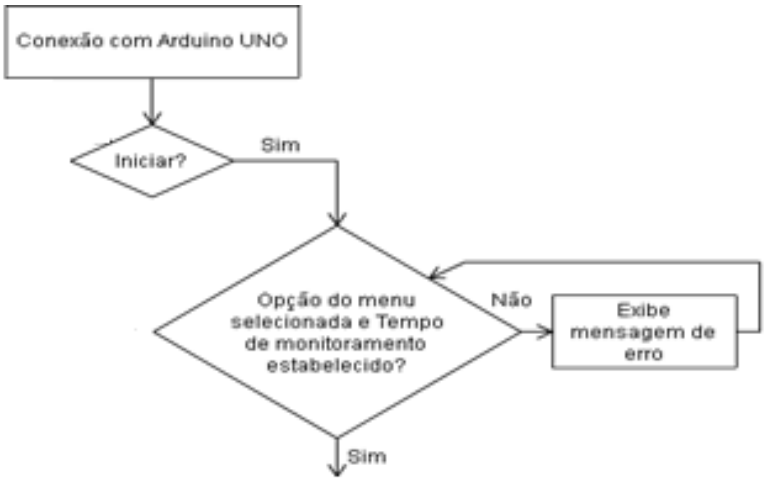

Figura 35: Fluxograma de verificação de parâmetros.

Fonte: Autoria própria.

Caso a opção do *menu* não seja válida aparece a mensagem de erro que pode ser vista na Figura 36, e essa verificação é feita através de um "*switch case*", onde a variável recebida

da função “*Menu*” é comparada com uma *string*, no caso a própria palavra “*Menu*”, se está for igual a mensagem é exibida e o usuário deve escolher outra opção.

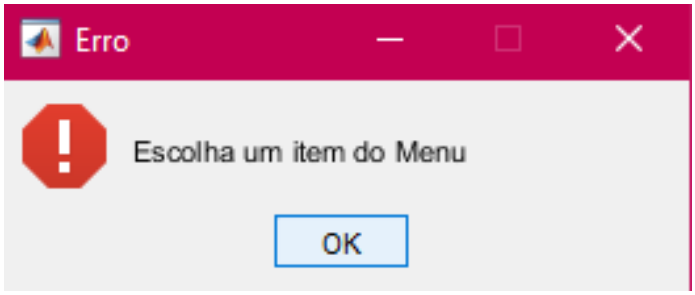

Figura 36: Mensagem de erro exibida caso o usuário não escolha um item do *menu*.
Fonte: Autoria própria.

E caso o usuário não digite um valor para o tempo de monitoramento, aparece a mensagem de erro presente na Figura 37, e isso é verificado pelo comando “*isempty*”, o qual verifica se a variável gravada pela função “Tempo de Monitoramento” é vazia, se isso ocorrer após a mensagem de erro o usuário deve digitar um valor para que sejam realizadas as leituras dos sensores.

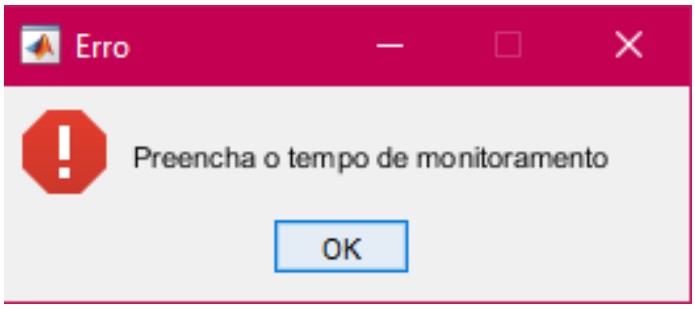

Figura 37: Mensagem de erro exibida caso o usuário não digite um tempo de monitoramento.
Fonte: Autoria própria.

As mensagens de erro são implementadas com o comando “*errordlg*”, que é um comando existente no próprio MATLAB®, e faz com que uma janela seja aberta e a mensagem desejada seja exibida.

4.7.3.2 Execução das leituras

Após as verificações e tudo estar correto inicia-se um laço de repetição que irá durar o tempo que o usuário determinou para o monitoramento, podendo ser visualizado no fluxograma da Figura 38. Este pode ser escolhido pelo usuário para ser em segundos, minutos, horas ou dias. A leitura dos sinais é realizada a cada 1 segundo, e para a duração de minutos, horas e dias, são realizadas multiplicações na função “Tempo de Monitoramento” por 60, 3600 e 86400, respectivamente.

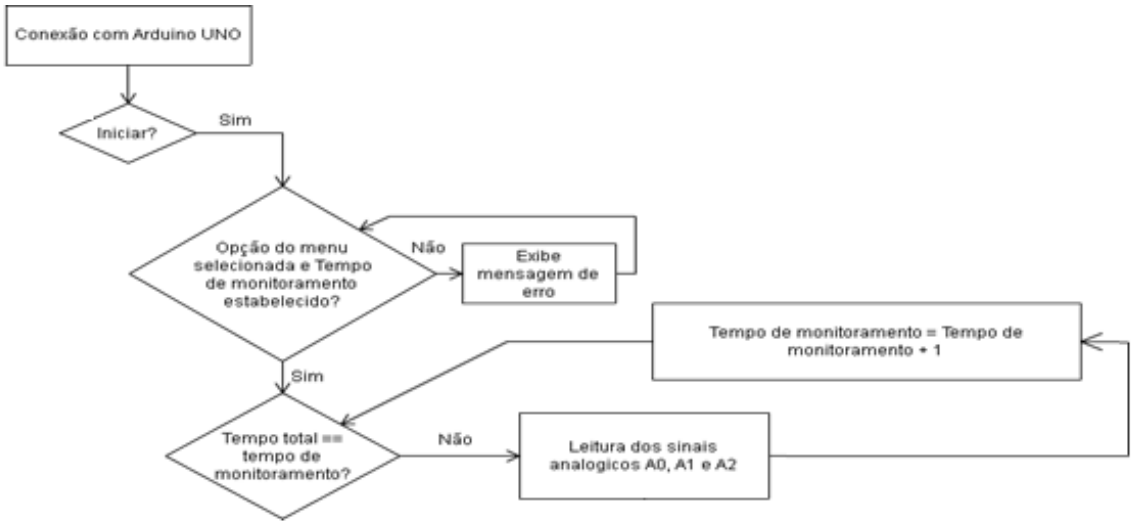

Figura 38: Fluxograma de execução das leituras dos sensores.

Fonte: Autoria própria.

Com esses dados estabelecidos, a função “Inicio” começa a realizar a leitura dos sinais dos sensores e ajusta seus valores para representação gráfica, como explicado anteriormente. Durante a execução do monitoramento, o botão de inicio fica verde, o que indica que o mesmo está sendo executado.

4.7.3.3 Plotagem e salvamento dos valores plotados nos gráficos

Com a leitura dos dados em andamento os gráficos são plotados em tempo real, e podem ser salvos com a função “Salvar Gráficos” a qualquer momento, e após o final do tempo são calculados os valores médios, mínimos e máximos dos sinais lidos, os quais serão utilizados para a função “Exibição dos Valores Médios, Mínimos e Máximos do Monitoramento”. O funcionamento geral da função pode ser visto no fluxograma da Figura 39.

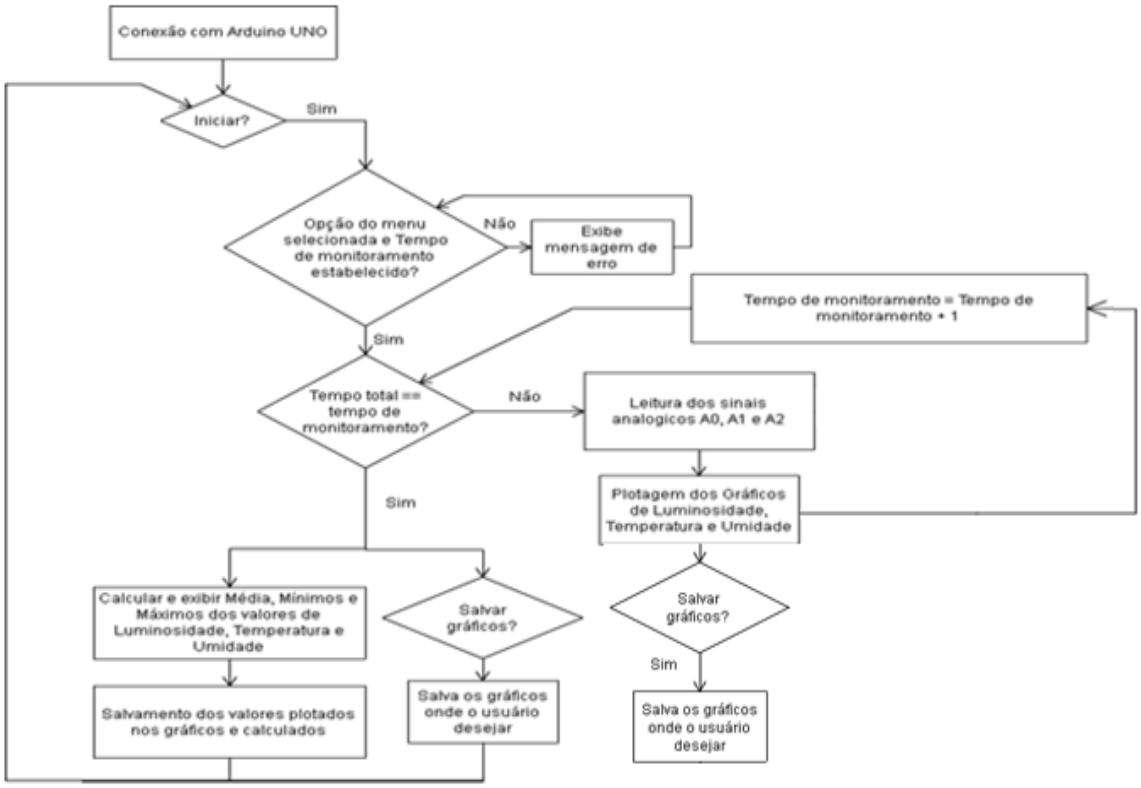

Figura 39: Fluxograma de plotagem dos gráficos e salvamento dos valores plotados e calculados.

Fonte: Autoria própria.

Todos os valores lidos e calculados são salvos em arquivos do tipo “.txt”, para que o usuário tenha acesso, mesmo que este não possua o *software* MATLAB®, e dessa forma é possível que o usuário utilize os dados para análises e estatísticas que podem auxiliar na produtividade e lucro.

4.7.3.4 Cálculo dos valores de média, mínimo e máximo

Como dito, os valores lidos pelos sensores são gravados em variáveis do tipo vetor, logo se percorridos podem ser encontrados os valores de mínimo e máximo através de comparações, sendo que o primeiro valor deve ser ignorado durante as comparações, pois sempre será igual a "zero", dessa forma o mesmo pode interferir nos cálculos.

Para se calcular a média todos os valores, menos o primeiro, são somados e posteriormente através do comando "*lenght*", o tamanho do vetor é obtido, para realizar a divisão. Considera-se que do tamanho obtido deve ser subtraído o valor 1, pois o primeiro valor é desconsiderado. A equação 9 representa como o cálculo é realizado.

$$Média = \frac{\sum_{i=1}^{i=n} Valores\ das\ leituras}{n-1} \quad (9)$$

4.7.3.5 Controle dos atuadores

A função "Inicio" também engloba o controle automático das saídas para os atuadores da estufa, sendo assim com os dados obtidos pela leitura dos sensores são realizadas comparações com valores pré-estabelecidos e caso atinja valores maiores ou menores que os determinados, um sinal pode ser enviado para que ocorra uma mudança em um atuador. As condições e mudanças realizadas podem ser visualizadas no fluxograma da figura 40.

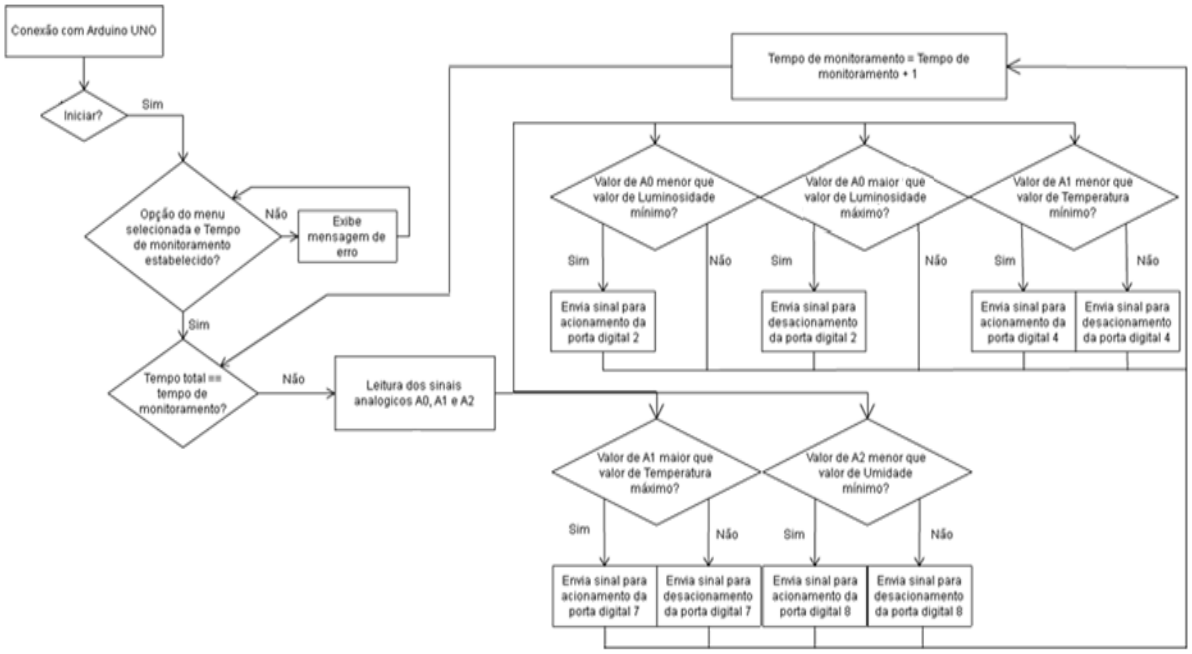

Figura 40: Fluxograma de controle dos indicadores.

Fonte: Autoria própria.

No caso tem-se três opções de frutas na função "*Menu*" e cada uma possui valores limites, sendo esses:

- Valor mínimo de luminosidade para que se acenda uma luz;
- Valor máximo de luminosidade para que se apague uma luz;
- Valor máximo de temperatura para que o ventilador seja ligado;
- Valor mínimo de temperatura para que um aquecedor seja ligado;
- Valor mínimo de umidade para que um irrigador seja desligado.

Levou-se em consideração os valores pré-estabelecidos, assim sinais são enviados através da função "*digitalWrite*", ou seja, se as condições acima citadas forem verdadeiras a saída recebe um sinal de nível alto, caso contrario recebe um sinal de nível baixo.

Por fim, após o tempo de monitoramento ser atingido, todas as saídas recebem um sinal de nível baixo, e assim o usuário define o que deseja fazer, podendo acionar as saídas de forma manual, ou iniciar um novo monitoramento.

4.7.4 Salvar Gráficos

A opção de salvar gráficos pode ser escolhida em qualquer momento do monitoramento, o que é visto no fluxograma da Figura 41, e a mesma irá salvar a imagem que está sendo apresentada nas janelas gráficas da interface gráfica.

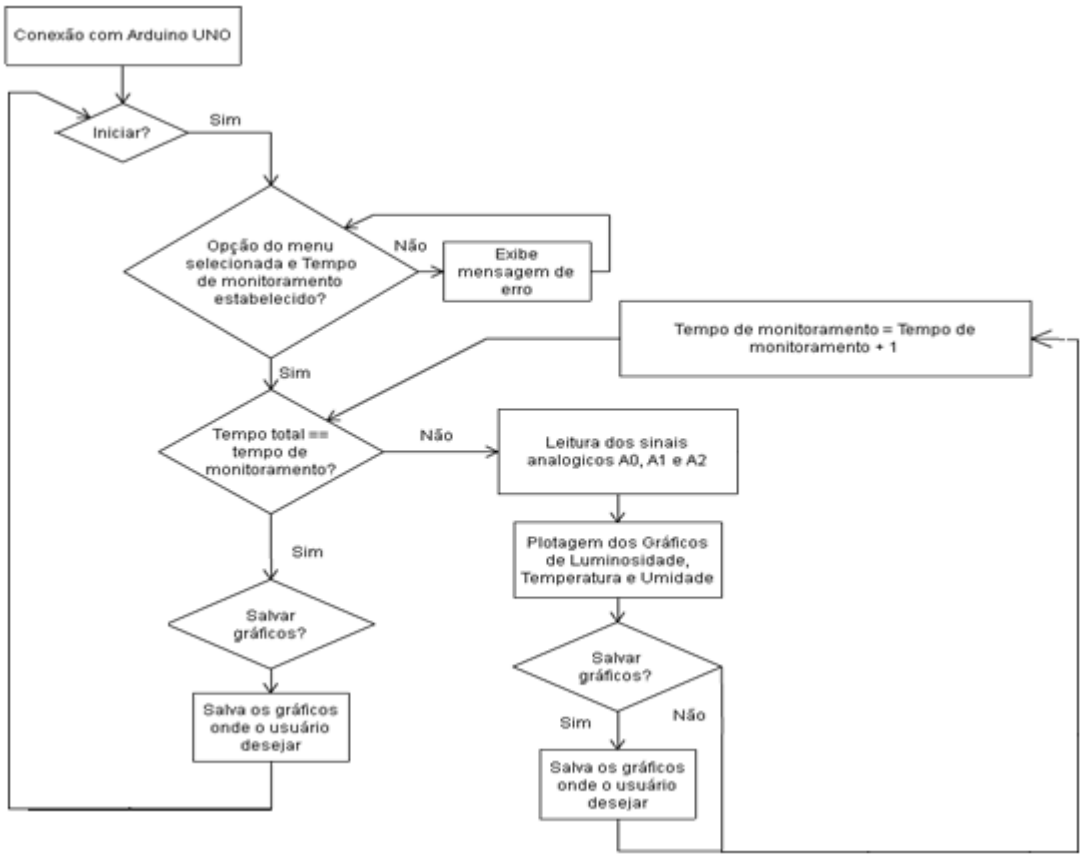

Figura 41: Fluxograma de salvamento dos gráficos.

Fonte: Autoria própria.

O método utilizado permite que o usuário selecione o local que deseja salvar a imagem, que será do tipo BMP (*Bitmap*), que é um tipo de imagem que contém a descrição de cada pixel. Com esse método cada imagem é salva separadamente, sendo assim ao clicar no botão “Salvar Gráficos” três janelas abriram-se para que o usuário selecione o local de armazenamento e salve o gráfico, a janela pode ser vista na Figura 42, e se for observado o título da janela informa qual gráfico está sendo salvo.

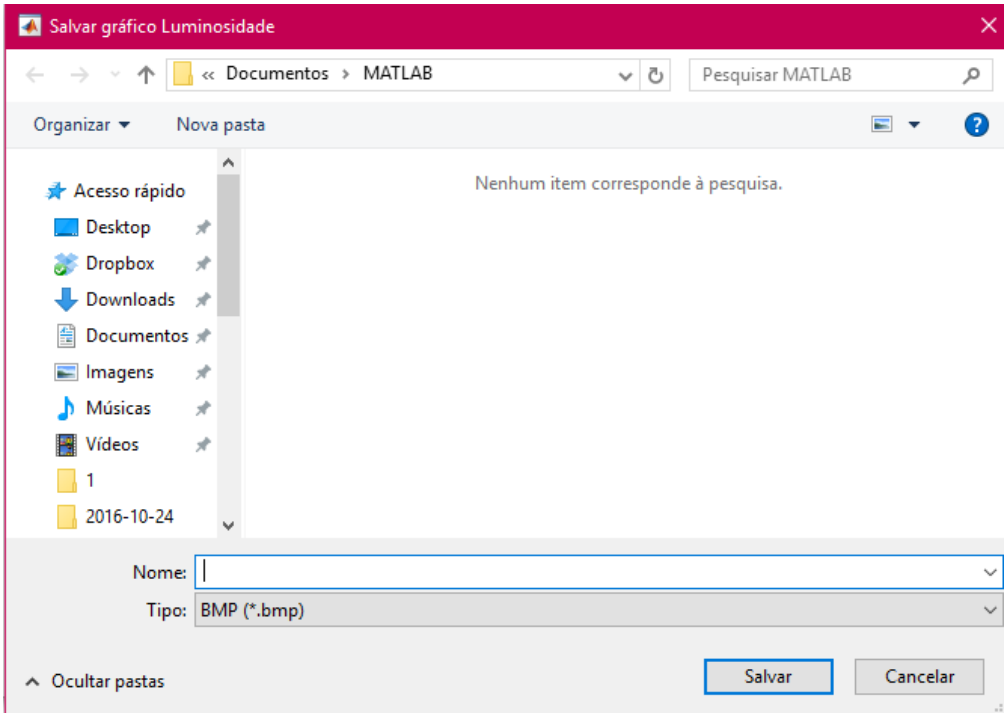

Figura 42: Janela para salvar os gráficos.

Fonte: Autoria própria.

4.7.5 Exibição dos Valores Médios, Mínimos e Máximos do Monitoramento.

Essa função é realizada com o auxílio da função Inicio, pois os valores utilizados para os cálculos são obtidos na leitura dos sensores que é realizada pela função citada, e o fluxograma de seu funcionamento pode ser visto na Figura 43.

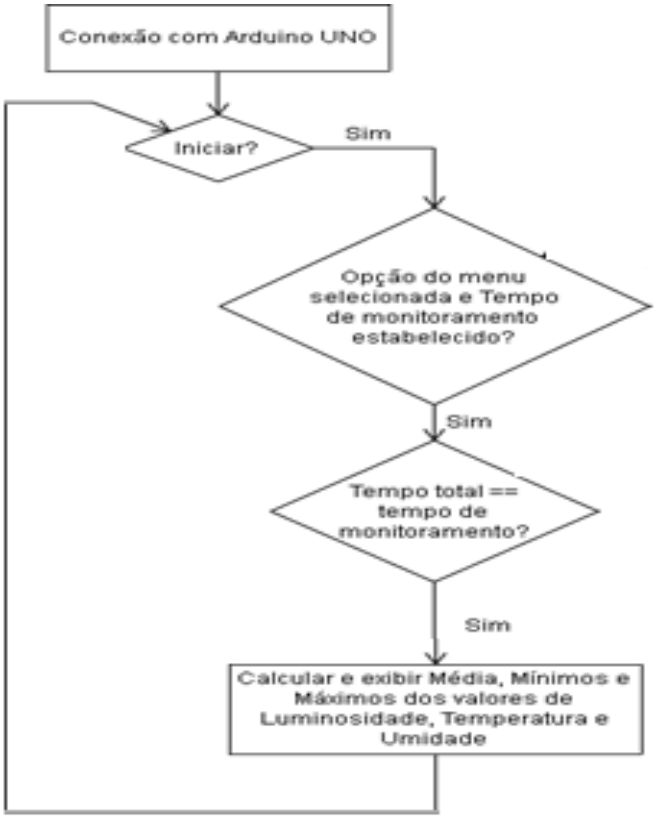

Figura 43: Fluxograma de exibição dos valores calculados.

Fonte: Autoria própria.

Os valores são salvos em vetores, sendo separados por Luminosidade, Temperatura e Umidade, logo são calculados os valores médios de cada leitura e encontrados os valores de mínimo e máximo dentro dos vetores. Estes são salvos em variáveis que posteriormente serão exibidas ao usuário através dos campos presentes na interface gráfica.

Esses valores ajudarão o usuário caso o mesmo deseje fazer comparativos entre os monitoramentos, para assim obter dados estatísticos e poder melhorar sua produção, para conhecer melhor o ambiente em que se produz.

4.7.6 Botões de Liga e Desliga dos Atuadores da Estufa

Os botões são implementados por diferentes funções, sendo uma para cada botão existente, assim além da função do botão de “Salvar Gráficos”, existem mais oito funções para botões.

Quatro dessas realizam o acionamento dos atuadores, enviando sinal lógico alto para as saídas do Arduino® através do comando "*digitalWrite*(Saida_Digital,1)", onde para cada função o valor da saída digital é um, correspondente ao pino de saída em que o atuador está ligado no Arduino®.

Já as outras quatro realizam o desligamento dos atuadores, enviando sinal lógico baixo para as saídas do Arduino® através do comando "*digitalWrite*(Saida_Digital,0)".

O funcionamento geral pode ser visto no fluxograma presente na Figura 44.

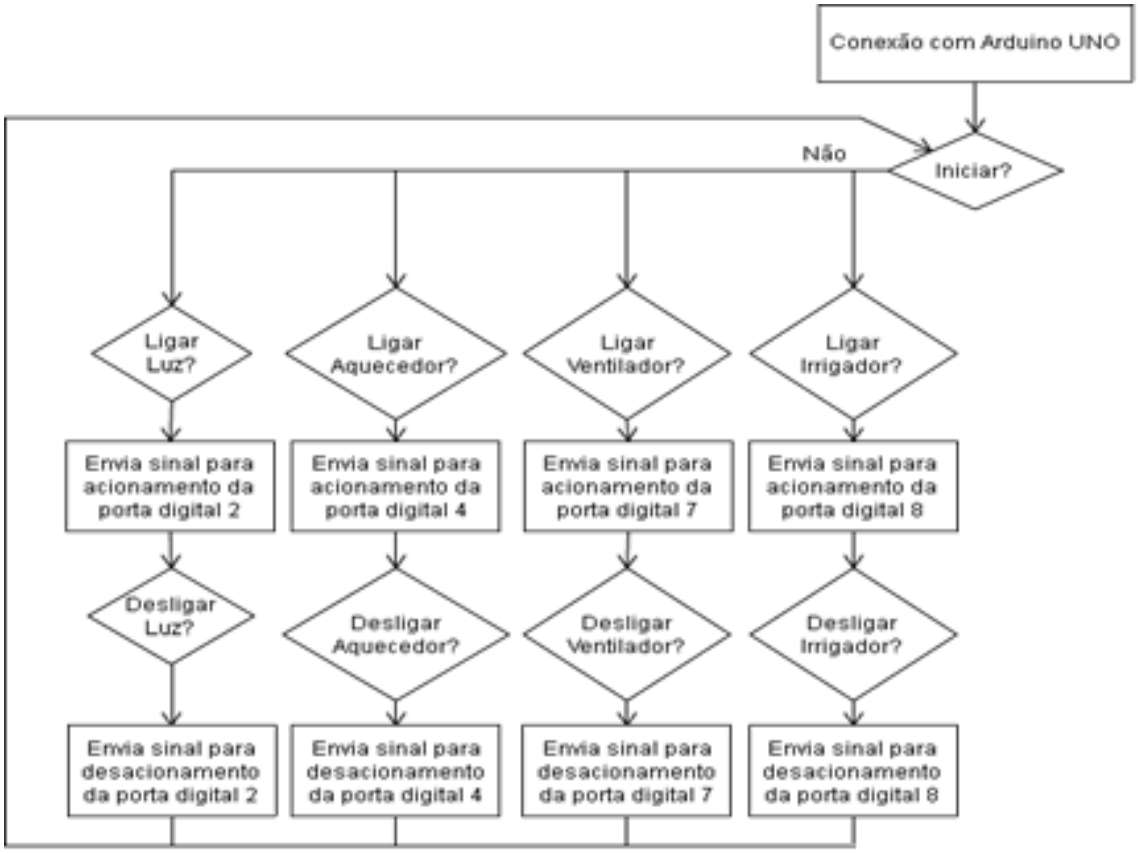

Figura 44: Fluxograma do funcionamento dos botões.

Fonte: Autoria própria.

4.8 EXECUTÁVEL

Após a implementação da plataforma é necessário que se crie um programa executável da interface, pois assim o usuário não necessita comprar a licença do *software* MATLAB®.

4.8.1 Criação do Projeto

Primeiramente deve-se abrir o *software* MATLAB®, e o mesmo deve ter instalado pelo menos um dos pacotes entre, MATLAB Compiler®, MATLAB Compiler SDK® e MATLAB Production Server®. Em seguida deve-se digitar na janela de comandos "*deploytool*", o que irá abrir uma nova janela, vista na Figura 45, onde se deve colocar o nome do arquivo desejado e escolher a pasta que irá salvá-lo.

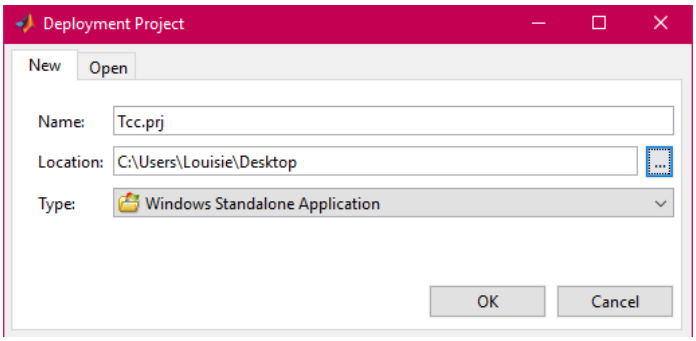

Figura 45: Janela de criação de projeto.
Fonte: Autoria própria.

Após essa etapa é necessário selecionar os arquivos do tipo ".m" e ".fig" na janela "*Windows Standalone Application*", e então na mesma janela constrói-se o projeto com os arquivos realizados, ao clicar no botão indicado na Figura 46.

Figura 46: Janela *Windows Standalone Application*, botão *"Build".*

Fonte: Autoria própria.

4.8.2 Criação do Pacote

Com o projeto criado, deve-se ir até a aba "*Package*" e adicionar o MATLAB Compiler Runtime®, e selecionar a primeira opção na janela que irá se abrir o que pode ser visto na Figura 47.

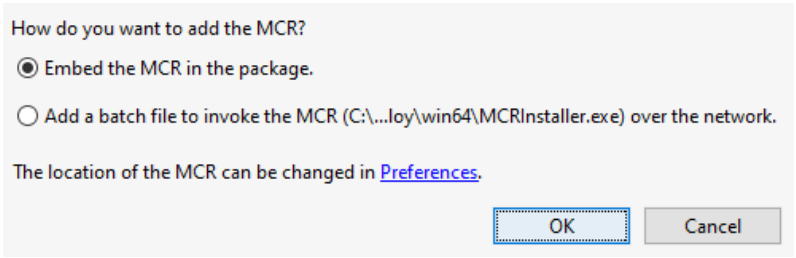

Figura 47: Janela para adicionar MCR.

Fonte: Autoria própria.

Posteriormente deve-se selecionar o projeto criado e clicar no botão mostrado na Figura 48, o que irá gerar o pacote do executável.

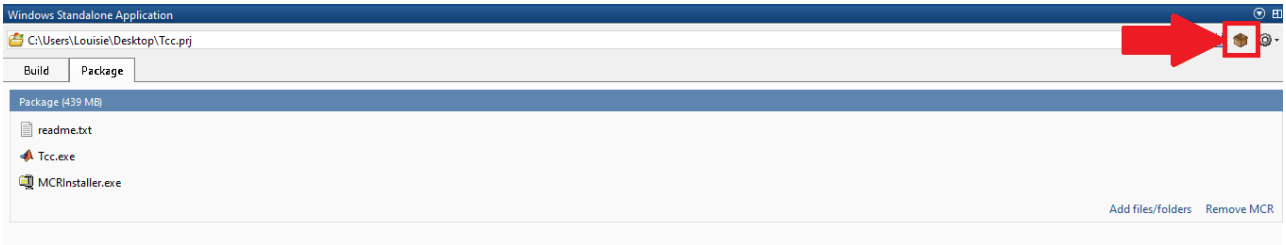

Figura 48: Janela "*Windows Standalone Application*", botão "*Package*".

Fonte: Autoria própria.

4.8.3 Instalação

Após ter sido criado o projeto e o pacote deve-se iniciar o projeto de instalação, sendo que este deve ser feito no computador do próprio usuário, para que o mesmo consiga utilizar a plataforma sem problemas.

Dessa maneira encerra-se a etapa de criação do executável e o programa está pronto para uso.

5 RESULTADOS

No presente capítulo são apresentados resultados obtidos através de simulações realizadas com o protótipo implementado, de acordo com o que foi explicado previamente, que pode ser visto na Figura 49. Tais simulações foram realizadas com o intuito de se representar variações de luminosidade, temperatura e umidade do solo presentes em uma estufa.

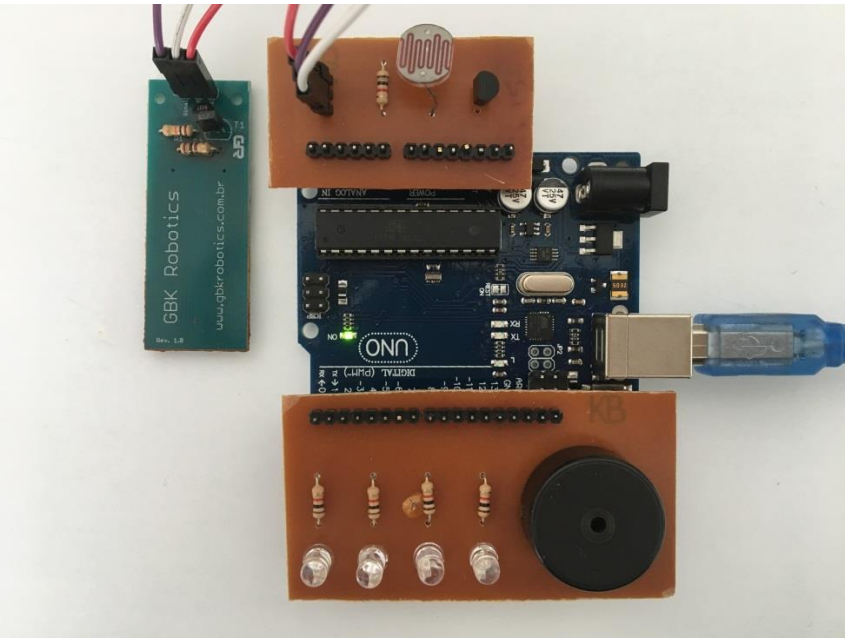

Figura 49: Protótipo implementado para leitura dos sensores e acionamento dos indicadores.
Fonte: Autoria própria.

Considerou-se o fato de que seria necessário deixar o protótipo em funcionamento para a realização do monitoramento por um longo período de tempo para que se obtivessem variações consideráveis de luminosidade, temperatura e umidade, optou-se pela realização de experimentos que utilizassem agentes que poderiam acelerar esse processo.

5.1 CONDIÇÕES PARA CONTROLE DOS ATUADORES

Para um melhor entendimento dos resultados obtidos é necessário saber as condições de luminosidade, temperatura e umidade atribuídas para cada item do *menu* na realização da plataforma. Como o *menu* foi feito de forma genérica os itens são “Fruta 1”, “Fruta 2” e “Fruta 3”, e suas condições podem ser vistas na Tabela 1.

Tabela 1: Condições de luminosidade, temperatura e umidade utilizados para obtenção dos resultados.

Condições	Fruta 1	Fruta 2	Fruta 3
Luminosidade (Lux)	$100 < L \leq 300$	$200 < L \leq 400$	$300 < L \leq 500$
Temperatura (ºC)	$20 < T \leq 25$	$25 < T \leq 30$	$30 < T \leq 35$
Umidade (%)	$U \geq 50$	$U \geq 40$	$U \geq 60$

Fonte: Autoria própria.

5.2 CONDIÇÕES PARA REALIZAÇÃO DAS SIMULAÇÕES

5.2.1 Luminosidade

No caso, para diferentes luminosidades o protótipo foi colocado em uma sala fechada com lâmpadas e também foi utilizada uma lanterna para maior intensidade luminosa.

Dessa forma, o sensor poderia receber todas as intensidades luminosas compreendidas em sua faixa de medição, inclusive valores que estão presentes em uma estufa.

5.2.2 Temperatura

Para uma rápida variação de temperatura o protótipo foi colocado em uma sala fechada com ar-condicionado, o mesmo estava programado para uma temperatura de 18ºC, sendo assim deveria ser aquecido para o aumento de temperatura, e resfriado para a diminuição de temperatura.

Para isso foi utilizado um secador de cabelo, que gradativamente elevou a temperatura ao redor do sensor, e uma pedra de gelo que possibilitou o decrescimento do valor de temperatura no mesmo.

5.2.3 Umidade

Para se captar diferentes medidas de umidade do solo o sensor foi introduzido em um vaso com terra primeiramente sem água para que a umidade lida pelo sensor tivesse um valor reduzido. Aos poucos se colocou água no mesmo vaso, para que a umidade fosse elevada, e por fim o sensor foi retirado da terra, para uma redução da umidade, porém sem zerar a mesma pois o sensor ainda estaria sujo.

5.3 SIMULAÇÕES

Os resultados foram obtidos através das simulações, e as mesmas foram apresentadas a seguir, onde se considerou as condições anteriormente apresentadas.

Foram realizadas simulações com o tempo de 10 minutos, para que fosse possível analisar cada situação e verificar o correto funcionamento da plataforma, e assim chegar a uma conclusão plausível.

5.3.1 Luminosidade

Como apresentado anteriormente, para as leituras de luminosidade, primeiramente o sensor encontrou-se em uma sala escura, logo uma lâmpada foi acesa, e por fim uma lanterna aproximada do sensor para o aumento de luminosidade.

Para se obter variações, as ações descritas foram repetidas no intervalo do tempo de monitoramento, o que pode ser visto na Figura 50.

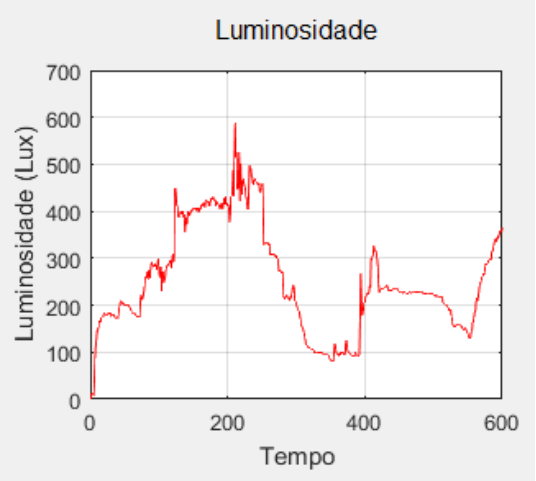

Figura 50: Gráfico de luminosidade no tempo de 10 minutos.

Fonte: Autoria própria.

Através dos pontos plotados no gráfico, o *software* realiza o cálculo de média de variação, e encontra os valores de mínimo e máximo, o que pode ser visto na Figura 51.

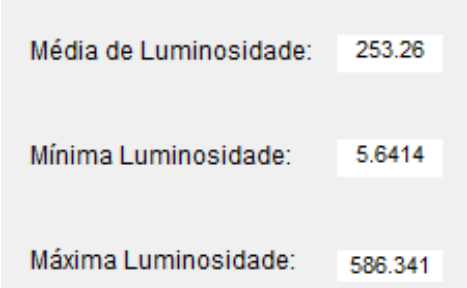

Figura 51: Valor médio, mínimo e máximo da leitura de luminosidade.

Fonte: Autoria própria.

Já na Figura 52 é possível ver o indicador de falta de luminosidade ligado quando necessário, pois a luminosidade ambiente não é suficiente.

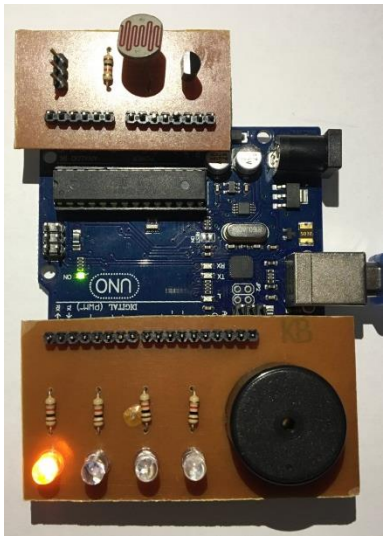

Figura 52: Indicador da saída de acionamento de luz ligado.

Fonte: Autoria própria.

5.3.2 Temperatura

As variações de temperatura foram realizadas com um secador de cabelo e uma pedra de gelo, considerou-se também que o protótipo foi colocado em uma sala com o ar-condicionado ligado programado em 18ºC, como dito anteriormente.

Assim aos poucos aproximou-se o secador para o aumento de temperatura, e em seguida afastou-se. Logo, colocou-se sobre o sensor uma pedra de gelo para a queda de temperatura. Esse procedimento foi repetido diversas vezes no tempo de monitoramento, gerando o gráfico presente na Figura 53.

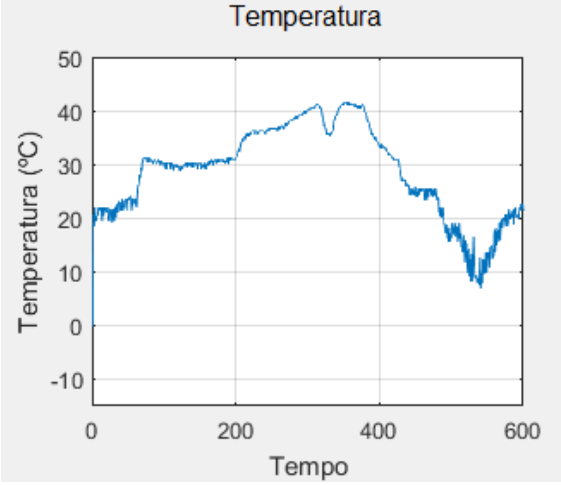

Figura 53: Gráfico de temperatura no tempo de 10 minutos.

Fonte: Autoria própria.

Através do gráfico nota-se a variação de temperatura, e o valor médio, mínimo e máximo dessa variação podem ser vista na Figura 54.

Figura 54: Valor médio, mínimo e máximo da leitura de temperatura.

Fonte: Autoria própria.

No caso dos indicadores para a variação de temperatura, tem-se um indicador para resfriamento e um para aquecimento.

O indicador de resfriamento pode ser visto na Figura 55, o qual é acionado quando a temperatura fica acima do valor considerado ideal.

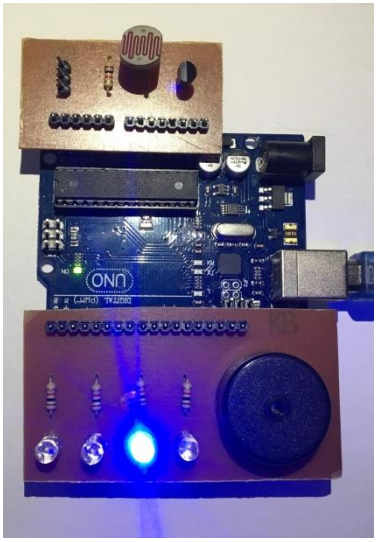

Figura 55: Indicador da saída de acionamento do ventilador ligado.

Fonte: Autoria própria.

Já o indicador de aquecimento pode ser visto na Figuras 56 , e o mesmo é acionado quando o valor de temperatura é inferior ao considerado ideal.

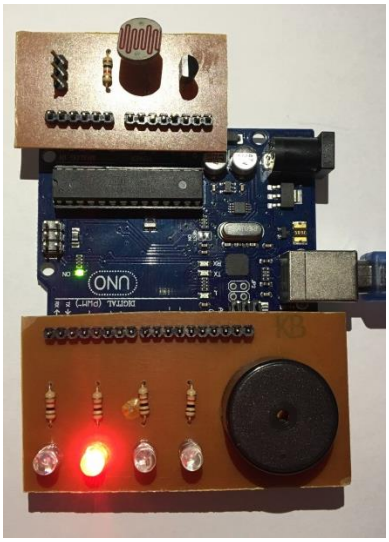

Figura 56: Indicador da saída de acionamento do aquecedor ligado.

Fonte: Autoria própria.

5.3.3 Umidade

Como dito anteriormente, o sensor de umidade de solo foi inserido em um recipiente com terra seca, e aos poucos foi adicionado água para a variação de umidade, e no final retirado do recipiente.

Nesse caso, para que a umidade voltasse ao estado anterior em um curto período de tempo, o sensor deveria ser retirado do recipiente e inserido em outro, que continha terra seca. No entanto, a retirada do sensor acarretaria em uma leitura inadequada, que prejudicou a simulação e o resultado final, pode-se notar o aumento de umidade e posterior decaimento pelo gráfico presente na Figura 57.

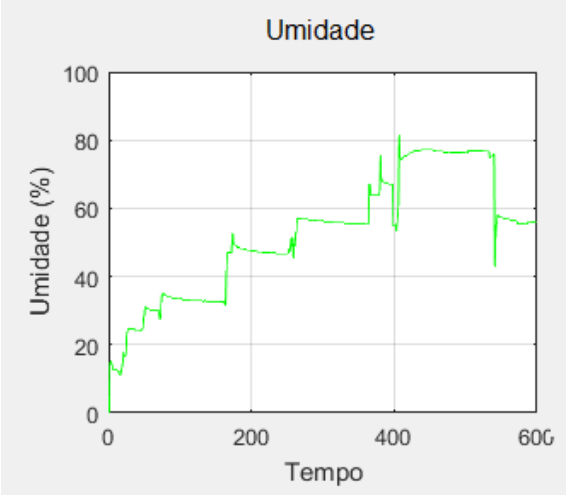

Figura 57: Gráfico de umidade no tempo de 10 minutos.

Fonte: Autoria própria.

Com o gráfico pronto foram obtidos os valores da variação, sendo esses, valor médio, valor mínimo e valor máximo de umidade atingido, os quais podem ser vistos na Figura 58.

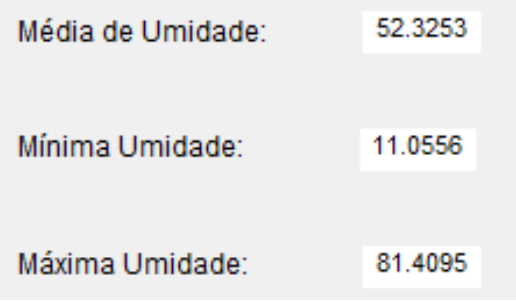

Figura 58: Valor médio, mínimo e máximo da leitura de umidade.

Fonte: Autoria própria.

Por fim, pode ser visto o funcionamento do indicador que representa o acionamento de um irrigador, esse estando ligado na Figura 5, o qual atuou de acordo com a necessidade de aumentar a umidade presente no solo.

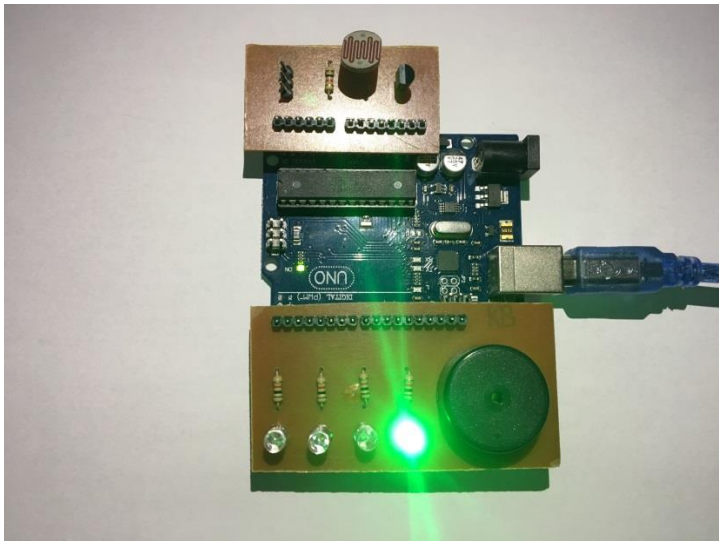

Figura 59: Indicador da saída de acionamento do irrigador ligado.

Fonte: Autoria própria.

6 CONCLUSÃO

Com o intuito da melhoria na produtividade e o uso consciente de recursos naturais, os produtores aderiram ao uso de tecnologias nos ambientes de cultivo. O uso de sistemas de monitoramento e controle se tornou mais comum, e como citado anteriormente, segundo Coutinho (2010), o produtor que não faz uso de tecnologias perde competitividade no mercado global.

Uma maneira de realizar o monitoramento de uma estufa é através de sensores que realizam a leitura dos sinais desejados e os transmitem para um *software* que irá fazer o controle do ambiente.

Dessa forma, este trabalho apresentou o estudo do uso de sensores, com o intuito de se escolher os que melhor adequam-se as leituras necessárias ao projeto, além da melhor forma de comunicação entre os sensores, a plataforma criada e os possíveis atuadores presentes na estufa.

A plataforma foi criada com o pensamento no usuário final, sendo que a mesma possui uma interface de fácil entendimento e com os recursos necessários para que o usuário possa monitorar a luminosidade, a temperatura e a umidade do ambiente. Bem como ligar ou desligar os atuadores manualmente quando o monitoramento não estiver sendo realizado, e obter os valores de média, mínimo e máximo atingidos durante um determinado tempo de monitoramento, o qual o próprio usuário determina.

Com a diversidade de culturas que podem ser produzidas por um mesmo produtor, a plataforma contém um *menu* que pode ser modificado, ao inserir ou excluir culturas a serem cultivadas, sendo que cada uma apresenta condições ideais para o cultivo, assim sendo personalizada para cada usuário, para suprir suas necessidades.

Além dos fatores citados acima a plataforma permite o salvamento de dados, sendo esses os valores obtidos nas leituras e os valores calculados, disponibilizados em um arquivo de texto, bem como as imagens gráficas plotadas. Assim permite ao usuário analisar os dados obtidos pelo monitoramento, sendo possível estudá-los e consequentemente realizar melhorias em sua produtividade.

Devido ao fato dos dados serem salvos a plataforma pode ser utilizada também para experimentos na área de produção agrícola que necessitem da leitura dos dados de luminosidade, temperatura e umidade.

Áreas da saúde como hospitais, UTIs (Unidade de Terapia Intensiva, Unidade de

Tratamento Intenso) e laboratórios necessitam de monitoramento e controle de luminosidade, temperatura e umidade. Bem como na área alimentícia, o armazenamento de matéria prima, e até mesmo do produto final necessitam de um monitoramento e controle adequado. Dessa forma a plataforma criada pode ser utilizada em outras áreas onde se deseje realizar o monitoramento e controle das variáveis de luminosidade, temperatura e umidade.

Por fim o projeto pode ser utilizado como base para futuros trabalhos, podem ser adicionados sensores, realizadas análises pela própria plataforma, a adição de atuadores e até mesmo a aplicação em outras áreas.

Dessa forma ao analisar os resultados obtidos e as possibilidades de continuidade do trabalho, pode-se concluir que o monitoramento e controle são feitos de forma eficiente. Sendo que as alterações das medidas de luminosidade, temperatura e umidade são rapidamente identificadas e mostradas ao usuário, e todos os dados salvos de forma clara.

REFERÊNCIAS

BEZERRA, Pedro André Martins; FONTANELLE, Luís Fernando Almeida; SANTOS, Luis Paulo Carvalho dos. **Apostila de MATLAB.** Fortaleza, Ceará, 2014. Universidade Federal do Ceará. Disponível em: <http://www.peteletrica.ufc.br/Apostilas/MATLAB - PET-EE.pdf>. Acesso em: 05 out. 2015.

BRAGA, Newton C. **Como funcionam os fotodiodos**. 2014. Disponível em: < http://www.newtoncbraga.com.br/index.php/como-funciona/4715-art1181>. Acesso em: Out 2015.

BRAGA, Newton C. **Como funcionam os sensores de temperatura**. 2014. Disponível em: < http://www.newtoncbraga.com.br/index.php/como-funciona/6097-art764>. Acesso em: Out 2015.

COUTINHO, Flávio. **Tecnologia na agricultura**. 2010. Disponível em: <http://meioambiente.culturamix.com/agricultura/tecnologia-na-agricultura>. Acesso em: Set 2015.

CUGNASCA, C. E.; HIRAKAWA, A. R.. **Comunicação Serial.** 2006. Disponível em: <http://www.pcs.usp.br/~pcs2497/aula serial.pdf>. Acesso em: Out. 2015.

FIGUEIREDO, Gilberto J. B. **CASA DA AGRICULTURA: Produção em Ambiente Protegido.** São Paulo: Cecor, v. 2, n. 14, 2011.

GASPAR, Pedro Dinis; SANTO, Antônio Espírito e SOUZA, J. A. M. Felippe de. **Apontamentos de MATLAB: Introdução ao MATLAB**. 2002. Disponível em: <http://www.demnet.ubi.pt/~felippe/texts3/apmatlab_vol2.pdf> Acesso em: Set. 2015.

GREGO, Maurício. **O hardware em 'código aberto'.** 2009. Disponível em: <http://info.abril.com.br/professional/tendencias/hardware-livre-leve-e-solto.shtml?4>. Acesso em: Out. 2015.

GUEDES, Italo. **Cultivo em estufas: driblando o imprevisível**. 2009. Disponível em: <http://scienceblogs.com.br/geofagos/2009/04/cultivo_em_estufas_driblando_o/>. Acesso em: Set 2015.

KAMPF, Atelene Normann. **Produção comercial de plantas ornamentais**. 2. ed. Agrolivros. 2005.

LITJENS, Otto Jacob. **Automação de estufas agrícolas utilizando sensoriamento remoto e o protocolo zigbee**. 2009. Escola de engenharia de São Carlos. São Carlos.

MARTINO, J. M. de. **Micro e Minicomputadores:** Hardware. 2004. Disponível em: <http://www.dca.fee.unicamp.br/courses/EA078/1s2004/arquivos/turma_ab/cap8.pdf>. Acesso em: Out. 2015.

MELO, Rosecléa L. O. **Implementação de controle digital PID com microcomputador**, 1985, Dissertação (Mestrado em Física) – Universidade Estadual de Campinas, Campinas.

MENDES J, José Jair Alves; STEVAN J., Sérgio Luiz. **LDR e Sensores de luz ambiente: funcionamento e aplicações**. Ponta Grossa: UTFPR, 2013.

O'BRIEN, Duane. **Construindo um Jogo de Laser Baseado no Arduino, Parte 1: Princípios Básicos do Arduino.** 2008. Disponível em: <http://www.ibm.com/developerworks/br/library/os-arduino1/index.html>. Acesso em: Set. 2015.

PATSKO, Luis Fernando. **Tutorial, Aplicações, Funcionamento e Utilização de Sensores.** 2006.

PORTA, Leonardo Dalla. **Sensor de umidade de solo**. 2016. Disponível em: < http://blog.usinainfo.com.br/sensor-de-umidade-de-solo/>. Acesso em: Jun. 2016.

SILVA, Clodoaldo. **Amplificadores Operacionais**. 2006. Disponível em: < http://www.clubedaeletronica.com.br/Eletronica/HTML/Circuitos%20comparadores.htm>. Acesso em: Set 2015.

SOARES, Marcio José. **Conectando um microcontrolador a um PC:** Comunicação serial RS-232 com microcontrolador PIC. 2014. Disponível em: <http://www.arnerobotics.com.br/eletronica/comunicacao_rs232_PIC.htm>. Acesso em: Set. 2015.

SOUZA, Fabio. **Arduino: Comunicação Serial**. 2014. Disponível em: <http://www.embarcados.com.br/arduino-comunicacao-serial/>. Acesso em: Out. 2015.

TANENBAUM, Andrew S.. **Redes de Computadores.** 5. ed. Pearson, 2003. Tradução de: Vandenberg D. de Souza.

THOMAZINI, Daniel; ALBUQUERQUE, Pedro Urbano Braga de. **Sensores Indutriais: Fundamentos e Aplicações**. 4. ed. Érica, 2012.
.

THOMPSON, Clive. **Build It. Share It. Profit. Can Open Source Hardware Work?** 2008. Disponível em: <http://www.wired.com/2008/10/ff-openmanufacturing/>. Acesso em: Out. 2015.

TONINI, Adriana M.; COUTO, Bráulio R.G.M. **Ensinando Geometria Analítica com uso do MATLAB**. 1999. Departamento de Ciências Exatas e Tecnologia do Centro Universitário de Belo Horizonte / DECET - UniBH.

VAN DER HOEVEN. **Estufas agrícolas e galvanização a fogo**. 2015. Disponível em: < http://www.vdh.com.br/?pg=home >. Acesso em Out. 2015.

WENDLING, Marcelo. **Sensores.** 2010. Disponível em: <http://www2.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/4---sensores-v2.0.pdf>. Acesso em: Out. 2015.

APÊNDICES

APÊNDICE A – CÓDIGO ARDUINO®

```
#if    defined(__AVR_ATmega1280__)    ||
defined(__AVR_ATmega2560__)
#define INTERNAL INTERNAL1V1
#endif
void setup() {
  Serial.begin(115200);
}
void loop() {
  static int  s   = -1;
  static int  pin = 13;
  int  val =  0;
  int  agv =  0;
  int  dgv =  0;
  if (Serial.available() >0) {
    val = Serial.read();
    switch (s) {
      case -1:
      if (val>47 && val<90) {
        s=10*(val-48);
      }
      if ((s>40 && s<90) || (s>90 && s!=340
&& s!=400)) {
        s=-1;
      }
      break;
      case 0:
      if (val>98 && val<167) {
        pin=val-97;
        s=1;
      }
      else {
        s=-1;
      }
      break;
      case 1:
      if (val>47 && val<50) {
        if (val==48) {
          pinMode(pin,INPUT);
        }
        else {
          pinMode(pin,OUTPUT);
        }
      }
      s=-1;
      break;
      case 10:
      if (val>98 && val<167) {
        pin=val-97;
        dgv=digitalRead(pin);
        Serial.println(dgv);
      }
      s=-1;
      break;
      case 20:
      if (val>98 && val<167) {
        pin=val-97;
        s=21;
      }
      else {
        s=-1;
      }
      break;
      case 21:
      if (val>47 && val<50) {
        dgv=val-48;
digitalWrite(pin,dgv);
      }
      s=-1;
      break;
      case 30:
      if (val>96 && val<113) {
        pin=val-97;
        agv=analogRead(pin);
Serial.println(agv);
      }
      s=-1;
      break;
```

```
      case 40:
      if (val>98 && val<167) {
        pin=val-97;
        s=41;
      }
      else {
        s=-1;
      }
      break;
      case 41:
      analogWrite(pin,val);
      s=-1;
      break;
      case 90:
      if (val==57) {
        Serial.println(0);
      }
      s=-1;
      break;
      case 340:
     #if            defined(__AVR__)            ||
defined(__PIC32MX__)
      switch (val) {
        case 48:
        analogReference(DEFAULT);
        break;
        case 49:
        analogReference(INTERNAL);
        break;
        case 50:
        analogReference(EXTERNAL);
        break;
        default:
        break;
      }
#endif
      s=-1;
      break;
      case 400:
      Serial.println(val);
      s=-1;
      break;
      default:
      s=-1;
    }
  }
}
```

APÊNDICE B – CÓDIGO MATLAB®

```
function varargout = Example1(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Example1_OpeningFcn, ...
                   'gui_OutputFcn',  @Example1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

% --- Executes just before Example1 is made visible.
function Example1_OpeningFcn(hObject, ~, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
delete(instrfind({'Port'},{'COM4'}))

% --- Outputs from this function are returned to the command line.
function varargout = Example1_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
clear a;
global a;
a = arduino('COM4');
a.pinMode(2, 'output');
a.pinMode(4, 'output');
a.pinMode(7, 'output');
a.pinMode(8, 'output');
a.pinMode(13, 'output');

% --- Executes on selection change in menu_tempo.
function menu_tempo_Callback(hObject, eventdata, handles)
conteudo2=get(hObject,'String');
valor2=get(hObject,'Value');
handles.mtempo=conteudo2(valor2);
guidata(hObject,handles);
```

```
% --- Executes during object creation, after setting all properties.
function menu_tempo_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in menu1.
function menu1_Callback(hObject, eventdata, handles)
conteudo=get(hObject,'String');
valor=get(hObject,'Value');
handles.escolha=conteudo(valor);
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function menu1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in light_off_button.
function light_off_button_Callback(hObject, eventdata, handles)
global a;
a.digitalWrite(2,0);

% --- Executes on button press in light_on_button.
function light_on_button_Callback(hObject, eventdata, handles)
global a;
a.digitalWrite(2,1);

% --- Executes on button press in irriga_off_button.
function irriga_off_button_Callback(hObject, eventdata, handles)
global a;
a.digitalWrite(8,0);

% --- Executes on button press in irriga_on_button.
function irriga_on_button_Callback(hObject, eventdata, handles)
global a;
a.digitalWrite(8,1);

% --- Executes on button press in hot_off_button.
function hot_off_button_Callback(hObject, eventdata, handles)
```

```
global a;
a.digitalWrite(4,0);

% --- Executes on button press in hot_on_button.
function hot_on_button_Callback(hObject, eventdata, handles)
global a;
a.digitalWrite(4,1);

% --- Executes on button press in cold_off_button.
function cold_off_button_Callback(hObject, eventdata, handles)
global a;
a.digitalWrite(7,0);

% --- Executes on button press in cold_on_button.
function cold_on_button_Callback(hObject, eventdata, handles)
global a;
a.digitalWrite(7,1);

% --- Executes on button press in save_button.
function save_button_Callback(hObject, eventdata, handles)
formatos = {'*.bmp','BMP (*.bmp)'};
%Luminosidade
rgb = getframe(handles.axes1);
[nome,rota] = uiputfile(formatos,'Salvar gráfico Luminosidade');
figura = figure;
unidades = get(handles.axes1,'Unidades');
posicao = get(handles.axes1,'Posicao');
objeto_1 = copyobj(handles.axes1,figura);
set(objeto_1,'Unidades',unidades);
set(objeto_1,'Posicao',[15 5 posicao(3) posicao(4)]);
set(figura,'Unidades',unidades);
set(figura,'Posicao',[15 5 posicao(3)+30 posicao(4)+10]);
saveas(figura,[rota nome])

%Temperatura
rgb = getframe(handles.axes2);
[nome,rota] = uiputfile(formatos,'Salvar gráfico Temperatura');
figura = figure;
unidades = get(handles.axes2,'Unidades');
posicao = get(handles.axes2,'Posicao');
objeto_1 = copyobj(handles.axes2,figura);
set(objeto_1,'Unidades',unidades);
set(objeto_1,'Posicao',[15 5 posicao(3) posicao(4)]);
set(figura,'Unidades',unidades);
```

```
set(figura,'Posicao',[15 5 posicao(3)+30 posicao(4)+10]);
saveas(figura,[rota nome])

%Umidade
rgb = getframe(handles.axes3);
[nome,rota] = uiputfile(formatos,'Salvar gráfico Umidade');
figura = figure;
unidades = get(handles.axes3,'Unidades');
posicao = get(handles.axes3,'Posicao');
objeto_1 = copyobj(handles.axes3,figura);
set(objeto_1,'Unidades',unidades);
set(objeto_1,'Posicao',[15 5 posicao(3) posicao(4)]);
set(figura,'Unidades',unidades);
set(figura,'Posicao',[15 5 posicao(3)+30 posicao(4)+10]);
saveas(figura,[rota nome])

% --- Executes on button press in begin_button.
function begin_button_Callback(hObject, eventdata, handles)

%Inicialização de variáveis
global a;
luminosidade=0;
temperatura=0;
umidade=0;
menu=1;

%luminosidade
min_lum = 4000;
max_lum = 0;
media_lum = 0;
soma_lum = 0;

%temperatura
min_temp = 4000;
max_temp = 0;
media_temp = 0;
soma_temp = 0;

%umidade
min_umid = 4000;
max_umid = 0;
media_umid = 0;
soma_umid = 0;
```

```
%Verifica se a opção escolhida no menu é "Menu"
switch cell2mat(handles.escolha)
case 'Menu'
   menu = 0;
end

%Verifica se o usuário digitou um tempo de monitoramento
if isempty(get(handles.edit_tempo,'String'))
   errordlg('Preencha o tempo de monitoramento','Erro');
elseif(menu==0)
   errordlg('Escolha um item do Menu','Erro');

% Inicio da execução do monitoramento
else
   for k=1:1:handles.tempo

      set(hObject,'BackgroundColor','green');

      % Luminosidade
      lum=a.analogRead(0)*(5/1023);
      ldr=((5-lum)/lum);
      lux=255.84*(ldr^(-10/9));
      luminosidade=[luminosidade,lux];
      axes(handles.axes1);
      plot(luminosidade,'r');
      grid on;
      axis([0 handles.tempo 0 700]); %eixos primeiro X, depois Y
      xlabel('Tempo');
      ylabel('Luminosidade (Lux)');

      %Temperatura
      temp=((a.analogRead(1))*5/1023)/0.01;
      temperatura=[temperatura,temp];
      axes(handles.axes2);
      plot(temperatura);
      grid on;
      axis([0 handles.tempo -15 50]); %eixos primeiro X, depois Y
      xlabel('Tempo');
      ylabel('Temperatura (ºC)');

      % Umidade
      umid=((a.analogRead(2))*5/1023)/0.019452;
      umidade=[umidade,umid];
      axes(handles.axes3);
```

```
plot(umidade,'g');
grid on;
axis([0 handles.tempo 0 100]); %eixos primeiro X, depois Y
xlabel('Tempo');
ylabel('Umidade (%)');

%Verifica a escolha e os parametros para acionamento ou
%desacionamento dos atuadores
switch cell2mat(handles.escolha)
case 'Fruta1'
    if (c_lum > 2000)
        a.digitalWrite(2,1);
    else
        a.digitalWrite(2,0);
    end
    if (temp <= 10)
        a.digitalWrite(4,1);
    else
        a.digitalWrite(4,0);
    end
    if (temp >= 15)
        a.digitalWrite(7,1);
    else
        a.digitalWrite(7,0);
    end
    if (umid <= 50)
        a.digitalWrite(8,1);
    else
        a.digitalWrite(8,0);
    end
case 'Fruta2'
    if (c_lum > 1500)
        a.digitalWrite(2,1);
    else
        a.digitalWrite(2,0);
    end
    if (temp <= 15)
        a.digitalWrite(4,1);
    else
        a.digitalWrite(4,0);
    end
    if (temp >= 20)
        a.digitalWrite(7,1);
    else
```

```
            a.digitalWrite(7,0);
        end
        if (umid <= 40)
            a.digitalWrite(8,1);
        else
            a.digitalWrite(8,0);
        end
    case 'Fruta3'
        if (c_lum > 2000)
            a.digitalWrite(2,1);
        else
            a.digitalWrite(2,0);
        end
        if (temp <= 25)
            a.digitalWrite(4,1);
        else
            a.digitalWrite(4,0);
        end
        if (temp >= 30)
            a.digitalWrite(7,1);
        else
            a.digitalWrite(7,0);
        end
        if (umid <= 60)
            a.digitalWrite(8,1);
        else
            a.digitalWrite(8,0);
        end
    end
    pause(1);
end

%Calculos de media, minimo e máximo

%luminosidade
for i=2:1:length(luminosidade)
    soma_lum = soma_lum + luminosidade(i);
    if(luminosidade(i) > max_lum)
        max_lum = luminosidade(i);
    end
    if(luminosidade(i) < min_lum)
        min_lum = luminosidade(i);
    end
end
```

```
    media_lum = soma_lum / (length(luminosidade)-1);
    set(handles.media_lum,'String',media_lum);
    set(handles.max_lum,'String',max_lum);
    set(handles.min_lum,'String',min_lum);

    %temperatura
    for i=2:1:length(temperatura)
        soma_temp = soma_temp + temperatura(i);
        if(temperatura(i) > max_temp)
            max_temp = temperatura(i);
        end
        if(temperatura(i) < min_temp)
            min_temp = temperatura(i);
        end
    end
    media_temp = soma_temp / (length(temperatura)-1);
    set(handles.media_temp,'String',media_temp);
    set(handles.max_temp,'String',max_temp);
    set(handles.min_temp,'String',min_temp);

    %umidade
    for i=2:1:length(umidade)
        soma_umid = soma_umid + umidade(i);
        if(umidade(i) > max_umid)
            max_umid = umidade(i);
        end
        if(umidade(i) < min_umid)
            min_umid = umidade(i);
        end
    end
    media_umid = soma_umid / (length(umidade)-1);
    set(handles.media_umid,'String',media_umid);
    set(handles.max_umid,'String',max_umid);
    set(handles.min_umid,'String',min_umid);

    %salva as leituras, media, minimo e maximo de cada sensor
    save(['luminosidade_' num2str(1)],'luminosidade','media_lum','min_lum','max_lum','-
ASCII');
    save(['temperatura_' num2str(1)],'temperatura','media_temp','min_temp','max_temp','-
ASCII');
    save(['umidade_' num2str(1)],'umidade','media_umid','min_umid','max_umid','-ASCII');

    %Envia sinal alto para o buzzer, para indicar o fim do monitoramento
    for i=1:1:1
```

```
        a.digitalWrite(13,1);
        set(hObject,'BackgroundColor','red');
        pause(1);
    end

    set(hObject,'BackgroundColor','white');

    %Envia sinal baixo para todos os atuadores
    a.digitalWrite(2,0);
    a.digitalWrite(4,0);
    a.digitalWrite(7,0);
    a.digitalWrite(8,0);
    a.digitalWrite(13,0);

end

function edit_tempo_Callback(hObject, eventdata, handles)
data1=get(hObject,'String');
handles.tempo=str2double(data1);

switch cell2mat(handles.mtempo)
case 'Segundos'
    handles.tempo = handles.tempo;
case 'Minutos'
    handles.tempo = handles.tempo*60;
case 'Horas'
    handles.tempo = handles.tempo*3600;
case 'Dias'
    handles.tempo = handles.tempo*86400;
end

guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function edit_tempo_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```