GUILHERME HASS BUENO

AUTOMATIZAÇÃO DE UMA ESTUFA AGRÍCOLA ATRAVÉS DO DESENVOLVIMENTO DE UM *SOFTWARE* PARA MONITORAMENTO E CONTROLE.

LAVRAS – MG

2020

GUILHERME HASS BUENO

AUTOMATIZAÇÃO DE UMA ESTUFA AGRÍCOLA ATRAVÉS DO DESENVOLVIMENTO DE UM *SOFTWARE* PARA MONITORAMENTO E CONTROLE.

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Lavras, como parte das exigências do curso de Engenharia de Controle e Automação, para obtenção do título de Bacharel.

Professor Dr. Fábio Domingues de Jesus

Orientador

LAVRAS – MG

2020

GUILHERME HASS BUENO

AUTOMATIZAÇÃO DE UMA ESTUFA AGRÍCOLA ATRAVÉS DO DESENVOLVIMENTO DE UM *SOFTWARE* PARA MONITORAMENTO E CONTROLE.

AUTOMATION OF AN AGRICULTURAL GREENHOUSE TROUGH THE DEVELOPMENT OF A *SOFTWARE* FOR MONITORING AND CONTROL.

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia de Controle e Automação, para a obtenção do título de Bacharel.

APROVADA EM 06 DE JULHO DE 2020.

Dr. Belisário Nina Huallpa- UFLA

Eng. Ana Cristina Porto Silveira

Professor Dr. Fábio Domingues de Jesus

Orientador

LAVRAS – MG

2020

Dedico mais esta conquista ao meu pai Sergio José Bueno e minha mãe Maria Cecília Von Zuben Hass Bueno, por todo o esforço que fizeram para essa conquista.

AGRADECIMENTOS

Gostaria de agradecer primeiramente aos meus pais e a minha família, pelo apoio incondicional e incentivo todos esses anos, que só assim me fizeram chegar até aqui, superando diversas barreiras.

Aos meus amigos que fiz nessa caminhada, que estiveram comigo nos momentos de diversão, de alegria, mas também nos momentos de dificuldade, podendo superar e passar por esses obstáculos juntos.

Ao meu orientador Fábio, por fazer parte do encerramento desse ciclo, estando comigo nesse final me auxiliando.

Aos funcionários da Universidade Federal de Lavras, que estiveram sempre ali, as vezes não diretamente, mas por traz me proporcionando um melhor dia a dia.

Ao time de logística da John Deere C&F que me apoiou com a ideia.

Foram dias de muito aprendizado e luta. A todo vocês, meu muito obrigado!!

“Antes de dormir todos nós pensamos, idealizamos, imaginamos, criamos. A diferença de você e o resto consiste em ao abrir os olhos, nada se apagará, nada se dissipará... você pegará uma caneta e planejará seu caminho até seu sonho. E quando outros falarem que será impossível, transforme esse comentário em combustível para sua mente."

RESUMO

A automatização das atividades e processos proporciona uma melhoria na qualidade dos produtos, gerando uma vantagem competitiva, devido a se trabalhar com uma precisão muito maior. Dessa maneira, investimentos em tecnologia para qualquer segmento de mercado, sempre proporcionará melhorias de processo, pois informações mais precisas e concretas serão retiradas para análise, e a partir disso, decisões serão tomadas com uma base de dados muito mais exata, diminuindo erros. Em termos de expansão da tecnologia na agricultura encontra-se a falta de opções de sistemas automatizados para pequenos agricultores. Por isso, esse trabalho visa o desenvolvimento de um *software*/protótipo para o monitoramento e controle de variáveis importantes para o cultivo. Por meio de entrevistas com pessoas do segmento, foi verificado quais seriam essas variáveis de maior relevância, sendo estas: temperatura, umidade e o pH. O controle controle da bomba de irrigação também é algo importante a ser controlado e monitorado. O protótipo foi avaliado como útil e de fácil utilização pelos usuários.

Palavras-Chave: Automatização de estufas. Desenvolvimento de *softwares*. Otimização de processos.

ABSTRACT

The automation of activities and processes provides an improvement in the quality of products, generating a competitive advantage, due to working with much greater precision. In this way, investments in technology for any market segment, always offer process improvements, more precise and concrete information is removed for analysis, and from that, decisions are applied based on much more accurate data, reducing errors. For this reason, this work aims to develop a software / prototype for the monitoring and control of important variables for cultivation. Through interviews with people in the segment, we saw which variables were most relevant. They were said to be of greater importance: temperature, humidity, pH and control of the irrigation pump. The prototype was evaluated as useful and easy for users to use.

Keywords: Automation of greenhouses. *Software* development. Process optimization.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

A maneira de se cultivar através de cultivos protegidos é algo muito antigo na humanidade. Em países com invernos muito rigorosos, viu-se a necessidade de criar uma maneira para o cultivo de um vegetal todos os períodos e estações do ano. Isso sem a criação do ambiente controlado não seria possível, pois as condições de temperatura não tornariam isso possível.

Mello (2013) afirma que a primeira estufa, nesta época chamada de "Casa de Vegetação", foi criada na Itália, no século XIII, se espalhando para Coreia e outros países da Europa no decorrer dos séculos XV a XIX . Claro que essas estufas não possuíam a tecnologia de controle e monitoramento que vemos nos dias atuais, porém permitiam que os vegetais fossem cultivados durante todas as estações do ano, o que era seu principal objetivo.

Hoje em dia, altas tecnologias são empregadas nas estufas agrícolas, com finalidade de monitorar e controlar suas variáveis, buscando sempre o melhor clima para o desenvolvimento das plantas ali presentes dentro dela.

Diante deste cenário apresentado, e do contexto que vivemos em nosso país, onde a agricultura é o carro chefe da economia, vemos que esse segmento carece de tecnologia, esse trabalho visa aprimorar o controle de uma estufa, mostrando as vantagens da tecnologia nessa área, além da criação de um protótipo para realização do monitoramento. A competição também é algo que vem crescendo em todos os segmentos, com a facilidade ao acesso à informação, sabe-se que quem não se atualiza pode ficar obsoleto no mercado.

Figura 1 – Agricultura de precisão.

Fonte: Blog Belagro.

1.2 Objetivos

Este trabalho de conclusão de curso teve como objetivo o desenvolvimento do protótipo físico de uma estufa agrícola e de um *software*, nomeado *"GHouse LG 3000"*, para monitoramento e controle de variáveis tais como: temperatura, umidade e o pH. E também para o acionamento da bomba de irrigação.

1.3 Estrutura do Trabalho

O trabalho está estruturado em 5 capítulos, conforme se segue:

- Capítulo 1: Introdução: são apresentados as motivações e objetivos do trabalho.
- Capítulo 2: Fundamentação teórica: apresentado o referencial teórico necessário para o entendimento do trabalho.
- Capítulo 3: Desenvolvimento: são apresentados todos os tipos de materiais desenvolvidos na realização do estudo, das ferramentas que foram utilizadas e todos os processos.
- Capitulo 4: Resultados e Discussão: apresentação do resultado que foi obtido com o projeto.
- Capitulo 5: Conclusão: verificação se os objetivos propostos foram alcançados.

2 REFERENCIAL TEÓRICO

Este capítulo tem como objetivo apresentar como a agricultura de precisão vem ganhando mercado, e quão fundamental ela é nos dias de hoje. Também, mostrar como a automação dessas atividades é fundamental para a otimização da produção.

2.1 Linguagem JAVA

Conforme Claro e Sobral (2008), JAVA é a linguagem de programação orientada a objetos desenvolvida em 1991 pela *Sun Microsystems*, tendo sua primeira versão lançada em 1996, capaz de criar desde aplicativos para desktop e *Web* até *softwares* robustos. Um sofware é denominado robusto quando realiza suas tarefas de forma correta em condições anormais, isto é, é capaz de lidar com erros durante a execução e lidar com entradas errôneas.

JAVA é uma linguagem bastante conveniente para o desenvolvimento de *softwares* que funcionem junto com a Internet, além de ser multiplataformas, podendo executar seus programas no *Microsoft Windows*, *Apple Macintosh* ou outro sistema operacional de forma igual em todos eles. Comparando-a com outras linguagens mais conhecidas, se assemelha à *C++* se forem desconsideradas suas características mais complexas, como, por exemplo, herança múltipla e ponteiro.

Atualmente a linguagem JAVA está presente em todas as áreas e mercados, sendo executada pelos dispositivos domésticos, como previsto no final do século XX e também por aparelhos portáteis e de uso pessoal, como smartphones e relógios de pulso.

2.1.1 Programação Orientada a Objetos

A Programação Orientada a Objetos, ou *OOP*, é uma forma de desenvolver um programa de computador como um conjunto de objetos que interagem entre si.

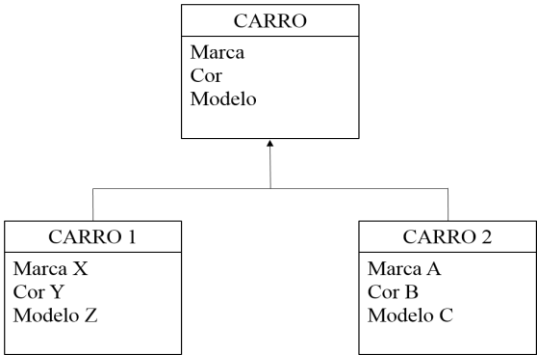
O desenvolvimento da linguagem *OOP* ocorre em módulos, ou blocos de códigos, correspondentes aos objetos e seus acoplamentos. Por meio da orientação a objetos obtém-se uma maior qualidade e agilidade no desenvolvimento do código partindo do princípio da reutilização do objeto, ou seja, um mesmo objeto que fora usado anteriormente pode ser útil para o desenvolvimento de um programa atual e pode ser incorporado ao mesmo. Desta forma garante-se também uma manuseabilidade e uma garantia maior do sistema, uma vez que já foram testados e executados previamente.

2.1.1.1 Objetos e Classes

São dois os nomes mais utilizados quando se trata de uma *OOP*, objeto e classe. Objetos possuem características que o torna único, o diferencia dos outros. A classe, por sua vez, é o agrupamento de objetos que contêm as mesmas características, ou em programação, chamados atributos, e os mesmos comportamentos, de acordo com Claro e Sobral (2008).

Para melhor elucidar estes dois conceitos, pode-se utilizar, por exemplo um carro, ilustrado na Figura 2. Carros possuem características em comum, como marca, cor, modelo, logo, pode-se criar uma classe "Carro". Entretanto, os carros se diferenciam entre si por causa das características supracitadas e, por isso, "Carro 1" e Carro 2" devem ser analisados como objetos que compõem a classe Carro.

Figura 2 – Classe e objeto.



Fonte: Do autor (2020).

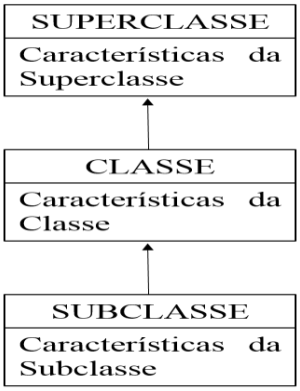
Cada objeto pode se comunicar entre si, por meio da troca de mensagens. Esta troca pode ser entendida como uma aplicação de métodos em alguns objetos. Ainda utilizando do exemplo citado anteriormente, quando o objeto "Carro 1" deseja que o objeto "Carro 2" execute um de seus métodos, ele envia uma mensagem a este. Assim, métodos são os procedimentos invocados quando um objeto recebe uma mensagem.

2.1.1.2 Encapsulamento das informações

O encapsulamento das informações é uma ferramenta muito utilizada quando se trata de *OOP*, pois permite ocultar informações irrelevantes para objetos que interagem com outros objetos. Nesta operação, os atributos de determinada classe serão modificados somente quando métodos que interajam com eles foram usados. Desta forma, existe a garantia de que não haverá uma manipulação indevida dos atributos garantida pela modificação por meio dos métodos.

2.1.1.3 Herança

Figura 3 – Herança



Fonte: Do autor (2020).

A Herença em Java, assim como no mundo real, representa a transmissão dos atributos e métodos a classes subordinadas. Pela figura 3 conseguimos observar isso. Um exemplo prático seria uma classe “Animal”, com atributos como “tamanho”, “peso”, e métodos como “correr” e “caçar”. Todos os objetos da classe “Animal” possuirão esses atributos e métodos. Agora se criamos uma classe “Mamífero” que vai herdar da classe “Animal”, ela possuirá seus próprios métodos e atributos, contudo também terá os da classe “Animal”, devido a “Mamífero” ser uma classe que herda de “Animal”. “Animal” é uma “Superclasse” de “Mamífero”. É importante ressaltar que não é possível o acesso de uma classe às características de sua subordinada, somente o contrário. Ou seja, “Mamífero” possui os atribustos e métodos de “Animal”. Entretanto “Animal” não possui os de “Mamífero”.

2.1.1.4 Interface

O conceito de interface é bastante utilizado nas implementações das interfaces gráficas no JAVA. Uma interface é, a comunicação entre o operador, ou meio externo ao programa, que pode conter, ou não, a transmissão dos dados para o programa.

Desta forma, basta que o operador do sistema saiba o que o objeto é capaz de fazer e não como ele agirá, utilizando, assim, o conceito de encapsulamento, já que somente as assinaturas dos métodos públicos são exibidas ao usuário.

2.1.2 Orientação a Objetos em JAVA

JAVA é considerada uma linguagem orientada a objetos embora não como um todo por causa de seus tipos primitivos de dados que não podem ser considerados classes tampouco objetos, mas que facilitam a manipulação dos dados pelos programadores. Alguns conceitos de

OOP são bastante difundidos nesta linguagem, como a ocultação de dados, o encapsulamento e a facilidade de manutenção.

Por meio do ocultamento de dados, problemas gerados pelo uso das variáveis globais podem ser identificados facilmente. Atribui-se restrições de acesso aos atributos e métodos do programa, então é inserido sobre o mesmo um controle sobre o que e quem pode acessar determinados métodos e atributos. Assim, em JAVA, os modificadores de acesso, como "*public"*, "*private"* e "*protected"* realizam a tarefa de ocultar os dados.

O encapsulamento se relaciona profundamente com a ocultação dos dados. Caso um método fora de uma determinada classe queira manipular um atributo, é preciso que esse chame um método definido internamente à classe que contém o atributo. Encapsulamento, então, é esta ocultação das informações e a manipulação por meio dos métodos. Em JAVA, o encapsulamento se dá por meio dos comandos "*get*" e "*set*", que são métodos que informam e modificam o valor de um atributo, respectivamente.

A facilidade de manutenção de uma *OOP* está relacionada ao fato de que os sistemas precisam constantemente de atualizações e as linguagens possuem a obrigação de fornecer os recursos para que os programas sejam modificados de acordo com as necessidades do meio. A ocultação dos dados torna o código mais simples e o encapsulamento permite uma melhor visualização da relação entre os dados e as ações do programa.

2.2 Agricultura de precisão

O Ministério da Agricultura, Pecuária e Abastecimento (Mapa) (2012) define a Agricultura de Precisão, ou AP, como um sistema de gerenciamento agrícola baseado na variação espacial e temporal da unidade produtiva e visa ao aumento de retorno econômico, à sustentabilidade e à minimização do efeito ao ambiente. Para a Embrapa (Empresa Brasileira de Pesquisa Agropecuária) (2014) é uma postura gerencial que leva em consideração a variabilidade espacial da lavoura para a obtenção de retorno econômico e ambiental. Já Bramley (2009) acrescenta a estas definições um termo, de que para ele, AP é um conjunto de tecnologias que promovem a melhoria na gestão dos sistemas de produção com base no reconhecimento de que o potencial de resposta das lavouras pode variar desde mesmo que em pequenas distâncias.

Assim sabendo o que, onde e quando fornecer, os resultados vão aumentar e os custos diminuir, evitando disperdicios.

O termo precisão, na visão de Molin, Amaral e Colaço (2015) pode ser contestado, uma vez que a palavra se refere ao grau de aproximação da grandeza mensurada ao valor verdadeiro. Entretanto, o termo a ser usado deveria ser exatidão, já que se busca uma agricultura com uma exatidão maior do que aquela praticada atualmente. Sendo assim, a automação proporciona uma maior exatidão ao processo.

As tecnologias utilizadas na agricultura de precisão são empregadas atualmente no campo. O conhecimento de que existe uma variedade nas áreas de produção gerada pelos diferentes relevos, solos, vegetação, dentre outros, está sendo progressivamente disseminado. Este conhecimento é útil para diversas culturas, sejam de pequenas ou grandes áreas e as diferenças existentes exigem tratamentos específicos para cada região, respeitando suas particularidades.

Percebe-se, na sociedade, uma expectativa muito grande de que as máquinas e equipamentos caros e sofisticados realizem os trabalhos autonomamente quando se trata de agricultura de precisão.

Uma aplicação da agricultura de precisão que não se dá com sucesso compromete a imagem e reputação das indústrias e prestadoras de serviços a ela relacionadas. Porém, o bom êxito desta prática traz bons resultados e gera um processo agrícola mais responsável e racional, uma vez que evita-se disperdícios devido ao maior controle da produção, além de demandar de uma qualificação da mão de obra e, consequentemente, apresentar-se de forma mais sustentável.

2.3 Cultivos em Estufas

O cultivo em estufa é uma ramificação dos cultivos protegidos, e têm como objetivo minimizar o efeito das intempéries climáticas, tais como excesso, ou falta, de chuvas, temperaturas muito elevadas ou baixas ao extremo. Estes fatores prejudicam tanto a qualidade do produto quanto o rendimento da produção acarretando em uma diminuição da rentabilidade da mesma. Esse cultivo já se encontra em uso na produção de mudas e iniciando no setor dos hortifrútis. Para as frutas, o emprego de mais destaque, atualmente, é o da produção de morangos no sul do Brasil.

Silva, Silva e Pagiuca (2014) caracterizam um cultivo protegido como uma técnica que possibilita o controle das variáveis supracitadas. Tal controle se traduz em ganho de eficiência e redução do efeito de sazonalidade, permitindo a oferta do produto durante épocas do ano que não a da colheita normalmente.

Cultivar em ambiente protegido é mais do que simplesmente oferecer cobertura às plantas. Para que a cultura se desenvolva de modo a ter uma alta produção, poucas perdas e uma melhor qualidade, ou seja, uma alta eficiência é preciso que se conheça as necessidades de cada espécie e suas particularidades e neste momento o produtor, pode contar com o auxílio da automação para manter as variáveis controladas.

Ao iniciar o cultivo em uma estufa é imprescindível ter um planejamento não só no que se diz respeito aos custos e benefícios na instalação das estruturas. É preciso se atentar a adequação das tecnologias ao ambiente físico, como a forma de construção, posicionamento em relação aos ventos e ao sol, o inundamento do solo quando em época de chuvas, a legislação ambiental e, principalmente, os benefícios que se almeja ao iniciar o cultivo.

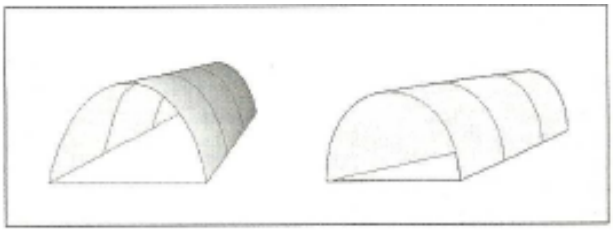
2.3.1 Tipos de Estufas

São várias as configurações das estufas de modo a oferecer uma melhor adaptação da plantação e um maior aproveitamento das características do local onde a estufa se encontra. É necessária cautela quando se for determinar a estufa que mais se encaixa ao meio e à disponibilidade financeira.

2.3.1.1 Estufas do Tipo Túnel

Este tipo de estufa, como da Figura 4, não é recomendado para climas quentes, úmidos ou secos, e tem como principais vantagens a sua construção simples, além de ser resistente a ventos fortes. Entretanto, possuem pouca ventilação se desprovidas de aberturas laterais, um desaproveitamento dos espaços próximos às janelas e baixa relação entre a área coberta e o volume.

Figura 4 – Estufa do tipo túnel

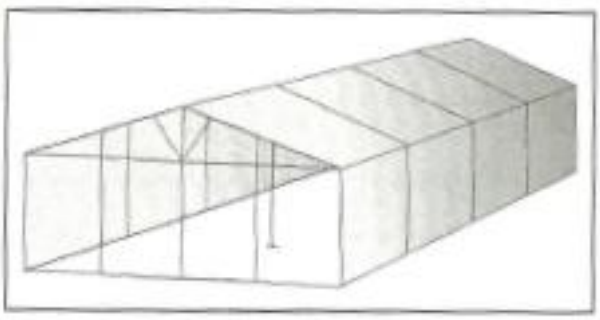


Fonte: Prof. Francisco Hevilásio Pereira

2.3.1.2 Estufas do Tipo Capela

Na configuração da Figura 5, a estufa possui telhados planos e retos, lembrando, como o próprio nome indica uma capela, podendo ser construída com materiais já existentes no local, como bambu e eucalipto. É adequada para locais que possuem grandes oscilações de temperaturas, porque seu telhado em pelo menos 30º impede a queda de água de condensação sobre as plantas.

Figura 5 – Estufa do tipo capela.

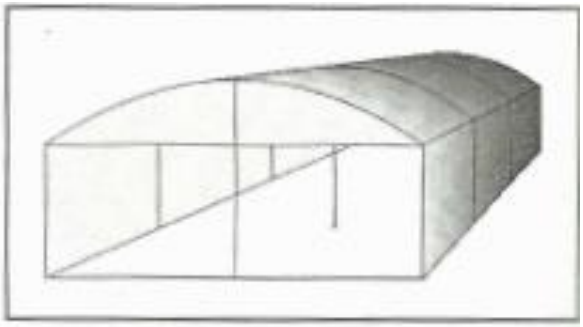


Fonte: Prof. Francisco Hevilásio Pereira

2.3.1.3 Estufas com Teto em Arco

O tipo de estufa mais comum no Brasil, ilustrado pela Figura 6, apresenta uma cobertura elíptica de fácil construção. Devido à sua configuração, possui grande aproveitamento dos raios solares e, apesar de apresentar um baixo custo de manutenção, o mesmo não ocorre para a sua implantação.

Figura 6 – Estufa com teto em arco.



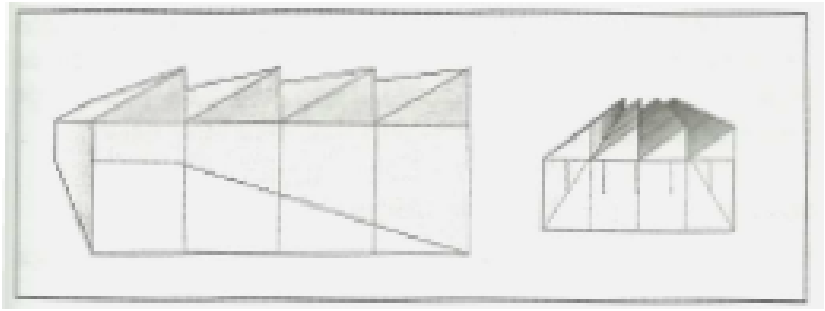
Fonte: Prof. Francisco Hevilásio Pereira

2.3.1.4 Estufas do Tipo Dente de Serra

Na estufa da Figura 7, a característica principal é o desenho da cobertura, semelhante aos dentes de uma serra, podendo ser de madeira ou metálica. Quando se trata de ventilação, se apresenta de forma eficiente, pois a instalação da cobertura deve ser no sentido dos ventos proeminentes com os dentes para o lado contrário, mas o mesmo não se pode dizer do

aproveitamento da luz solar. Devido à boa ventilação, é indicada para regiões quentes, sejam elas úmidas ou secas.

Figura 7– Estufa do tipo dente de serra.



Fonte: Prof. Francisco Hevilásio Pereira

2.4 Otimização de Processos

De acordo com Moreira (2009), um processo produtivo é a combinação de fatores que visam a obtenção de um produto final. Em processos produtivos, para se ter um produto final é necessária a incorporação destes fatores para que haja a sua transformação no que se deseja.

Lemos (2015) afirma que a organização dos processos é indispensável. Faz-se necessário o mapeamento dos processos, uma proposta de mudanças baseadas na análise prévia e o controle das mudanças realizadas. Processos, ainda segundo Lemos (2015) são formas de sintetizar as atividades de trabalho em organizações administrativas. São eles que indicam as direções, as ações e rotinas em que se deve organizar o trabalho de forma a se adaptar às necessidades dos clientes e novas tecnologias e inovações, sendo fundamentais para o bom funcionamento das instituições administrativas.

Quando se fala em processos, dois aspectos devem ser considerados, a eficiência e a eficácia. Para Drucker (2002) a eficiência está relacionada ao nível operacional e ao bom uso dos recursos das empresas, ou seja, fazer mais com menos, enquanto a eficácia se relaciona com o nível gerencial, com a conclusão dos objetivos propostos pela organização.

A busca pela maior eficiência em conjunto com a maior eficácia é o que se pode chamar otimização de um processo. Uma vez que se consegue aliar estes dois fatores, o setor em questão conquista o melhor uso possível dos recursos que necessita para a realização da tarefa e, assim, atinge suas metas, obtendo um lucro maior com o menor descarte possível de insumos, maximizando o faturamento e a produtividade consequentemente.

A otimização dos processos produtivos, quando executada de forma eficaz, eficiente e adequada, melhora a capacidade de gerir cada etapa produtiva, podendo antecipar e responder às mudanças do mercado e aumentar as oportunidades que possam surgir. Uma gestão voltada aos processos, segundo Costa (2008), é uma ferramenta que permite compreender como são criados os produtos, ou serviços, nas empresas e identifica prováveis ineficiências e problemas que normalmente seriam dificilmente encontrados.

2.5 Arduino

O Arduino é um microcontrolador baseado no ATmega328 de baixo custo para aquisição(ARDUINO,2020), por isso é muito utilizado na construção de protótipos e até em maiores dimensões. É uma plataforma de prototipagem eletrônica livre, baseada em uma interface de fácil uso.

OArduino é responsável pela comunicação entre sensores e motores por meio da implementação do código feito na plataforma da placa e que a alimenta, a fim de gerar o controle da temperatura e estabilizar, assim, o sistema.

A linguagem de programação utilizada pela plataforma é bastante semelhante à C/C++, facilitando o desenvolvimento de programas a serem implementados pela placa (ARDUINO, 2020). A placa Uno em específico possui 6 entradas analógicas, 14 pinos de entrada/ saídas digitais para a comunicação do sistema, uma conexão *USB* para estabelecer junção da placa a um computador com um cabo, uma tomada de força, um cabeçalho *ICSP* e um botão de *reset*.

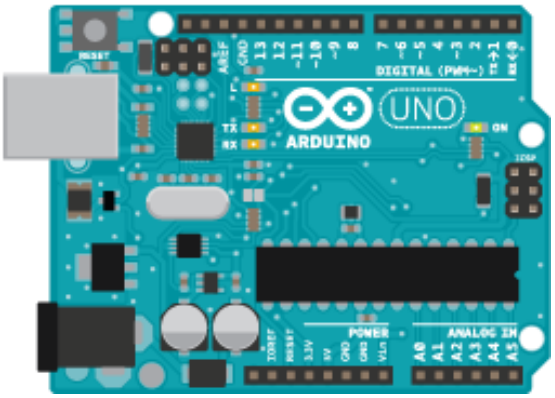
Tabela 1 – Especificações da placa do Arduino Uno

Especificação	Valor no Arduino Uno
Microcontrolador	ATmega328
Tensão de Funcionamento	5V
Tensão recomendada de entrada	7-12V
Tensão limite de entrada	6-20V
Corrente DC por entrada e/ou saída	40mA
Corrente DC no pino 3.3V	50mA
Memória Flash	32kB (ATmega328), dos quais 0,5kB são utilizados pelo carregador de inicialização
SRAM	2kB (ATmega328)
EEPROM	1kB (ATmega328)
Clock	16MHz

Fonte: Arduino (2020)

O modelo utilizado no decorrer deste trabalho é o Arduino Uno, identificado pela Figura 8 e suas especificações se encontram na Tabela 1.

Figura 8 – Arduino Uno.



Fonte: Arduino (2020).

3 MATERIAIS E MÉTODOS

Neste capítulo encontram-se as ferramentas e materiais utilizados no desenvolvimento do protótipo, com o objetivo de informação e familiarização das técnicas utilizadas. Também aqui será descrito o problema, as características dos *softwares* selecionados e os processos empregados para a execução do trabalho.

3.1 Processo

Juntamente com profissionais e professores da área foi possível definir as prioridades dentro do desenvolvimento do *software*, considerando os principais pontos que deveriam ser abordados dentro dele. O início do projeto foi constituído de pesquisas e entrevistas a fim de obter maior conhecimento sobre o segmento em que o *software* seria desenvolvido. Compreender e estudar a linguagem Java foi fundamental para o desenvolvimento do protótipo, além do estudo da programção do Arduino, para que fosse demonstrado o *software* atuando fisicamente. A partir de toda essa prévia, foi dado início ao que hoje chama-se de "*Ghouse-LG3000*".

3.1.1 Etapas do processo de desenvolvimento do *software*

Primeiramente foi desenhado um fluxograma de atividades para o desenvolvimento do *software*, como na Figura 9.

Figura 9 - Etapas do processo de desenvolvimento.

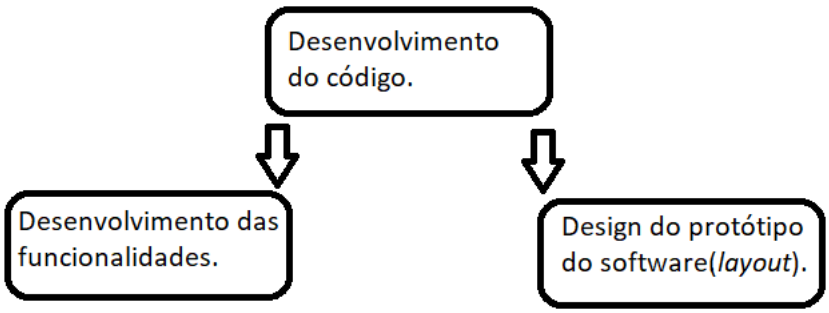
Fonte: Do autor (2020).

O processo inicia-se mapeando todas as informações necessárias para o desenvolvimento do protótipo. Quando se trata de informações, consideram-se os problemas que com ela podem ser resolvidos.

Depois do mapeamento das informações, é feito um processo de filtragem, ou seja, analisa-se viabilidade de cada elemento mapeado. Nesta etapa, define-se o que realmente o *software* deve apresentar e consequentemente fazer o monitoramento e controle.

Assim, se da início ao processo de programação, ou seja, desenvolvimento do código, como mostrado na Figura 9. A Figura 10 abaixo mostra como foi dividida essa etapa, ou seja, primeiramente desenvolveu-se as funcionalidades do *software*. Estando tudo executando, não apresentando erros, seguiu-se para próxima etapa onde foi definido o *layout*.

Figura 10 - Etapas do desenvolvimento do código.



Fonte: Do autor (2020)

Todo o processo de desenvolvimento foi feito na linguagem Java. Houve uma discussão sobre uma maneira de se armazenar os dados obtidos pelos sensores para um *Excel*. Para a sequência desse protótipo, esse seria um próximo passo a ser feito no código, uma conexão com uma planilha do *Excel*, armazenando as informações coletadas durante um certo horizonte de tempo, para futuro do gerenciamento das informações. A escolha do *Excel* como forma de armazenamento foi feita pois é um *software* onde consegue-se armazenar planilhas e além disso criar gráficos do comportamento das variáveis.

Após o desenvolvimento do código, encontra-se o teste feito no protótipo físico da estufa, no qual o usuário será responsável por monitorar, gerenciar e controlar a estufa. Esse gerenciamento será feito através de:

- Controle da temperatura;
- Controle da bomba;

- Visualização da umidade.

A seguir no trabalho serão apresentadas as telas do *software* onde o usuário realizará a operação e gerenciamento.

O usuário terá a visibilidade das informações através da tela principal do *software*, como o tempo da bomba, a temperatura, a umidade, podendo alterar de acordo com a necessidade do momento. O pH também é outra variável de suma importância.

3.2 Descrição do problema

A tecnologia hoje, mesmo que mais acessível, ainda não está presente para todos os lares devido ao custo. Dentro do segmento agrícola ve-se essa carência. Esse protótipo vem para mostrar que é possível criar uma solução fácil e barata para esse segmento do mercado.

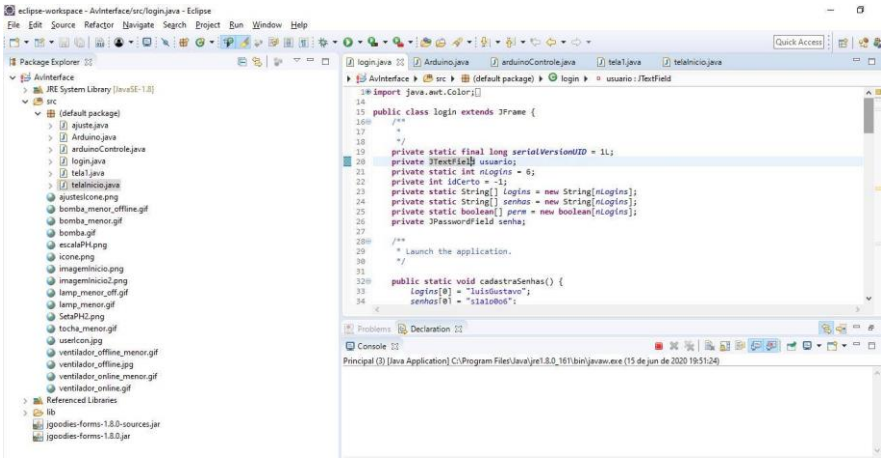
Com o desenvolvimento desse *software*, espera-se uma visualização rápida e simplificada e sem a necessidade de uma aferição manual, algo que é suscetível ao erro, para a análise correta das variáveis da estufa, pois o *software* apresenta simplicidade no *layout* e possibilita satisfazer as necessidades do mercado.

3.3 *Softwares* utilizados

A seguir são apresentados os softwares utilizados para o desenvolvimento do trabalho.

- *Eclipse*: É um ambiente de desenvolvimento Java, entretanto consegue suportar outras linguagens através de plugins. O projeto foi desenvolvido pela empresa americana *IBM*, no ano de 2001. O Eclipse é a *IDE* Java mais utilizada no mundo na atualidade. A função do *Eclipse* no projeto foi desenvolver toda a programação do *software* necessária. A Figura 11 ilustra a tela de desenvolvimento do projeto, mostrando as classes que existem neles.

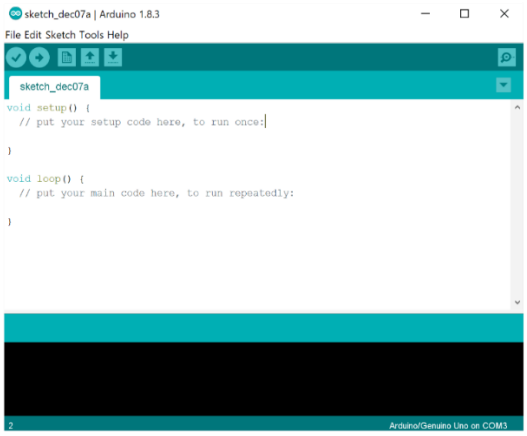
Figura 11 - Tela de desenvolvimento do *software* Eclipse.



Fonte: Do autor (2020).

- *Arduino IDE:* É um ambiente de desenvolvimento em que são criados os códigos, ou seja, é utilizado para escrever e fazer o *upload* para placas compatíveis com o Arduino. Em nosso projeto foi utilizado o Arduino Uno.

Figura 12 - Tela de desenvolvimento do *software* "Arduino IDE".



Fonte: Suporte Microsoft

3.4 Avaliação final

O protótipo do *software* foi avaliado e obteve nota máxima pois atendeu as necessidades técnicas, isto é, apresenta variáveis importantes para o processo. E no que se diz a respeito da sua utilização comercial, também possui um ótimo desempenho. Esse segmento carece muito de inovações, principalmente em nosso país, mesmo tendo a agricultura como principal atividade que movimenta e faz crescer o nosso Produto Interno Bruto (PIB).

A aplicação mostrou-se viável em aplicação prática. O sensor *DHT11* teve um problema técnico que foi resolvido com a troca do equipamento. Feita a troca, todo o sistema operou corretamente.

O *software* obteve um bom desempenho quanto ao tempo de resposta e quanto às informações coletadas. Esse avaliação foi realizada de acordo com a alteração da temperatura e funcionamento dos atuadores.

4 RESULTADOS E DISCUSSÕES

Foi desenvolvido o protótipo de um *software* com finalidade de gerenciar variáveis da estufa, monitorando e controlando as mesmas. O foco desse processo consistiu nas principais variáveis que foram levantadas por professores e estudantes da área.

O protótipo foi desenvolvido utilizando programação na linguagem Java e a programação específica do Arduino, além de toda montagem física da estufa, simulando um caso real.

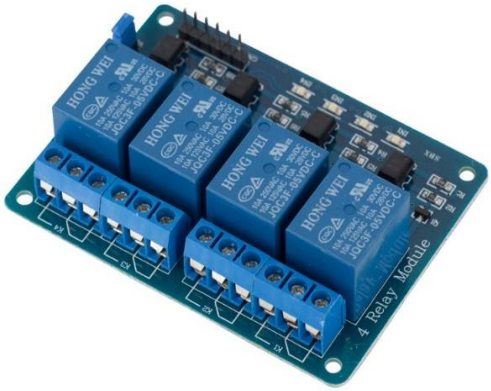
4.1 O protótipo

Além do *software*, também foi criado uma estrutura física, tendo alguns componentes simulando atuadores e sensores.

- Arduino Uno;

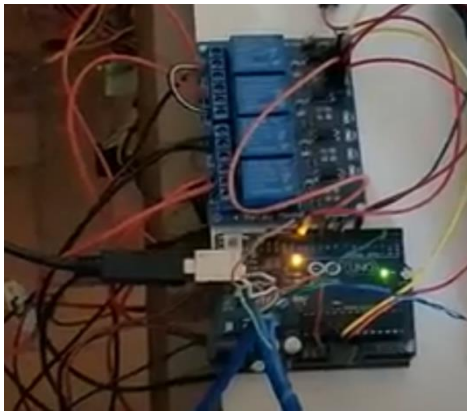
-Módulo Relé 5V 4 canais;

Figura 13 – Módulo relé 4 canais 5V.



Fonte: FilipeFlop

Figura 14 – Arduino Uno conectado ao relé



Fonte: Do autor (2020)

-Ventoinhas de *CPU*;

Figura 15 – Ventoinhas de *CPU* simulando os exaustores.

Fonte: Do autor (2020)

-Lâmpada 10W;

Figura 16 – Protótipo da estufa com a lâmpada para aquecimento acesa.

Fonte: Do autor (2020)

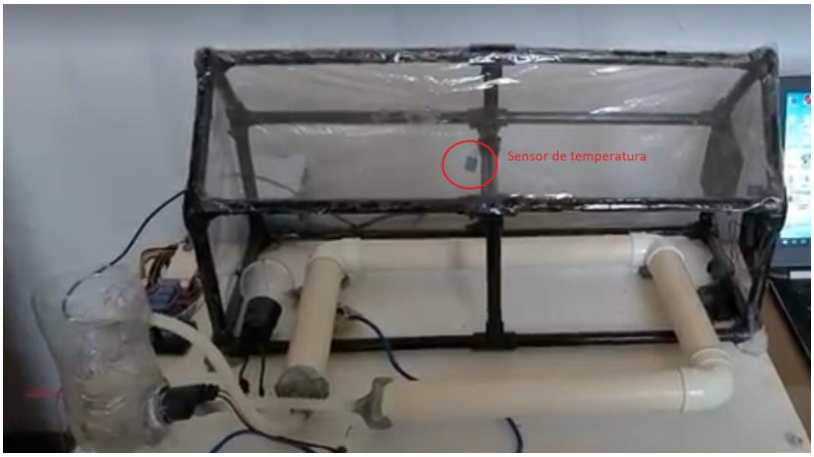
-Bomba de gasolina;

Figura 17 – Bomba de gasolina BOSCH.

Fonte: Do autor (2020)

-DHT11(sensor de temperatura);

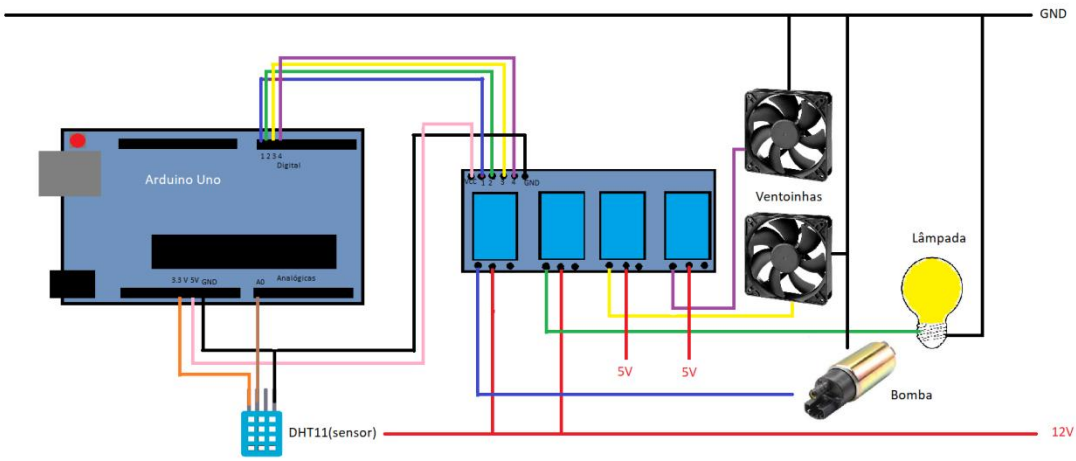
Figura 18 – Sensor de temperatura da estufa.



Fonte: Do autor (2020)

O *software* foi desenvolvido a partir do estudo das variáveis mais importantes dentro de um cultivo protegido, sendo elas: umidade, temperatura e pH. Na tela do *software* é indicado em tempo real essas variáveis para, a partir delas, executar o monitoramento e controle da estufa.

Figura 19 – Esquemático das ligações.



Fonte: Do autor(2020)

4.1.1 Tela de carregamento inicial

Essa é a tela inicial, ao clicar são carregadas as informações iniciais do *software*, com as condições iniciais das variáveis.

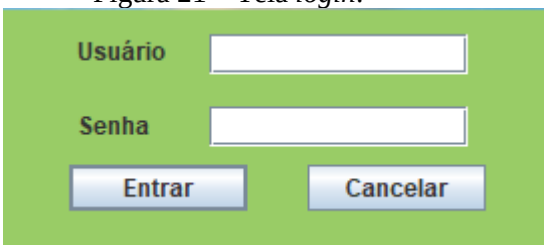
Figura 20 – Tela de carregamento inicial.

Fonte: Do autor (2020)

4.1.2 Tela de login

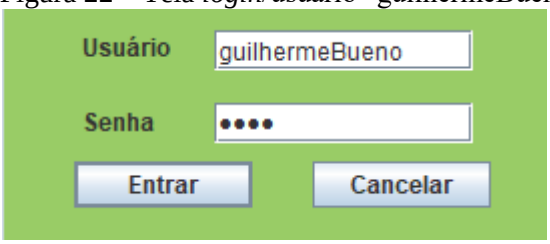
Nessa tela o operador colocará a seu usuário e a senha. Exemplificando mostramos nas figuras abaixo o login do usuário “Guilherme Bueno” e do usuário “João Henrique”. Outra ferramenta presente no acesso são as permissividades. Nas imagens abaixo veremos que o acesso do “João Henrique” é limitado a visualização. Isso foi feito para que nem todos os usuários consigam realizar modificações, sendo que isso poderia gerar problemas, se não feito da maneira correta.

Figura 21 – Tela *login*.



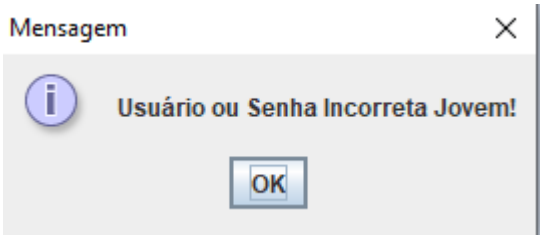
Fonte: Do autor (2020)

Figura 22 – Tela *login*/usuário “guilhermeBueno”



Fonte: Do autor (2020)

Figura 23 – Mensagem de erro no acesso.

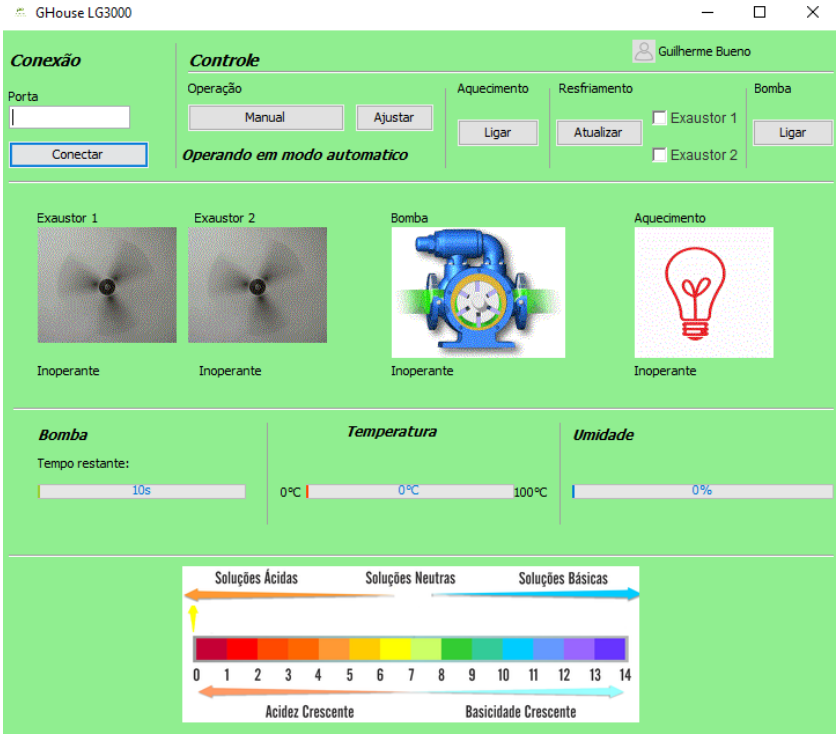


Fonte: Do autor (2020)

4.1.3 Tela principal

Ao realizar o *login*, a tela principal abrirá em seu monitor. O usuário poderá realizar o monitoramento e controle da sua estufa.

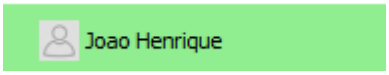
Figura 24 – Tela principal



Fonte: Do autor (2020)

A tela principal, ao efetuar o login, indica qual usuário está utilizando o protótipo. Na Figura 25, no canto superior direito o usuário é “guilhermeBueno”. Se o “*login*” tivesse sido feito pelo “João”, na tela apareceria como na Figura 26.

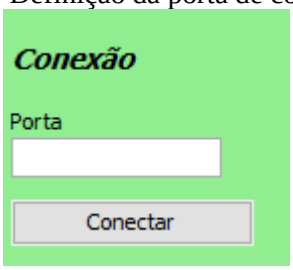
Figura 25 – Acesso “JoaoHenrique”



Fonte: Do autor (2020)

Primeiro passo é realizar a comunicação com o seu computador como ilustrado na Figura 26. Nesse protótipo, a COM4 é a porta responsável pela comunicação.

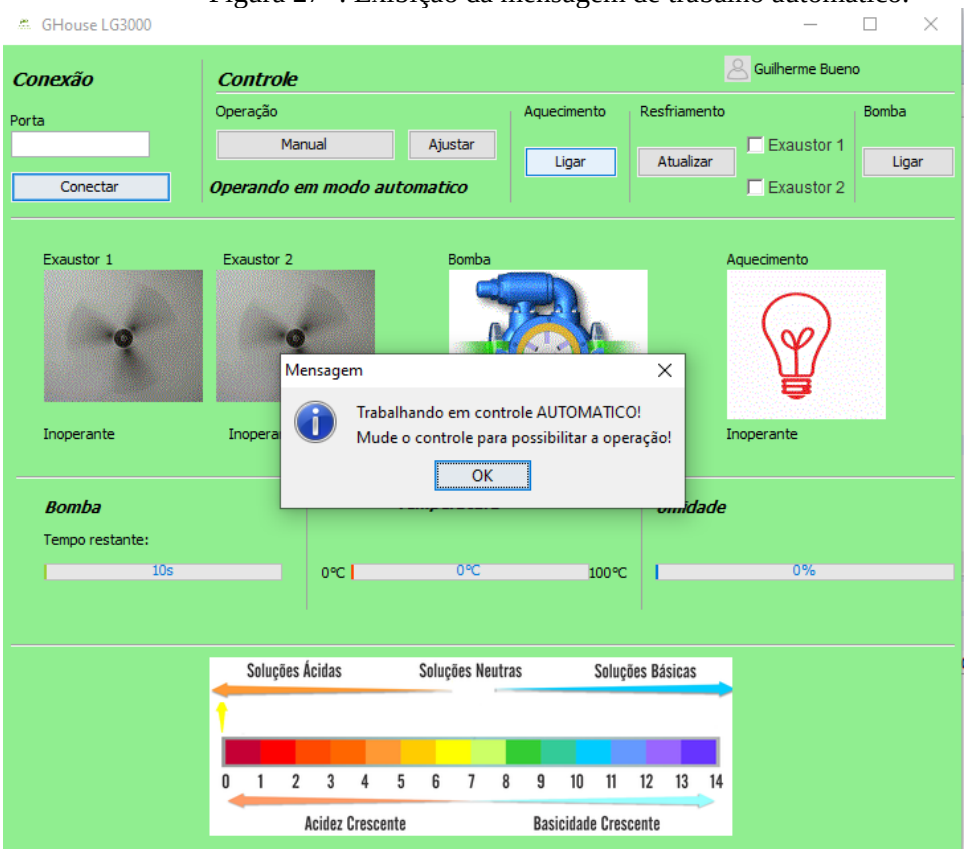
Figura 26 - Definição da porta de conexão



Fonte: Do autor (2020)

Enquanto o modo automático estiver ativado, nenhuma alteração manual poderá ser feita. O *software* exibirá a mensagem da Figura 27 para o operador.

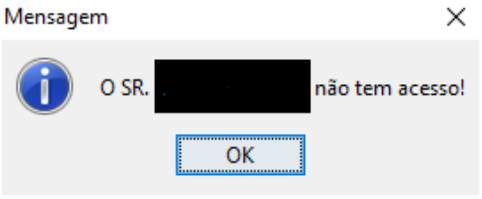
Figura 27 -. Exibição da mensagem de trabalho automático.



Fonte: Do autor (2020)

Somente terão acesso os operadores previamente definidos. Caso o operador seja apenas um visualizador, a Figura 28 aparecerá para ele.

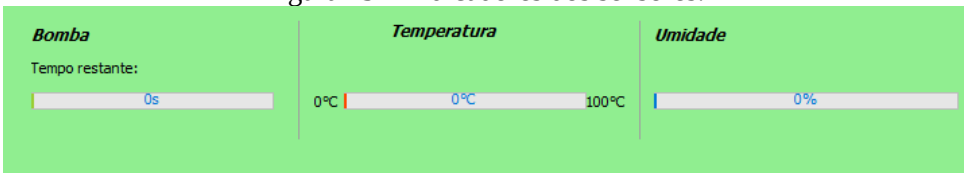
Figura 28 - Mensagem ao operador sem acesso.



Fonte: Do autor (2020)

A Figura 30 mostra os valores em tempo real da temperatura e da umidade dentro da estufa. Ou seja, ao acessar o *software*, esses valores são carregados e mostram para o usuário o estado inicial dessas variáveis. Além do tempo da bomba, caso esteja em modo automático.

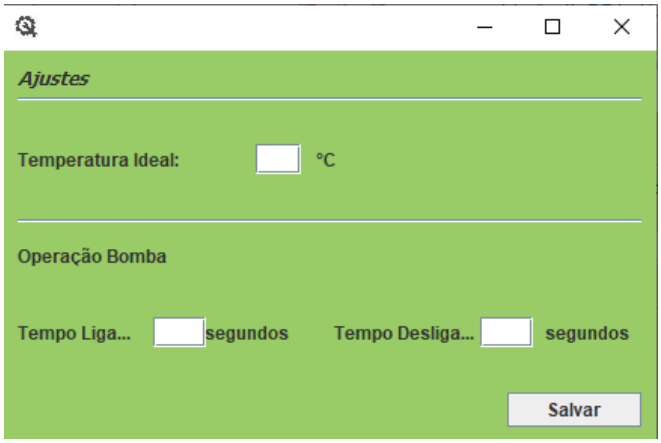
Figura 29 – Indicadores dos sensores.



Fonte: Do autor (2020)

4.1.4 Tela de ajustes

Figura 30 - Tela de ajustes de parâmetros

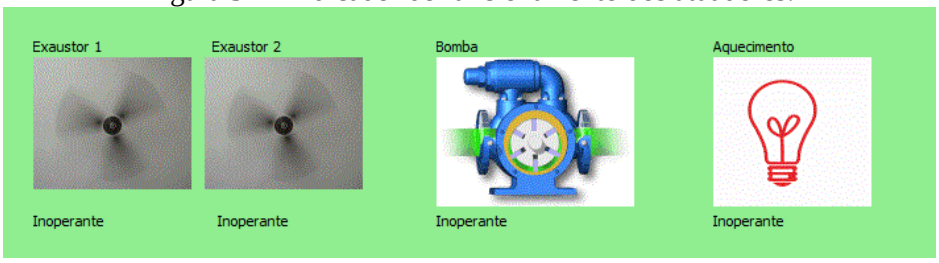


Fonte: Do autor (2020)

Essa tela, da Figura 30, é responsável pelo controle das variáveis quando está operando em modo automático. Definindo a temperatura ideal, as ventoinhas que simulam os exaustores, e a lâmpada responsável pelo aquecimento, trabalharão em sintonia, a fim de manter a temperatura estabelecida pelo operador. As lacunas definidas para “Tempo Ligado” e “Tempo Desligado” são para o funcionamento da bomba. Foram definidos esses tempos, caso a bomba trabalhasse de maneira alternada.

A Figura 31 contém figuras que simulam o funcionamento dos exaustores, lâmpada e bomba. A figura se torna dinâmica em caso de funcionamento. Os exaustores aparecerão rodando. A bomba terá o fluxo ligado. O aquecimento será demonstrado pela lâmpada piscando.

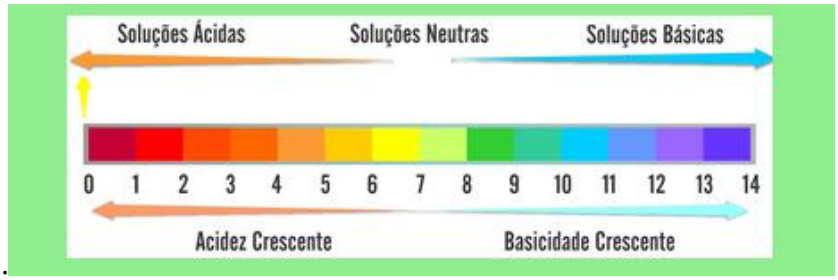
Figura 31 – Indicador de funcionamento dos atuadores.



Fonte: Do autor (2020)

A Figura 32 representa o pH da água. Variável importante dentro de qualquer cultivo na agricultura.

Figura 32 - Indicador de pH



Fonte: Do autor (2020)

4.1.5 Avaliação de discussão do protótipo

O microcontrolador Arduino Uno foi utilizado no projeto possui 6 entradas analógicas tendo um número limitado de conexão de sensores, contudo para o protótipo que apresenta sensores o processamento foi utilzado sem atrasos, modo que , na tela da figura 29 os indicadores mostravam a sua "variação" quase instantaneamente. Também, foi verificado no projeto que mesmo utilizando apenas uma entrada é possível ter alta quantidade de informações. Em relação às 13 entradas/saídas digitais do Arduino Uno verificou-se que o controle e monitoramento de uma estufa agrícola foi suficiente, contudo para uma aplicação prática têm-se a necessidade de analisar o projeto com também o número de entradas/saídas digitais necessárias e o tempo de processamento, mas verifica-se que projeto conceitual apresenta sua viabilidade de apliação para pequenas estufas e de grande escala.

O sensor de umidade e temperatura *DHT11* apresentou problemas de funcionamento, deixando de coletar informações para o software. Devido a isso, teve-se que trocar por um novo equipamento. Este sensor possui uma faixa de umidade relativa de 20 a 80% e uma faixa de temperatura de 0 a 50°C . Para aprimoramentos no desenvolvimentos, melhores sensores estão disponíveis no mercado, como o *DHT22*, com uma resolução melhor que o *DHT11*.

O *software* desenvolvido possui apenas 3,66MB e o seu funcionamento mostrou-se "leve" e de resposta rápida aos comandos um protótipo. Deste modo, o desenvolvimento do software em JAVA apresenta uma grande aplicabilidade para realização da interface homem máquina por ser de fácil programação para profissionais da área de engenharias e computação. Também, foi verificado que a programação para realizar a interação entre o software e o arduino é realizado com poucas linhas de programação apresentando sua facilidade de aplicação. Ao fim do projeto, o protótipo foi levado para avaliação, onde recebeu nota máxima, sendo avaliado quanto ao caráter empreendedor, se seria uma idéia de negócio, e também quanto aos aspectos técnicos.

5 CONCLUSÃO

O desenvolvimento desse protótipo foi orientado para uma forma simples, desenvolvendo um layout amigável para a utilização, podendo resultar futuramente em um projeto comercial atendendo um segmento do mercado agrícola onde não necessite de grandes investimentos.

Todos os equipamentos e dispositivos utilizados funcionaram do modo esperado. O sensor de umidade e temperatura *DHT11* teve que ser trocado devido a um problema técnico. Para um projeto futuro, é possível que haja a troca dos sensores, adquirindo equipamentos com maior qualidade e precisão, além do sensor de pH, tendo um custo maior ao projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

ARDUINO. **Arduino Uno.** Disponível em <https://store.arduino.cc/usa/> Acesso em 23 jun. 2020.

BRANMLEY, R.G.V. **Lessons from nearly 20 years of Precision Agriculture research, development and adoption as a guide to its appropriate application.** Vietnam**:** Crop and Pasture Science, 2009.

CLARO, D.B.; SOBRAL, J.B.M. **Programação em JAVA.** Florianópolis: Copyleft Pearson Education, 2008.

COSTA JUNIOR, E.L. **Gestão em processos produtivos.** Curitiba: Ibpex, 2008.

DUCKER, P. Disponível em <https://administradores.com.br/artigos/entenda-a-diferenca-entre-eficiencia-e-eficacia-de-uma-vez-por-todas> Acesso em 02 jul.2020

EMBRAPA. **Agricultura de Precisão: Resultados de um Novo Olhar**. Brasília, 2014.

LEMOS, G.B. **Otimização de Processos Organizacionais**. Disponível em <https://www.maxwell.vrac.puc-rio.br/29151/29151.PDF> Acesso em 30 jun.2020.

MELLO, S.C. **Introdução ao Cultivo Protegido.** Disponível em <https://www.passeidireto.com/arquivo/38148402/cultivo-protegido-na-olericultura> Acesso em 10 jul. 2020.

MINISTÉRIO DA AGRICULTURA, PECUÁRIA E ABASTECIMENTO. Portaria nº 852 – Art 1º **Criar a Comissão Brasileira de Agricultura de Precisão**. Brasília, 2012.

MOLIN, J.P; AMARAL, L.R.; COLAÇO, A.F. **Agricultura de Precisão.** São Paulo: Oficina de Textos, 2015.

MOREIRA, D.A. **A Administração da Produção e Operações.** São Paulo: Cengage Learning, 2009.

SILVA, B.A.; SILVA, A.R.; PAGIUCA, L.G. **Cultivo protegido em busca de mais eficiência produtiva!.** Disponível em <https://www.cepea.esalq.usp.br/hfbrasil/edicoes/132/mat_capa.pdf>Acesso em 26 jun. 2020.

ANEXO A – Arduino

```
#include <SimpleDHT.h>
//Tabela---------
/* a = aquecimento ligado
b = aquecimento desligado
c = exaustor 1 ligado
d = exaustor 1 desligado
e = exaustor 2 ligado
f = exaustor 2 desligado
g = bomba ligada
h = bomba desligada
# = quando recebido envia temperatura e humidade
*/
SimpleDHT11 dht11;
int pinDHT11 = 2;//sensor
int ex1=6;
int ex2=10;
int lamp=13;
int bomba=8;
byte temperatura = 0;
byte humidade = 0;
void setup()
{
//Start the serial port at 9600 baud rate.
Serial.begin(115200);
//Set pin 13 for output.
pinMode(ex1, OUTPUT);
pinMode(ex2, OUTPUT);
pinMode(lamp, OUTPUT);
```

```
pinMode(bomba, OUTPUT);
digitalWrite(ex1,LOW);
digitalWrite(ex2,LOW);
digitalWrite(lamp,LOW);
digitalWrite(bomba,LOW);
Serial.println("0,0");//inicializa a temperatura na serial para evitar a coleta de dados invalidos no começo da execução
}
long int tempoDHT=millis();
void loop()
{
if((millis()-tempoDHT)>1000)
{
dht11.read(pinDHT11, &temperatura, &humidade, NULL);
tempoDHT=millis();
}
char entrada;
entrada = Serial.read();
switch(entrada){
case 35:
Serial.print((int)temperatura);
Serial.print(",");
Serial.println((int)humidade);
break;
case 97:
digitalWrite(lamp,HIGH);
break;
case 98:
digitalWrite(lamp,LOW);
break;
```

```
case 99:
digitalWrite(ex1,HIGH);
break;
case 100:
digitalWrite(ex1,LOW);
break;
case 101:
digitalWrite(ex2,HIGH);
break;
case 102:
digitalWrite(ex2,LOW);
break;
case 103:
digitalWrite(bomba,HIGH);
break;
case 104:
digitalWrite(bomba,LOW);
break;
```

ANEXO B – Ajuste JAVA

```
import java.awt.EventQueue;

import java.awt.Font;

import java.awt.Image;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.IOException;

import javax.imageio.ImageIO;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JSeparator;

import javax.swing.JTextField;

import javax.swing.border.EmptyBorder;

import java.awt.Color;

public class ajuste extends JFrame {

	private JPanel contentPane;

	private JTextField textTempIdeal;

	private int tempIdeal;

	private int tligado;

	private int tdesligado;

	private boolean s=false;

	private JTextField textTL;
```

```
private JTextField textTD;
public boolean isS() {
	return s;
}public void setS(boolean s) {
	this.s = s;
}public int getTempIdeal() {
	return tempIdeal;
}public void setTempIdeal(int tempIdeal) {
	this.tempIdeal = tempIdeal;
}/**
 * Launch the application.
 */
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
		public void run() {
			try {
				ajuste frame = new ajuste(1,1,1);
				frame.setVisible(true);
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	}
}
```

ANEXO C – Arduino JAVA

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import com.fazecast.jSerialComm.SerialPort;
public class Arduino {
	private static final long serialVersionUID = 1L;
	private static int taxa = 115200;
	private boolean conexao;
	private static SerialPort porta;
	private boolean estadoEx1=false;
	private boolean estadoEx2=false;
	private boolean aquece=false;
	private boolean bomba=false;
	public boolean isEstadoEx1() {
		return estadoEx1;
	}public void setEstadoEx1(boolean estadoEx1) {
		this.estadoEx1 = estadoEx1;
	}public boolean isEstadoEx2() {
		return estadoEx2;
	}public void setEstadoEx2(boolean estadoEx2) {
		this.estadoEx2 = estadoEx2;
	}public boolean isAquece() {
```

```
        return aquece;
}public void setAquece(boolean aquece) {
        this.aquece = aquece;
}public boolean isBomba() {
        return bomba;
}public void setBomba(boolean bomba) {
        this.bomba = bomba;
}static PrintWriter saida;
private static BufferedReader entrada;
public boolean isConexao() {
        return conexao;
}public void setConexao(boolean conexao) {
        this.conexao = conexao;
}public int getTaxa() {
        return taxa;
}public void setTaxa(int taxa) {
        this.taxa = taxa;
}public SerialPort getPorta() {
        return porta;
}public void setPorta(SerialPort porta) {
        this.porta = porta;
}public static PrintWriter getSaida() {
        return saida;
}public static void setSaida(PrintWriter saida) {
```

```
        Arduino.saida = saida;
    }public String getEntrada() {
        String linha = "0,0";
        saida.print("#");
entrada=newBufferedReader(newInputStreamReader(porta.getInputStream()));
        try {
            if(entrada==null) {
                linha="0,0";
            }else {
                if(porta.isOpen()) {
                    linha = entrada.readLine();
                    saida.flush();
                }
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }return linha;
    }public void setEntrada(BufferedReader entrada) {
        this.entrada = entrada;
    }public static long getSerialversionuid() {
        return serialVersionUID;
    }public void controlaAquecimento(boolean estado){
        if(porta.isOpen())
```

```
        {
//Conforme a tabela, "a" representa liga e "b" desliga
        if(estado) {
                saida.print("b");
                aquece = true;
        }else {
                saida.print("a");
                aquece = false;
        }
                saida.flush();
        }else   {
                System.out.println("Ligue o Arduino!!!");
        }
}public void controleExaustores(int acao) {
        if(porta.isOpen())      {
                switch(acao) {
                case 1:
                        saida.print("d");
                        estadoEx1=true;
                        break;
                case 2:
                        saida.print("c");
                        estadoEx1=false;
                        break;
```

```
                case 3:
                        saida.print("f");
                        estadoEx2=true;
                        break;
                case 4:
                        saida.print("e");
                        estadoEx2=false;
                        break;
                }
                saida.flush();
        }else{
                System.out.println("Ligue o Arduino");
        }
}public void controlaBomba(boolean estado) {
        if(porta.isOpen() == true)
        {
                //Conforme a tabela, "g" representa liga e "h" desliga
                if(estado) {
                        saida.print("h");
                        bomba=true;
                }else {
                        saida.print("g");
                        bomba=false;
                }
```

```
                saida.flush();
            }else{
                System.out.println("Ligue o Arduino");
            }
        }public void conecta(String nomePorta){
            porta = SerialPort.getCommPort(nomePorta);
            porta.setBaudRate(taxa);
        if(porta.isOpen() == false){
            try {
            //Open the USB port and initialize the PrintWriter.
            porta.openPort();
            Thread.sleep(1000);
            saida = new PrintWriter(porta.getOutputStream());
            } catch(Exception c){}
                //Update the console and status.
                System.out.println("Sistema Conectado, pronto para execução :D");
                conexao=true;
                //lblStatus.setText("Status: Connected");
            }else {
                //If the port couldn't be opened print out to the console.
                System.out.println("Porta errada!!");
                conexao=false;
                //lblStatus.setText("Status: Opening USB failed");
}       }       }
```

ANEXO D – Controle do Arduino JAVA

```
//Classe de controle automatico da estufa
public class arduinoControle {
	private static boolean controle = false;
	private long cont = 0;
	private boolean estadoAquece = false;
	private boolean estadoEx1 = false;
	private boolean estadoEx2 = false;
	private boolean estadoBomba = false;
	private int setpointTemperatura = 25;
	private int tempoDesligado = 10;
	private int tempoLigado = 10;//em segundos
	public int getTempoDesligado() {
		return tempoDesligado;
	}
	public void setTempoDesligado(int tempoDesligado) {
		this.tempoDesligado = tempoDesligado;
	}
	public int getTempoLigado() {
		return tempoLigado;
	}
	public void setTempoLigado(int tempoLigado) {
		this.tempoLigado = tempoLigado;
	}
```

```
public int getSetpointTemperatura() {
	return setpointTemperatura;
}
public void setSetpointTemperatura(int setpointTemperatura) {
	this.setpointTemperatura = setpointTemperatura;
}
private int tempOpBomba = 10;//tempo em segundos
private int dados[] = {0,0};
public int getTempOpBomba() {
return tempOpBomba;
}
public void setTempOpBomba(int tempOpBomba) {
	this.tempOpBomba = tempOpBomba;
}
public long getCont() {
	return cont;
}
public void setCont(long cont) {
	this.cont = cont;
}
public boolean isEstadoAquece() {
	return estadoAquece;
}
public void setEstadoAquece(boolean estadoAquece) {
```

```
        this.estadoAquece = estadoAquece;
}
public boolean isEstadoEx1() {
        return estadoEx1;
}
public void setEstadoEx1(boolean estadoEx1) {
        this.estadoEx1 = estadoEx1;
}
public boolean isEstadoEx2() {
        return estadoEx2;
}
public void setEstadoEx2(boolean estadoEx2) {
        this.estadoEx2 = estadoEx2;
}
public boolean isEstadoBomba() {
        return estadoBomba;
}
public void setEstadoBomba(boolean estadoBomba) {
        this.estadoBomba = estadoBomba;
}
public arduinoControle() {
}
public int[] leitura(Arduino a) {
        String linha = a.getEntrada();
```

```
        String hum="",tem="";
        for(int i=0;i<linha.length();i++) {
                if(linha.charAt(i)==44) {
                        tem=linha.substring(0, i);
                        hum=linha.substring(i+1, linha.length());
                        i=linha.length();
                }
        }
        if(tem!=null&&hum!=null) {
                dados[0]=Integer.parseInt(tem);
                dados[1]=Integer.parseInt(hum);
        }
        return dados;
}
public void controla(Arduino a) {
        /*if(isControle()) {
                if(cont==2*tempOpBomba)
                {
                        cont=0;
                }
                if(cont<tempOpBomba) {
                        a.controlaAquecimento(true);
                        a.controleExaustores(1);
                        a.controleExaustores(3);
```

```
                a.controlaBomba(true);
                estadoEx1=true;
                estadoEx2=true;
                estadoAquece = true;
                estadoBomba=true;
        }
        else {
                a.controlaAquecimento(false);
                a.controleExaustores(2);
                a.controleExaustores(4);
                a.controlaBomba(false);
                estadoEx1=false;
                estadoEx2=false;
                estadoAquece = false;
                estadoBomba = false;
        }
        cont++;
}*/
if(isControle()) {
        int temp = dados[0];
        cont++;
        if(estadoBomba==false) {
                a.controlaBomba(false);
                if(cont==tempoDesligado) {
```

```
                estadoBomba=true;
                cont=0;
        }
}
if(estadoBomba) {
        a.controlaBomba(true);
        if(cont==tempoLigado) {
                estadoBomba=false;
        cont=0;
        }
}
if(temp<setpointTemperatura) {
        a.controlaAquecimento(true);
        a.controleExaustores(2);
        a.controleExaustores(4);
}
if(temp>setpointTemperatura) {
        a.controlaAquecimento(false);
        a.controleExaustores(1);
        a.controleExaustores(3);
}
estadoEx1 = a.isEstadoEx1();
estadoEx2 = a.isEstadoEx2();
estadoAquece = a.isAquece();
```

```
            }
        }
        public static boolean isControle() {
            return controle;
        }
        public void setControle(boolean controle) {
            this.controle = controle;
        }
        public void inicia() {
            controle = true;
        }
        public void para() {
            controle=false;
            cont=0;
        }
}
```

ANEXO E – Login JAVA

```
import java.awt.Color;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
public class login extends JFrame {
	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	private JTextField usuario;
	private static int nLogins = 6;
	private int idCerto = -1;
	private static String[] logins = new String[nLogins];
	private static String[] senhas = new String[nLogins];
	private static boolean[] perm = new boolean[nLogins];
```

```
private JPasswordField senha;
/**
 * Launch the application.
 */
public static void cadastraSenhas() {
        logins[0] = "luisGustavo";
        senhas[0] = "s1a1p0o6";
        perm[0] = true;
        logins[1] = "guilhermeBueno";
        senhas[1] = "1234";
        perm[1] = true;
        logins[2] = "arthurMiranda";
        senhas[2] = "1234";
        perm[2] = true;
        logins[3] = "joaoHenrique";
        senhas[3] = "1234";
        perm[3] = false;
        logins[4] = "matheus";
        senhas[4] = "1234";
        perm[4] = false;
        logins[5] = "luisFelipe";
        senhas[5] = "1234";
        perm[5] = false;
}
```

```
/**
 * Create the frame.
 */
public login() {
	cadastraSenhas();
	getContentPane().setBackground(new Color(153, 204, 102));
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setBounds(100, 100, 273, 119);
	FrmConfiguracoes();
	getContentPane().setLayout(null);
	this.setLocationRelativeTo(null);
	JButton entrar = new JButton("Entrar");
	entrar.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent arg0) {
			String s = senha.getText();
			String u = usuario.getText();
			for(int i=0;i<nLogins;i++) {
				if(s.equals(senhas[i])&&u.equals(logins[i])) {
					idCerto=i;
					i=nLogins;
				}
			}
			if(idCerto>-1) {
				tela1 frame = new tela1(perm[idCerto],logins[idCerto]);
```

```
                        frame.setVisible(true);
                        dispose();
                    }
                    else {
                        JOptionPane.showMessageDialog(null, "Usuário ou
Senha Incorreta Jovem!");
                    }
                }
            });
            getRootPane().setDefaultButton(entrar);
            entrar.setBounds(35, 77, 89, 23);
            getContentPane().add(entrar);
            JButton btnNewButton_1 = new JButton("Cancelar");
            btnNewButton_1.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    dispose();
                }
            });
            btnNewButton_1.setBounds(154, 77, 89, 23);
            getContentPane().add(btnNewButton_1);
            usuario = new JTextField();
            usuario.setBounds(105, 12, 129, 20);
            getContentPane().add(usuario);
            usuario.setColumns(10);
```

```
            JLabel lblNewLabel = new JLabel("Usu\u00E1rio");
            lblNewLabel.setBounds(39, 13, 46, 14);
            getContentPane().add(lblNewLabel);
            JLabel lblNewLabel_1 = new JLabel("Senha");
            lblNewLabel_1.setBounds(39, 50, 46, 14);
            getContentPane().add(lblNewLabel_1);
            senha = new JPasswordField();
            senha.setBounds(105, 48, 129, 20);
            getContentPane().add(senha);
        }
        public void FrmConfiguracoes() {
            Image iconeTitulo = null;
           try {
              iconeTitulo = ImageIO.read(getClass().getResource("icone.png"));
           } catch(IOException ex) {
           }
            setIconImage(iconeTitulo);
        }
}
```

ANEXOF – Tela 1 JAVA

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Image;
import java.io.IOException;
import java.util.Timer;
import java.util.TimerTask;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JProgressBar;
import javax.swing.border.EmptyBorder;
public class telaInicio extends JFrame {
	private JPanel contentPane;
	private int cont=0;
	int charging=0;
	int ts = 1;
	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
```

```
                public void run() {
                        try {
                                telaInicio frame = new telaInicio();
                                frame.setUndecorated(true);
                                frame.setVisible(true);
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
        }
}
/**
 * Create the frame.
 */
public telaInicio() {
        setForeground(Color.WHITE);
        setBackground(Color.WHITE);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        FrmConfiguracoes();
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
```

```
                this.getContentPane().setBackground(Color.WHITE);
                JLabel img = new JLabel("");
                img.setBackground(Color.WHITE);
                this.setLocationRelativeTo(null);
                img.setBounds(5, -30, 445, 300);
                ImageIcon i = new ImageIcon(this.getClass().getResource(
"imagemInicio2.png"));
                JLabel lblCarregando = new JLabel("Loading");
                lblCarregando.setBounds(21, 223, 116, 14);
                lblCarregando.repaint();
                contentPane.add(lblCarregando);
                img.setIcon(i);
                contentPane.add(img);
                JProgressBar progressBar = new JProgressBar();
                progressBar.setForeground(new Color(153, 204, 102));
                progressBar.setBounds(10, 236, 414, 14);
                progressBar.setMaximum(ts*100);
                contentPane.add(progressBar);
                final long tempoAuto = 100;//executa a cada 2.5 segundo;
                Timer timer = new Timer();
                TimerTask timerTask = new TimerTask() {
@Override
public void run() {
        cont++;
```

```
if(cont%5==0)
        charging++;
switch(charging) {
        case 1:
                lblCarregando.setText("Loading");
                break;
        case 2:
                lblCarregando.setText("Loading.");
                break;
        case 3:
                lblCarregando.setText("Loading..");
                break;
        case 4:
                lblCarregando.setText("Loading...");
                break;
}
if(charging==4) {
        charging=0;
}
progressBar.setValue(cont);
if(cont==ts*100) {
        login l = new login();
        l.setUndecorated(true);
                                l.setVisible(true);
```

```
                                        dispose();
        }
}
};
timer.scheduleAtFixedRate(timerTask, tempoAuto, tempoAuto);
        }
        public void FrmConfiguracoes() {
                Image iconeTitulo = null;
           try {
              iconeTitulo = ImageIO.read(getClass().getResource("icone.png"));
           } catch(IOException ex) {
           }
                setIconImage(iconeTitulo);
        }
}
```

ANEXO G – Tela Início JAVA

```
import java.awt.Color;
import java.awt.EventQueue;
import java.awt.Image;
import java.io.IOException;
import java.util.Timer;
import java.util.TimerTask;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JProgressBar;
import javax.swing.border.EmptyBorder;
public class telaInicio extends JFrame {
	private JPanel contentPane;
	private int cont=0;
	int charging=0;
	int ts = 1;
	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
```

```
                public void run() {
                        try {
                                telaInicio frame = new telaInicio();
                                frame.setUndecorated(true);
                                frame.setVisible(true);
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
        });
}
/**
 * Create the frame.
 */
public telaInicio() {
        setForeground(Color.WHITE);
        setBackground(Color.WHITE);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        FrmConfiguracoes();
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
```

```
                this.getContentPane().setBackground(Color.WHITE);
                JLabel img = new JLabel("");
                img.setBackground(Color.WHITE);
                this.setLocationRelativeTo(null);
                img.setBounds(5, -30, 445, 300);
                ImageIcon i = new ImageIcon(this.getClass().getResource(
"imagemInicio2.png"));
                JLabel lblCarregando = new JLabel("Loading");
                lblCarregando.setBounds(21, 223, 116, 14);
                lblCarregando.repaint();
                contentPane.add(lblCarregando);
                img.setIcon(i);
                contentPane.add(img);

                JProgressBar progressBar = new JProgressBar();
                progressBar.setForeground(new Color(153, 204, 102));
                progressBar.setBounds(10, 236, 414, 14);
                progressBar.setMaximum(ts*100);
                contentPane.add(progressBar);
                final long tempoAuto = 100;//executa a cada 2.5 segundo;
                Timer timer = new Timer();
                TimerTask timerTask = new TimerTask() {
@Override
public void run() {
```

```
cont++;
if(cont%5==0)
        charging++;
switch(charging) {
        case 1:
                lblCarregando.setText("Loading");
                break;
        case 2:
                lblCarregando.setText("Loading.");
                break;
        case 3:
                lblCarregando.setText("Loading..");
                break;
        case 4:
                lblCarregando.setText("Loading...");
                break;
}
if(charging==4) {
        charging=0;
}
progressBar.setValue(cont);
if(cont==ts*100) {
        login l = new login();
        l.setUndecorated(true);
```

```
                                        l.setVisible(true);
                                        dispose();
        }
}
};
timer.scheduleAtFixedRate(timerTask, tempoAuto, tempoAuto);
        }
        public void FrmConfiguracoes() {

            Image iconeTitulo = null;
          try {
             iconeTitulo = ImageIO.read(getClass().getResource("icone.png"));
          } catch(IOException ex) {
          }
            setIconImage(iconeTitulo);
        }
}
```