DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Automação de máquinas para beneficiamento de frutos

Relatório submetido à Universidade Federal de Santa Catarina como requisito para aprovação na disciplina

DAS 5511: Projeto de Fim de Curso

Filipe Odozynski da Silva

Florianópolis, agosto de 2016

Automação de máquinas para beneficiamento de frutos

Filipe Odozynski da Silva

Esta monografia foi julgada no contexto da disciplina
DAS5511: Projeto de Fim de Curso
e aprovada na sua forma final pelo
Curso de Engenharia de Controle e Automação

Prof. *Leandro Buss Becker*

_____
Assinatura do Orientador

Banca Examinadora:

Jackson Antonio Caregnato
Orientador na Empresa

Prof. Leandro Buss Becker
Orientador no Curso

Prof. Daniel Costa Ramos
Avaliador

Antonio Adalberto Duarte Junior
Rodrigo Holz Schuler
Debatedores

AGRADECIMENTOS

Primeiramente a Deus, por ter me dado força para superar todas as dificuldades.

Aos meus pais, Joel e Nana, que tem me apoiado ao longo dessa caminhada.

Aos meus orientadores o Professor Leandro Buss Becker, que me orientou com dedicação não medido esforços e criticas bucando tirar o meu melhor, e o Jackson Caregnato, que me mostrou como a MF funciona e me ajudou a entender melhor os processos de classificação.

Ao sr. Ailton Bitencourt e a sua família, que foram os responsáveis pela oportunidade de trabalhar neste projeto dentro da MF máquinas em Fraiburgo.

A Agrícola Catarinense e ao seu gerente de produção o técnico agrícola Celso Kafer Marinês.

A Viviana e ao meu irmão Gabriel, que juntamente me ajudaram a formatar esse trabalho.

Aos técnicos da MF, em especial ao Pedro Moraes, Renan Basquerote e ao Thiago Lima que me ajudaram não apenas no projeto, mas com conversas animadas e um ótimo ambiente de trabalho.

E finalmente a minha amada Juliana que sempre me motivou e teve paciência comigo.

SILVA, Filipe Odozynski da; BECKER, Leandro Buss. **Automação de máquinas para beneficiamento de frutos**. Florianópolis, 2016. (83 f.) Projeto de Fim de Curso (Graduação em Engenharia de Controle de Automação) – Universidade Federal de Santa Catarina, Florianópolis, 2016.

RESUMO

A MF máquinas LTDA contempla o amplo mercado da agricultura envolvendo automação/fabricação de maquinário para beneficiamento de frutos e legumes. Após a MF passar por um processo de reformação, fez-se imprescindível a implementação de melhorias ao maquinário para classificação de frutos e legumes: os classificadores. Urge, então, a motivação ao desenvolvimento de um sistema próprio visando a redução de custo e que atenda o escopo da empresa. Têm-se como objetivos: 1) Construção de *software* com comunicação serial para o gerenciamento de classificadores. Busca-se um programa de fácil configuração e operação provendo funcionalidades como configurar os calibres, peso mínimo e máximo, além de retornar informações ao operador (velocidade de operação da máquina e peso total classificado). 2) Criação de *firmware* para controle dos atuadores da máquina, encarregando-o pelo elo final do processo. Portanto, foi elaborado um software que está em fase de validação e um firmware que cumpre as solicitações do cliente.

Palavras-chave: Automação. Engenharia de Software. Classificação de frutos.

ABSTRACT

MF máquinas LTDA includes the agriculture's broad market involving automation/manufacturing for machinery to classification process improvement of fruits and vegetables. After going through a reforming process, it became essential to implement upgrades in classification of fruits and vegetables machinery: the classifiers. Thereafter, starts motivation for the development of a delivery system in order to reduce cost, according to the company's scope. The aim of the project is: 1) To build a software with serial communication for managing classifiers, seeking an easy setup and an operation program providing features like set gauges, minimum and maximum weight and the return information to the operator (machine operating speed and total weight rated). 2) Firmware creation for controlling the machine actuators, linking this with the entire process. Therefore, it has elaborated software that is under validation step and a firmware that fulfills customer requests.

Keywords: Automation. Software Engineer. Fruit Classification.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

APIs	Application Programming interfaces
CPU	Central Processor Unit
DTE	Digital Transducer Electronics
I/O	Input/Output
IDE	Integrated Development Environment
JMV	Java Machine Virtual
OpenUP	Open Unified Process
PC	Personal Compilter
UML	Unified Modeling Language ou Linguagem de Modelagem Unificada
UP	Unified Process

SUMÁRIO

1 INTRODUÇÃO

O presente trabalho está vinculado à disciplina “Projeto de Final de Curso”, tendo como objetivo, adaptar um sistema de automação para um classificador de frutos e legumes na empresa MF Máquinas Ltda., situada em Fraiburgo - Santa Catarina. Tal empresa, fundada em 1996, busca em seus valores suprir a necessidade do mercado em relação a empresas especializadas no ramo de fabricação e manutenção em máquinas para beneficiamento de frutas e legumes, sendo o seu foco a melhoria dos processos de limpeza, classificação e embalagem de frutos e legumes, através de sistemas automatizados.

Segundo Pinto (2015), a automação proporciona uma série de vantagens, as quais proporcionam padronização, maior eficiência e menor custo. Pinheiro (2004), ainda afirma que sendo a automação a capacidade de se executar comandos, obter medidas, regular parâmetros e controlar funções automaticamente, sem a intervenção humana, esta torna-se também sinônimo de assimilação, ou seja, transcende da função mais simples a mais complexa, existindo assim, um ou mais sistemas que permitam que um dispositivo seja controlado de modo inteligente, tanto individualmente quanto em conjunto. De acordo com Silveira (2003), com a automatização substituímos os gastos de energia humana, como o esforço muscular e mental, por elementos eletromecânicos computáveis. Esse conceito de automação é visto em diversos cenários, inclusive cotidianos, como, por exemplo, a máquina de lavar roupa para a lavadeira, a fotocopiadora para o escrivão, ou o robô para o operário industrial. Visto isso, pode-se aplicar tais vantagens e princípios à classificação de frutos e legumes e torná-la mais lucrativa aos produtores.

Dentre as maquinas produzidas dentro da MF, a que tem relevância para esse trabalho é o classificador eletrônico. Este é responsável por pesar individualmente cada fruto e classifica-lo de acordo com o seu peso.

A MF produz a parte física dos classificadores, porém o sistema lógico, composto por um *software* responsável pelo gerenciamento do classificador e um controlador de saídas responsável pela atuação são licenciados por outra empresa, dificultando sua manutenção e personalização, uma vez que o sistema já se encontra instalado em mais de 50 máquinas pelo Brasil.

Diante disso, teve-se como objetivos a elaboração de um *software* próprio da empresa que exerça as funções de gerenciamento da classificação, seja de fácil operação e configuração, gere arquivos e interaja com os módulos já existentes; bem como o desenvolvimento de um *firmware* para o controlador de saídas, já que o atual é terceirizado.

2 MF MÁQUINAS LTDA E SEUS EQUIPAMENTOS DE CLASSIFICAÇÃO

A MF Máquina Ltda. (MF MÁQUINAS, 2016) foi fundada em 1996 por um grupo de mecânicos industriais que possuíam uma vasta experiência em maquinários para classificação de maçãs em Fraiburgo, SC. Nesse contexto, a MF percebe um nicho na prestação de serviços as empresas da região, já que estas detinham maquinário importado com alto custo de manutenção sem representantes a nível nacional para classificadores. O crescimento da MF e o aprimoramento técnico fizeram com que a empresa se voltasse a assistência e atendimentos aos pequenos e médios produtores de frutos com o comércio de classificadores menores, estes inicialmente mecânicos e posteriormente eletrônicos.

2.1 TIPOS DE CLASSIFICADORES

A MF divide os seus classificadores em dois tipos:

- Mecânico;
- Eletrônico.

2.1.1 Classificador Mecânico

Os classificadores mecânicos foram os primeiros produzidos pela empresa. Seu sistema de controle e pesagem é feito por meio de um mecanismo associado à queda do fruto atrelado a um sistema de molas, onde as molas ficam nas saídas e através da sua constante elástica deixa um fruto cair ou não. As principais limitações nesta categoria são:

- Baixa versatilidade: o conjunto de molas é regulado para apenas um determinado peso e uma saída específica. Logo, um aumento na quantidade de frutos com tamanhos homogêneos irá sobrecarregar a saída, consequentemente congestionando a operação da máquina.
- O desgaste das molas: por ser um processo mecânico, naturalmente a mola sofre uma degradação e perde precisão ao classificar o fruto.
- Operação: A regulação das molas não é trivial, portanto, necessita-se de profissional devidamente capacitado para realizar a calibragem do maquinário. No

estado de São Paulo – por exemplo – a variedade de frutos selecionados torna inexequível o uso de um classificador mecânico, já que o gasto com calibragem inviabiliza o seguimento. Diferentemente, em Fraiburgo/SC, a monocultura da maçã permite o emprego e serventia deste método.

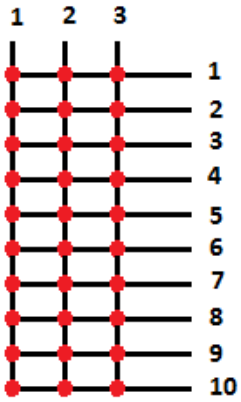
Todas as limitações citadas confinaram e restringiram o mercado e, por conseguinte, difundiu-se o objetivo de projetar e implementar um novo modelo de classificador para solucionar as exigências.

2.1.2 Classificador Eletrônico

Os classificadores eletrônicos emergem como alternativa as defasagens dos mecânicos. Esses possuem uma série de vantagens: rápida configuração e distribuição de frutos, desgaste mecânico menor, calibração simples e fácil operação, a qual dispensa longos períodos de capacitação ou experiência prévia.

Os classificadores fundamentam-se em relação ao número de linhas e o número de saídas. Deste modo, pensa-se num classificador como uma matriz, onde as colunas representam as linhas e as linhas representam as saídas. Por exemplo, quando falamos que um classificador é de três linhas e de dez saídas, sabemos que ele possui três células de carga (cada linha possui uma unidade de pesagem) e trinta solenoides de acionamento (pois cada saída tem três solenoides, já que o exemplo prevê três linhas). A figura 1 demonstra a ideia, onde cada ponto vermelho é um solenoide.

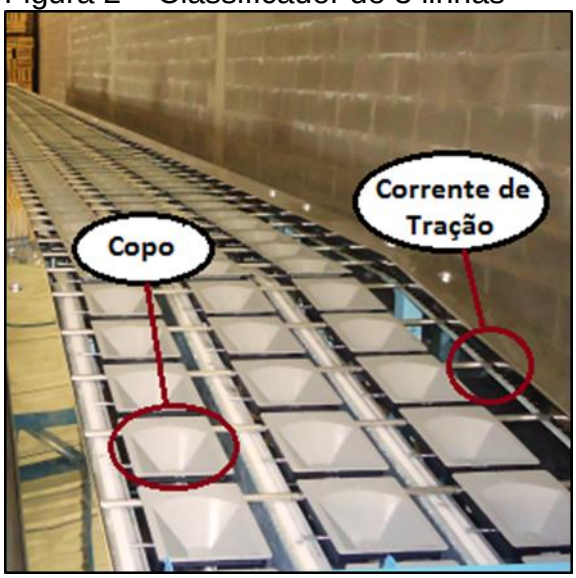
Figura 1 – Classificador de 3 linhas



Fonte: Do Autor.

O classificador eletrônico é uma máquina, a qual manuseia um conjunto de bandejas individuais, também conhecidas como copos. Cada copo transporta um fruto, o qual é tracionado por uma corrente, conduzindo os copos da pesagem até a queda adequada. A figura 2 evidencia um copo e seu transporte sobre a corrente de tração.

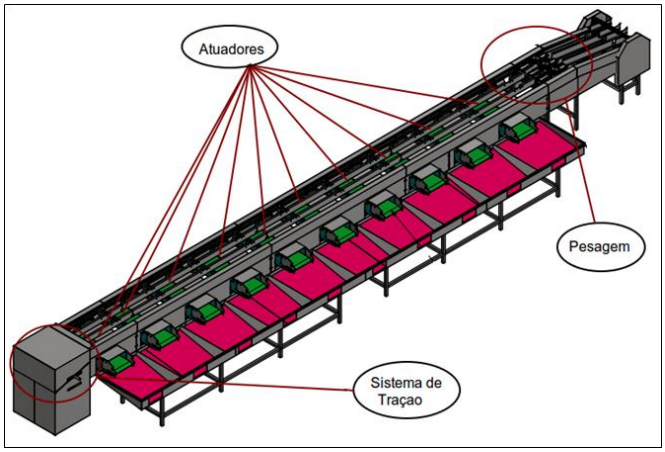
Figura 2 – Classificador de 3 linhas



Fonte: Do Autor.

Pode-se dividir o classificador em três grandes setores: o de medição, composto pelas células de carga; o de atuação, representado pelos solenoides e o de processamento onde se localiza o computador. A Figura 3 propõe uma visão geral do classificador e seus elementos.

Figura 3 – Visão geral de um classificador de 3 linhas e 10 saídas

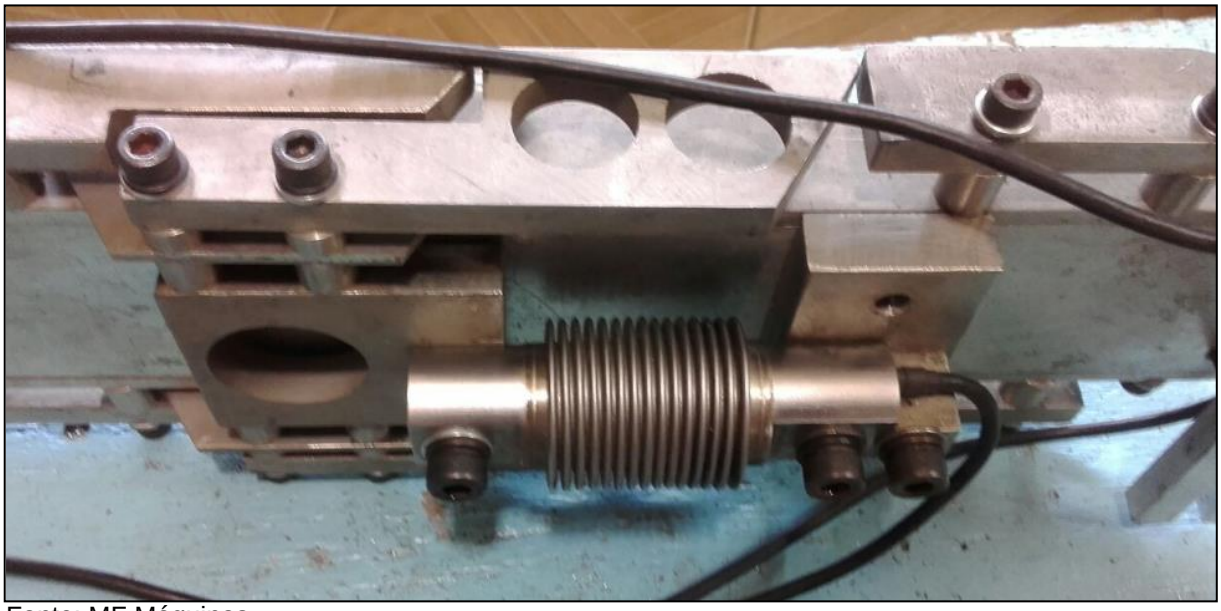


Fonte: Do Autor.

2.1.2.1 Pesagem

A pesagem é feita no início do processo, cada linha possui uma base com uma célula de carga do tipo z6c3/10 kg, fabricada pela HBM, sendo pesado individualmente cada copo.

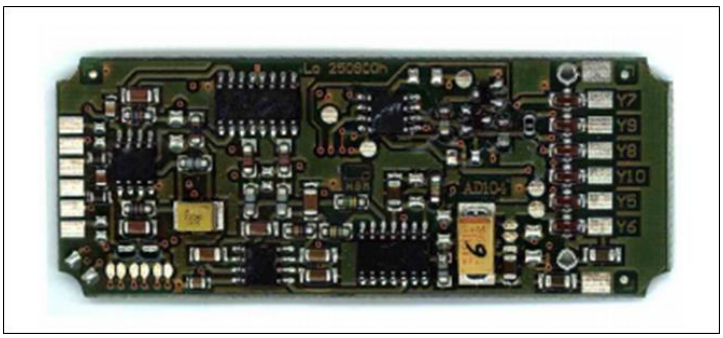
Figura 4 – Base de célula de carga com célula de carga z6c3/10 kg



Fonte: MF Máquinas.

O sinal dessa célula é tratado por um *DTE* (*Digital Transducer Electronics*) modelo AD104(HBM®). O AD104 possui um conjunto de filtros digitais, opção de disparo de pesagem externa (*External trigger*) e uma interface serial do tipo RS-232 que permite receber e enviar dados já no formato ASCII.

Figura 5 – AD 104



Fonte: HBM Measurement With Confidence

Ressalta-se a importância da opção de disparo de pesagem externa, uma vez que esta é responsável por indicar o início de uma nova pesagem por meio de um sensor fotoelétrico difuso da fabricante Omron; modelo e3fa-dp22.

Figura 6 – Sensor fotoelétrico

Fonte: OMRON.

2.1.2.2 Atuação

A atuação é feita por solenoides acionados individualmente em um circuito de potência. Esse circuito recebe o sinal de um microcontrolador do tipo *ATmega2560* que, por sua vez, está conectado ao computador, o qual se interliga pela comunicação serial (RS-232).

A figura 7 apresenta um suporte com um solenoide no guia da máquina.

Figura 7 – Solenoide fixada no guia

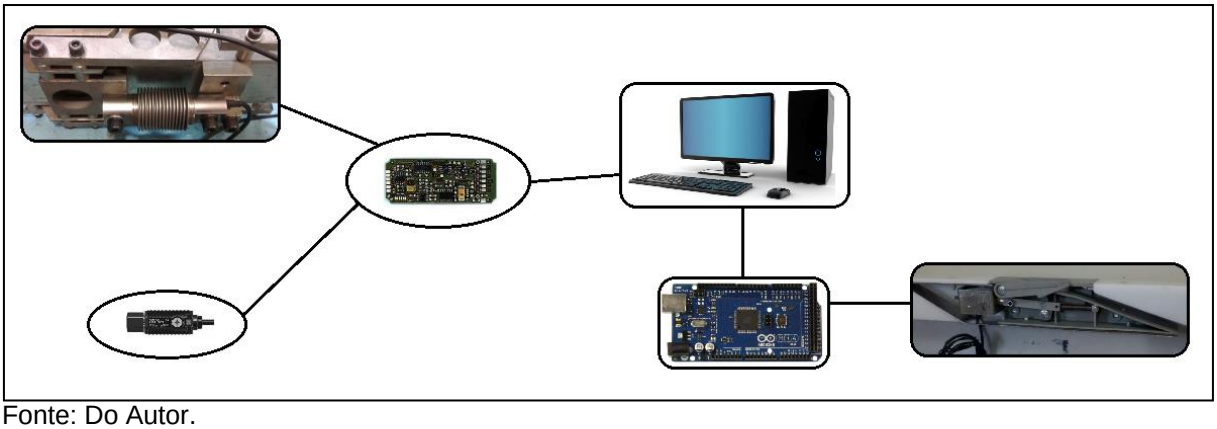
Fonte: MF máquinas.

2.1.2.3 Gerenciamento

O sensor fotoelétrico detecta a passagem de um copo e dispara a pesagem. A célula de carga capta essa variação de peso e o DTE (AD_104) conduz e processa a informação para um PC (*Personal Compilter*). No computador está a parte de gerenciamento responsável por receber o dado provindo do AD_104, verifica-se em sua base de dados qual é a ação adequada e avalia-se quando e qual solenoide deve ser ativado. Além disso, o gerenciamento é responsável pela interação homem- máquina, administração de arquivos e apresentação dos dados referentes à produção.

A figura 8 demonstra como todos os componentes estão ligados.

Figura 8 – Ligação dos componentes de automação



Fonte: Do Autor.

3 CLASSIFICADORES ELETRÔNICOS E SEUS DESAFIOS

Os classificadores eletrônicos produzidos pela MF possuem um sistema de gerenciamento com as seguintes funções:

- Abrir/Fechar comunicação com o módulo AD_104: como discutido na seção 2.1.2.1 o módulo (AD_104) possui um conjunto de funções, como por exemplo, a função de transmissão contínua de pesagem ou de zeragem, cabendo ao sistema gerenciar o envio e recebimento de mensagens;
- Trabalho dos dados recebidos: o módulo AD_104 envia dados de pesagem ao sistema, que após avaliar esse peso, cruzando com os valores de sua base de dados, decidindo assim onde determinado fruto deve ir, ou qual função executar devido a outro tipo de mensagem recebida;
- Atuação frente aos dados: após receber e tratar a mensagem, o sistema deve atuar. Podendo desde imprimir uma mensagem de erro ou ainda mandar uma mensagem ao controlador de saídas, informando qual a saída indicada ou mudando o tempo de atuação em uma determinada saída;
- Abrir/Fechar comunicação com o Controlador de saídas: o sistema também é responsável por configurar o controlador de saída e enviar mensagens sobre qual(is) saídas deverão ser acionadas;

Pelo fato do software e do controlador de saídas serem terceirizados, há um alto custo na manutenção das licenças. A defasagem de atualizações não comporta as soluções previstas pela empresa, gerando obstáculos na confecção de um produto personalizado em relação a alterações ou premência de novas funcionalidades. Até porque, inicialmente o sistema é resultado de uma adaptação que – invariavelmente – contém inúmeras funções desnecessárias. Diante dessas circunstâncias, urgem-se o desenvolvimento de um *software* para gerenciar o classificador e um *firmware* para o controlador de saídas, ambos idealizados a atender as exigências e requisitos prévios, além de primarem por manutenção facilitada e autônoma.

3.1 GERENCIADOR

Para a parte de gerenciamento, o *software* terá que atender uma série de necessidades tanto de processamento, quanto de *hardware*. A lista a seguir lista as principais necessidades:

- Comunicação serial: uma vez que este é o padrão adotado no módulo AD_104;
- Tempo de execução: delay entre o recebimento do peso do fruto e a ação disparada.
- Interface simples e objetiva: o classificador eletrônico deve ser de fácil operação, não sendo fundamental experiência prévia ou curso de capacitação para manejo.
- Sistema de configuração objetivo: aumentar a usabilidade e autonomia das configurações.

3.2 CONTROLE DE SAÍDA

Um firmware será projetado para administrar o controlador de saída já existente, de acordo com as seguintes exigências:

- Comunicação serial: devido ao protocolo de comunicação disponível no *hardware* do controlador;
- Tempo de execução: ser capaz de receber o comando vindo do PC e atender esse comando;
- Configurável: atender a número de saídas e linhas variáveis, o qual depende das exigências do cliente;

3.3 OBJETIVOS

Os dois grandes objetivos são o desenvolvimento de um *software* e de um *firmware*, para a substituição dos já existentes.

3.3.1 Software

As metas propostas para o desenvolvimento do *software* foram:

- Escolha de uma metodologia de desenvolvimento;
- Elaboração de documentos: requisitos, caso de uso, diagrama de classes e protótipos de tela;
- Codificação;
- Validação através de testes;
- Entrega do *software*;
- Treinamento do corpo técnico da empresa.

3.3.2 Firmware

Para o *firmware* as metas foram:

- Integração com o novo *software*: relacionado a criação de um conjunto de códigos, sendo que esse deve ser documentado;
- Versátil as diferentes configurações de máquinas existentes: o número de linhas e de saídas é variável, deve existir apenas um *firmware* que consiga ser implantando em todas as máquinas.
- Compatível com o controlador de saídas já existente: usar os componentes do controlador de saída visando a compatibilidade com o controlador de saída atual.

4 GERENCIADOR MF

Objetivando o desenvolvimento de um *software* que gerencie a máquina e que se comunique com outros sistemas, percebe-se um conteúdo referente a engenharia de *software*, esta formalmente define-se como “a aplicação de abordagens sistemáticas, disciplinadas e quantificáveis ao desenvolvimento, operação e manutenção de *software*, além do estudo dessas abordagens”. (IEEE COMPUTER SOCIETY, 2004).

Há diversos métodos propostos ao desenvolvimento de um software, no entanto Tosing (2008) destaca que “qualquer que seja o método que vier a ser escolhido para o desenvolvimento de um *software*, estará espelhado em um ciclo de vida de desenvolvimento. Este ciclo de vida pode ser entendido como um roteiro de trabalho, constituído em geral de macro etapas com objetivos funcionais na construção de um *software*, onde também é possível visualizar-se a interdependência existente entre essas macro etapas”. Dentre esses métodos, o selecionado para o desenvolvimento do software é uma variação do Processo Unificado (UP), o OpenUP (Open Unified Process).

O UP foi criado por três importantes pioneiros da orientação a objetos nos anos 1990 (JACOBSON; BOOCH; RUMBAUGH, 1999 *apud* WAZLAWICK, 2013), tendo como pilares:

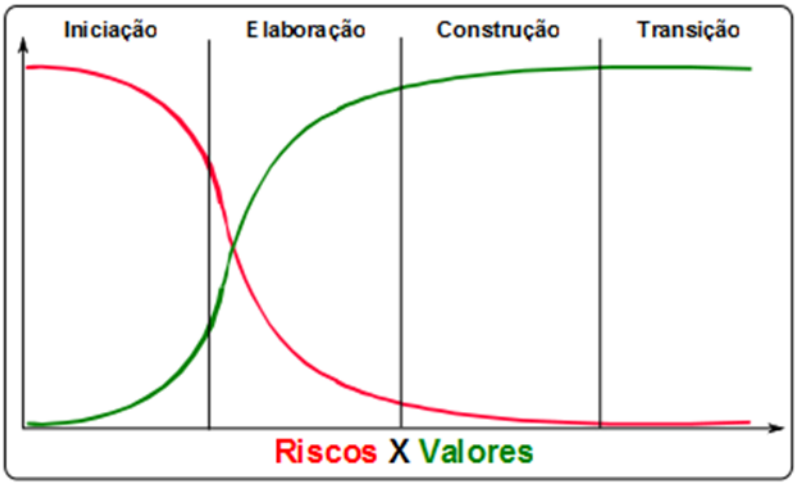
- Ser dirigido por casos de uso;
- Ser centrado na arquitetura, esta entendida como um modelo que define a estrutura de informação, suas possíveis operações e sua organização em componentes;
- Ser iterativo e incremental;
- Ser focado em riscos;

Compilado – em analogia a versão UP - e mantido pela Fundação Eclipse, o OpenUp/Basic é um método independente de ferramenta e de baixa cerimônia, ou seja, não é exigido grande precisão e detalhamento nos documentos, é um processo voltado para pequenas equipes, co-localizadas; flexível, o qual permite ser modificado ou estendido para perfazer e suprir as condições do projeto. Útil para equipes com pouca experiência, em contato direto com o cliente e que não possuem papéis tão rigorosos, não representando responsabilidades individuais sobre as

tarefas. Um papel não se limita a descrever o principal executor de uma tarefa, ao invés disso os papéis incluem uma perspectiva sobre colaboração fornecendo executores adicionais para cada tarefa. Esta perspectiva sobre os papéis fortalece a nova geração de processos de desenvolvimento de *software*, mais focada na interação das pessoas. (WAZLAWICK, 2013).

O ciclo de vida do OpenUP pode ser visto na figura 9, deixando claro suas fases e mostrando a relação de riscos por valores.

Figura 9 – Ciclo de vida do modelo OpenUP



Fonte: Medium (2014).

Ressalta-se que o projeto não está concluído, faltando a fase de transição que será futuramente implementada.

4.1 INICIAÇÃO

Na iniciação os objetivos foram:

- Levantamento de requisitos;
- Criação dos diagramas de caso de uso;
- Protótipos das primeiras telas.

4.1.1 Requisitos

Os requisitos são divididos em dois tipos: funcionais e não funcionais. Sommerville (2011) define os requisitos funcionais como declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações específicas, ao passo que os requisitos não funcionais são restrições aos serviços ou funções oferecidas pelo sistema. Ressalta-se que, na prática, tal tarefa por muitas vezes é árdua por ser passível de interpretações. Logo, deixaram-se os requisitos o mais detalhado possível, muitas vezes dando a impressão de serem funcionalidades do sistema.

Iniciou-se com um conjunto de três entrevistas, informais, onde se tentou chegar a um consenso sobre os requisitos. Essas entrevistas foram realizadas com o proprietário da empresa Sr. Ailton Bitencourt, com o responsável técnico da parte elétrica/eletrônica, Pedro Moraes que possui contato direto com as máquinas e experiência no mercado, com o gerente de produção da empresa Agrícola Catarinense, Sr. Celso Kafer Marinês, que tem sua produção localizada próxima a sede da MF e com o responsável pela área de projetos Jackson Caregnato.

Esses requisitos foram refinados buscando a clareza para ambos os lados, os requisitos não funcionais podem ser vistos no quadro 1 do apêndice A e os requisitos funcionais no quadro 2 do apêndice A.

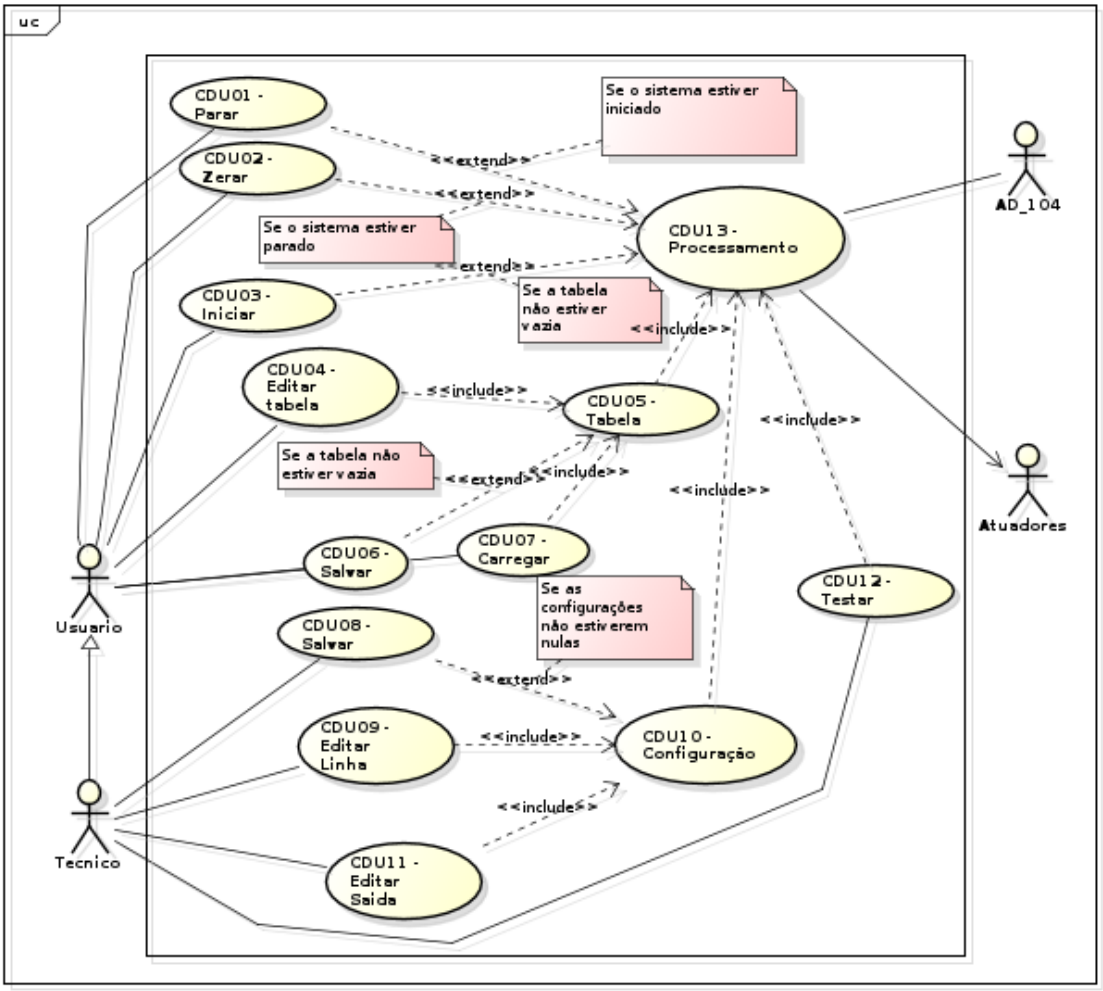
4.1.2 Casos de Uso

Em paralelo aos levantamentos de requisitos começou-se a esboçar o diagrama de caso de uso que, segundo Guedes (2011), tem como objetivo apresentar uma visão externa geral das funcionalidades que o sistema deverá oferecer aos usuários, sem se preocupar com a questão de como tais funcionalidades serão implementadas.

O primeiro passo foi a definição dos atores, na qual um ator é um usuário do sistema, este também pode representar algum *hardware* especial ou mesmo outro *software* que interaja com o sistema, portanto, qualquer elemento que faça fronteira com o sistema. (GUEDES,2011)

O diagrama de caso de uso pode ser visto na figura 9 a seguir:

Figura 10 – Diagrama de caso de uso



Fonte: Do Autor.

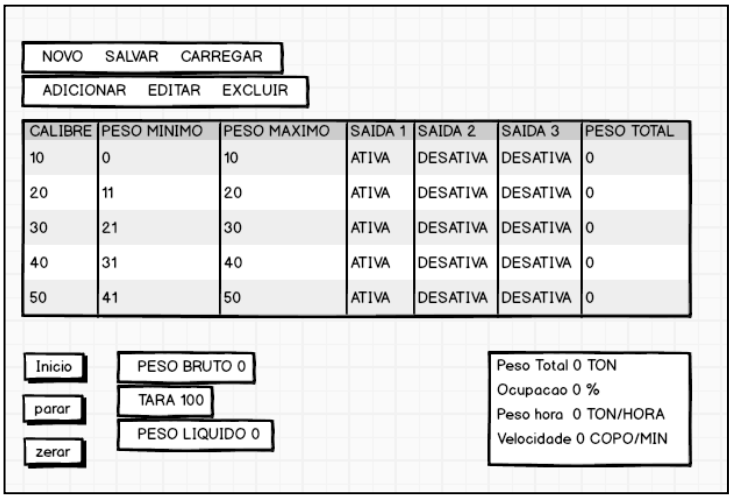
Os quadros com a discrição detalhada de cada caso de uso estão no apêndice B.

4.1.3 Protótipos das Primeiras Telas

Paralelamente a construção dos diagramas de caso de uso, iniciou-se a confecção das telas de acordo com as funcionalidades previstas, atendendo as solicitações do cliente quanto às ações da interface projetada.

Serão apresentados os esboços de cada tela e uma breve explicação das funcionalidades.

Figura 11 – Tela Principal

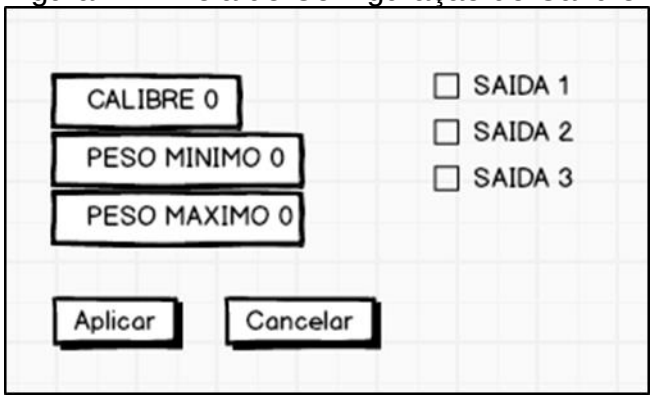


Fonte: Do Autor.

Esta é a tela principal que o Ator Usuário tem acesso. A partir dela é possível controlar toda a classificação de fruto. É a tela pela qual se “Inicia”, “Parar” ou “Zerar” a pesagem, sendo que para se iniciar, o processo tem que estar parado e vice versa. Para zerar as células é necessário que a maquina esteja parada. Nela se adiciona e/ou edita o calibre. Também é possível salvar essa tabela de calibres ou ainda abrir uma já existente nos arquivos.

Nela estão as saídas que interessam ao “Usuário”: “Peso total” (peso de total de frutos que passou pela maquina), a “Ocupação da máquina”, “Peso hora” (estimativa de peso por hora calculado pela ocupação e pela velocidade) e velocidade (copos/minuto).

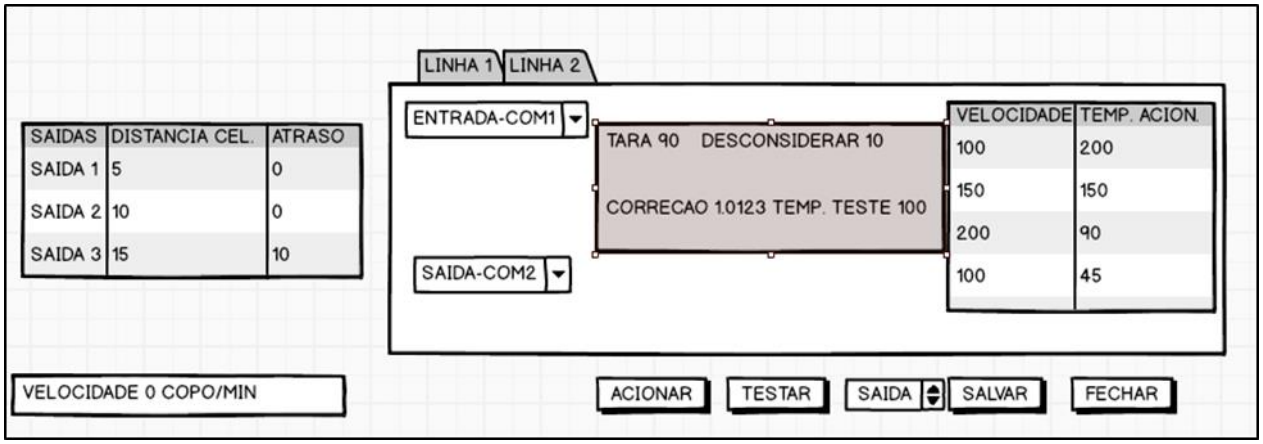
Figura 12 – Tela de Configuração do Calibre



Fonte: Do Autor.

Na figura 12 é apresentada a tela que configura um calibre, ao clicar no botão da tela principal Editar ou Adicionar, ela é aberta, nela podemos definir o peso máximo, o calibre, e qual (is) saída (s) aquele calibre deve ir.

Figura 13 - Tela de configuração

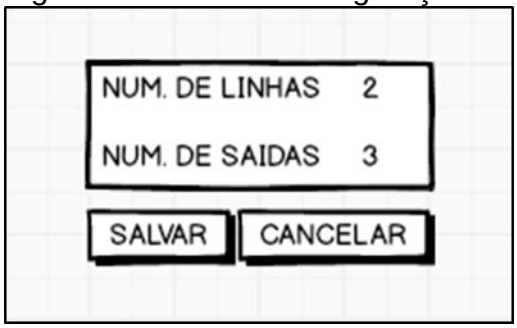


Fonte: Do Autor.

A figura 13 representa a tela de configuração, aonde o Ator Técnico tem acesso para configurar o sistema, definindo a distância dos copos a partir da base de pesagem, o tempo de atraso especifico para cada saída, caso seja necessário. Ainda define para cada linha a porta de entrada (AD_104) e saída (Controlador de Saída) de comunicação, a tara (peso a ser desconsiderado por ser o peso do copo), o valor a ser desconsiderado (caso o peso seja igual ou inferior) o fator de correção, o tempo de teste e a tabela de velocidade por tempo de acionamento.

Há opções de “Acionar” uma saída especificada ou a de “Testar” periodicamente uma saída selecionada. Além disso, possui um indicador de velocidade utilizado no auxílio do acerto dos tempos de quedas (tabela de velocidade por tempo de acionamento).

Figura 14 - Tela de configuração de linhas e saídas



Fonte: Do Autor.

Esta tela é responsável por configurar o número de linhas e o número de saídas, buscando atender as necessidades dos clientes.

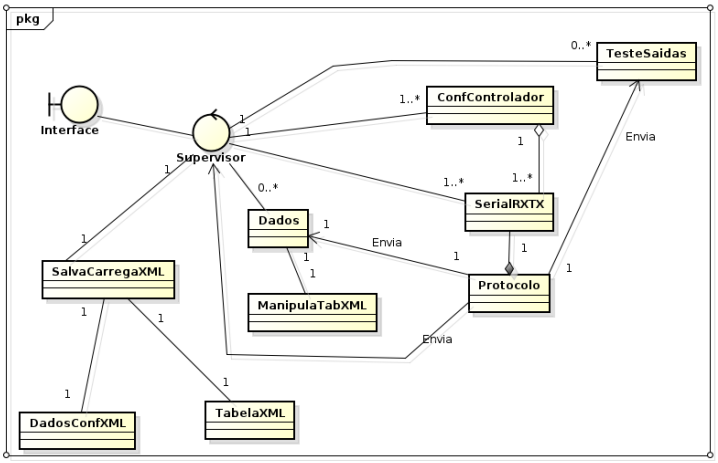
4.2 ELABORAÇÃO

Com os requisitos e os diagramas prontos começou-se a trabalhar na arquitetura do software. Como já é sabido o OpenUP é focado nos riscos – principalmente - fase de concepção, uma vez que se foca mais nos casos de uso de maior risco. Wazlawick (2013) define que o risco é maior quanto mais complexo, difíceis de trabalhar e imprevisíveis forem os casos de uso, contudo é na fase de Elaboração que o risco cai consideravelmente, basta observar a figura 8 (figura do ciclo de vida), posto que deve-se ao fato da arquitetura ter maior cuidado com os casos de uso de maior risco já nas primeiras iterações.

A arquitetura é focada na estrutura da informação. Nesta fase é criado o diagrama de classes no qual uma classe é como um gabarito que dará “estrutura” a um objeto. Pela definição de uma classe, são descritos os atributos que o objeto terá e como as funcionalidades poderão ser aplicadas ao objeto. Tais funcionalidades são descritas por métodos. (DCA, 2000).

O diagrama final de classes pode ser visto na figura 15, para facilitar a compreensão não foi adotado o padrão que a UML define, portanto apenas demostramos as classes e suas relações.

Figura 15 - Diagrama de Classes



Fonte: Do Autor.

Os quadros que definem os atributos e os métodos, além de apresentar um breve resumo de cada classe estão no Apêndice C.

Cabe a parte de elaboração a escolha da tecnologia de codificação. Esta deve ser orientada a objetos, devido a metodologia de projeto escolhida, uma vez que a OpenUP é centrada no UML. Somado a isso e em relação aos requisitos não funcionais, o software pode rotear em diferentes plataformas e participar do gerenciamento de arquivos.

Dentre as opções o Java mostrou-se uma boa escolha por possuir comunidade com mais de nove milhões de programadores em todo o mundo que, de forma eficiente, é testado, refinado, estendido e comprovado por desenvolvedores, arquitetos e entusiastas. (JAVA, 2016). Sendo uma linguagem de programação orientada a objetos, possui inúmeras vantagens como a de multiplataformas tendo uma máquina virtual a JMV (Java Machine Virtual), programação paralela (*multithreading*) e ricas coleções de classes existentes nas bibliotecas de classe, que também são conhecidas como APIs do Java ou Java APIs (Application programming interfaces). (DEITEL, 2009).

4.3 CONSTRUÇÃO

Com a arquitetura do sistema pronta, inicia-se a codificação, os testes, e consequentemente os retrabalhos.

4.3.1 Codificação

Como decidido na fase de Elaboração, a linguagem de desenvolvimento é o Java por ter se mostrado uma excelente ferramenta ao possuir a vantagem de, pelas suas APIs, agilizar os processos de desenvolvimento.

Trata-se da codificação em três subseções por ordem decrescente de risco.

4.3.1.1 Comunicação Serial

A API RXTXcomm – selecionada para o desenvolvimento - é baseada na API Javacomm distribuída pela própria Sun, com a vantagem de ser portável para Linux, Windows e Mac, enquanto que a Javacomm em sua atual versão só é portável para Linux. (DEVMEDIA, 2007).

Com a RXTXcomm são criados objetos responsáveis por:

- Visualização das portas seriais disponíveis;
- Abertura e fechamento da comunicação serial;
- Configuração das opções de comunicação (BaundRate, pariedade, endereço da porta serial);
- Leitura/Escrita de dados na porta serial;

A RXTXcomm aplica o conceito de interface Listener. Tal interface define um objeto com vários dependentes anexados a ele. A partir do momento que este objeto muda de estado todos os seus dependentes são avisados, deste modo, tem-se por função “ouvir” alterações no objeto monitorado. (PRECISO ESTUDAR SEMPRE, 2015). Logo, todo evento na porta serial é tratado pelo objeto criado pela classe que implementa a API RXTXcomm.

O objeto SerialRXTX, responsável pela comunicação serial, implementa a API RXTXcomm. Ao chegar um dado à porta serial, o objeto SerialRXTX o envia para o objeto Protocolo. Neste há um método objetivado a tratar esse dado com intuito de ser transformado e usado pelo resto do software.

O objeto Protocolo herda da classe *Observable* a característica para ser observável por outros. Por tanto, é responsável por enviar o dado já transformado em “peso atual” aos demais. O objeto Dados, por exemplo, é o responsável por cruzar o “peso atual” com a tabela de calibres e assim decidir quando e onde aquele fruto deve ir. Outro exemplo é o objeto Supervisor, este sendo o que irá retornar ao objeto Interface o “peso atual” para imprimir na tela o peso naquele exato momento.

4.3.1.2 Entrada e Saída de Arquivos

A ideia da empresa é de possivelmente vender outros tipos de soluções mais voltadas para o gerenciamento, solicitou-se um tipo de arquivo que poderia ser

usado para exportar planilhas, ou ainda que pudesse ser de fácil manuseio, por isso usa-se o XML, eXtensible Markup Language, linguagem de marcação recomendada pela W3C na criação de documentos com dados organizados hierarquicamente. (W3SCHOOLS.COM, 2016).

A API Java Architecture for XML Binding (JAXB) facilitou a estruturação e a função salvar/carregar dos arquivos.

O princípio é de que haja um objeto que armazene as informações, no caso os objetos DadosConfXML e TabelaXML. Esses trabalham com o objeto SalvaCarregaXML por ser o elo de ligação do software com os arquivos do PC, ou seja, o SalvaCarregaXML irá receber o objeto DadosConfXML ou o TabelaXML e irá salvar em um arquivo devidamente nomeado no formato desejado. Alternativamente, este irá pegar um arquivo com a extensão suportada pelo sistema e transformará as informações neles contidas em um objeto DadosConfXML ou o TabelaXML, para que o software possa trabalhar.

4.3.1.3 Interface

Por fim foi tratada a parte de interface com o usuário, utilizou-se o API Swing, uma biblioteca gráfica oficial inclusa em qualquer JRE ou JDK. O look-and-feel do Swing é único em todas as plataformas onde roda, seja Windows, Linux ou outra. Isso implica que a aplicação terá exatamente a mesma interface (cores, tamanhos etc). Com Swing, não importa qual sistema operacional, qual resolução de tela, ou qual profundidade de cores: sua aplicação se comportará da mesma forma em todos os ambientes. (CAELUM, 2016).

Neste ponto a API Swing mostrou-se de grande utilidade, pois há uma função para gerar janelas e gerenciar os arquivos para salvar/carregar. Fez se uso de um objeto do tipo JFileChooser para essa parte do programa.

4.3.2 Testes

Os testes ocorrem em paralelo com a codificação, sempre que se implementa um novo método, este é testado. Esses testes foram divididos em dois estágios: bancada e máquina.

4.3.2.1 Em Bancada

O objetivo é validar a questão de comunicação entre os periféricos e o tempo de acionamento.

Para esses testes foi montada uma bancada, que era composta por três Arduinos. Dois deles simulavam o AD_104 criando pesos aleatórios e o outro o controlador de saídas.

Para simular o AD_104 e permitir que o software desenvolvido “converse” com esse, fez-se um estudo em cima deste módulo e de suas especificações. No quadro 1 de comandos AD_104 são apresentados os principais comandos enviados ao módulo implementado no Arduino.

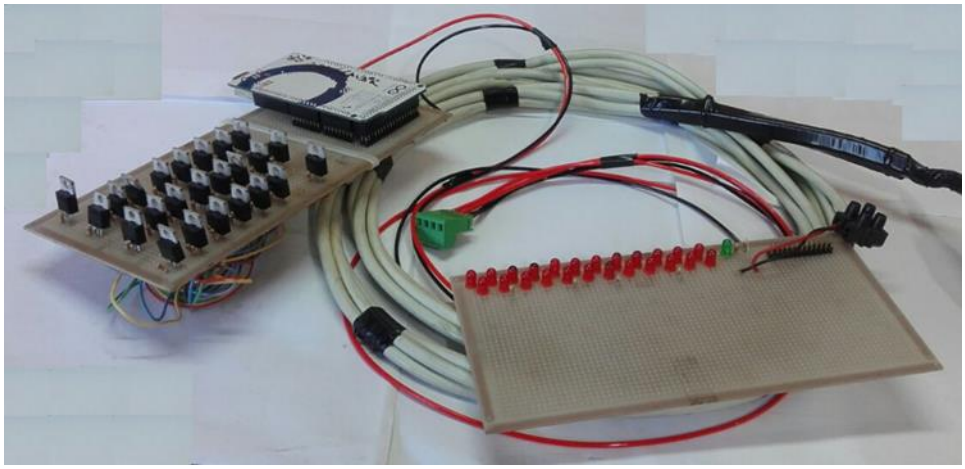
Quadro 1 – Comandos enviados ao módulo AD_104

Comando	**Função**
BDR(...)	Que define a taxa do Baund Rate, no caso do projeto por padrão sempre é adotado (BDR115200).
S(...)	Define o endereço de acesso ao módulo AD_104, no projeto por padrão foi adotado o (S31).
COF(...)	Que define o formato de sada de medidas, por definição é usado o (COF131), que impone o envio ininterrupto de pesos (1), que o sistema mande o peso bruto (3) medido e medidas sejam estáveis (1).
STP	Comando para cessar o envio de medidas.
CDL	Zeramento da pesagem, este comando é utilizado para transferir o valor de pesagem atual para a memória de zero.

Fonte: HBM.

Por não haver nenhum controlador de saída na empresa, optou-se por construir um com finalidade de teste. A figura 15 demonstra esse sistema, os *leds* indicam as saídas acionadas e a parte que possui o Arduino Mega2560 representa o controlador de saídas.

Figura 16 - Controlador de saídas para teste



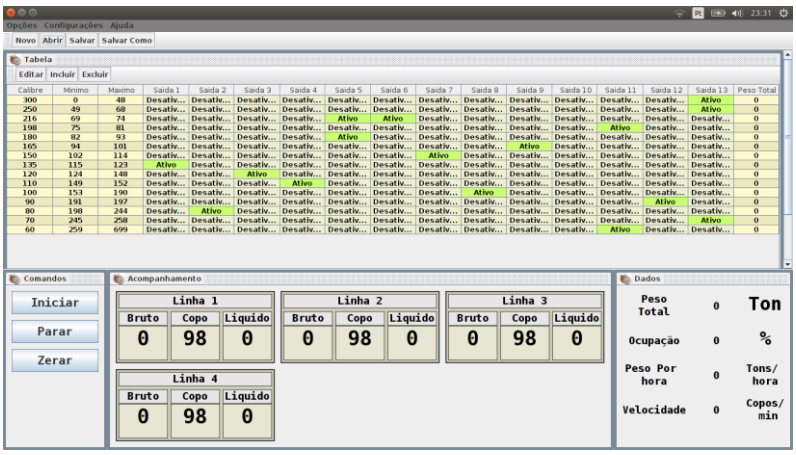
Fonte: Do Autor.

Os testes se mostraram satisfatórios pois foi possível comunicar com todos os modulos, porém não se tinha uma noção correta de funcionamento, pois o tempo de acionamento era muito pequeno.

Com relação a gravação e leitura de arquivos o software gera arquivos no formato desejado e seguindo o padrão XML, sem defeitos relatados ao abrir/salvar esses arquivos.

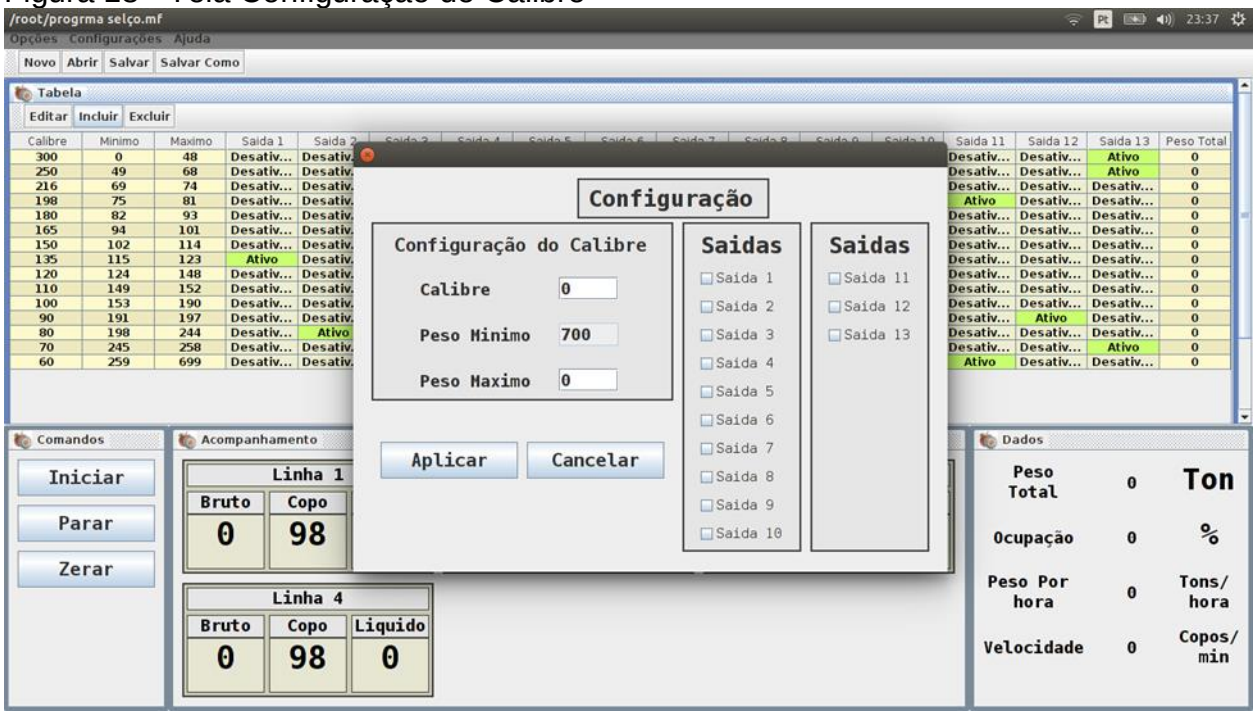
Já em relação à interface o desenvolvimento foi feito juntamente com o cliente e técnicos, os quais operariam o software. O resultado das telas pode ser visto nas figuras a seguir:

Figura 17 - Tela principal



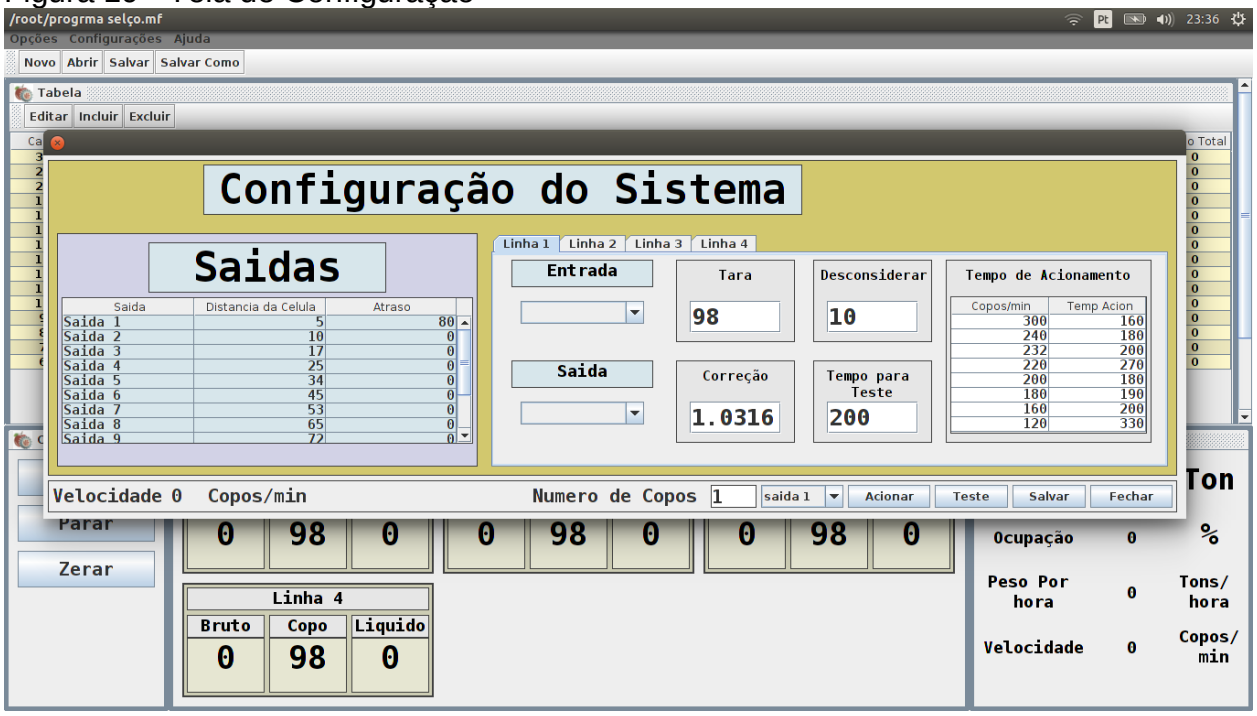
Fonte: Do Autor.

Figura 18 - Tela Configuração de Calibre



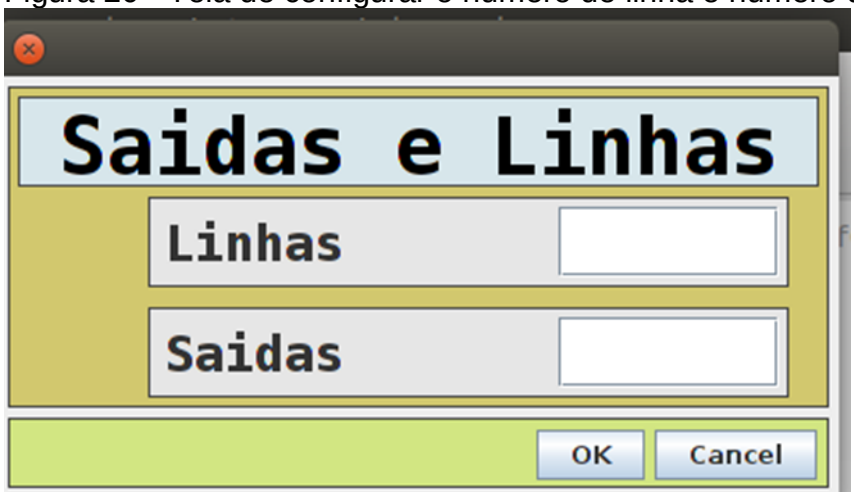
Fonte: Do Autor.

Figura 19 - Tela de Configuração



Fonte: Do Autor.

Figura 20 - Tela de configurar o número de linha e número de saídas



Fonte: Do Autor.

4.3.2.2 Em Máquinas

Fizeram-se os testes em dois classificadores distintos. O primeiro em fase de produção e o segundo pertencente a Agrícola Catarinense, cedido pelo Sr. Celso.

O classificador que está em produção no galpão da empresa é de duas linhas e dez saídas. Neste testou-se o sistema durante mais tempo, apenas com bolas de tênis de pesos diferentes usadas para simular frutos. Visualizaram-se determinadas funcionalidades e retrabalhos – por exemplo – analisou-se a distância da célula de carga até as saídas e tempo de acionamento.

Já na Agrícola Catarinense só era possível usar a máquina fora de horário de expediente com a presença de um técnico da MF. Mesmo testado com frutos não foi desenvolvido um método preciso de avaliação. Sugeriu-se então, que o sistema fosse testado em um turno de trabalho dentro da Agrícola Catarinense, pois esta possui um controle de qualidade e assim, conseguiríamos avaliar se o sistema estava realmente mandando os frutos para a saída correta. Devido a falta de tempo, até a data da escrita deste relatório não foram realizados testes durante a produção para avaliar o sistema.

5 FIRMWARE PARA O CONTROLADOR

Para o desenvolvimento do *firmware* do microcontrolador embarcado, devemos ter em mente as reais necessidades e conhecer o microcontrolador que se irá empregar. A empresa trabalha com um controlador de saídas acoplado a um Arduino em todas as máquinas de classificação.

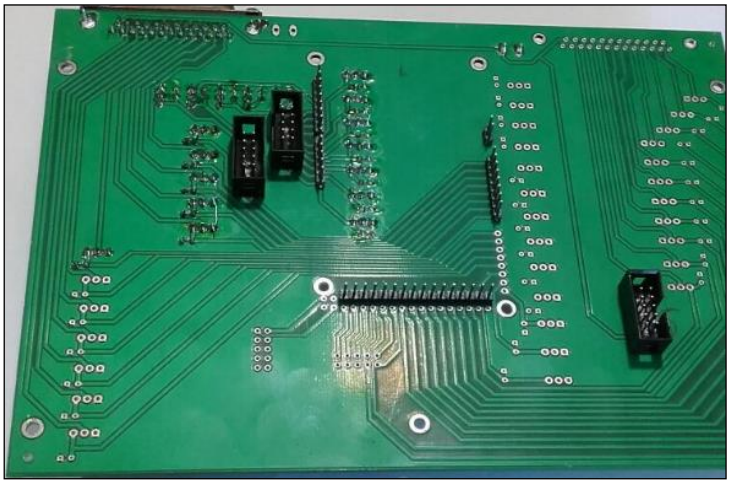
O modelo usado pelo controlador de saída é o Arduino Mega2056 que possui um microcontrolador ATmega2560, possui 54 portas digitais (I/O), 4 UARTs (portas seriais via hardware), um *clock* de 16 MHz, um conector de energia do tipo banana e uma conexão fácil com o PC através de uma interface USB. (ARDUINO, 2016).

Por se tratar de um sistema simples o qual irá receber dados e atuar, não foi usada nenhuma metodologia específica para o seu desenvolvimento. Realizou-se um estudo em relação ao controlador de saídas visando entender quais portas do Arduino eram usadas. A partir disso criou-se um conjunto de códigos para configurar e atuar.

5.1 CONTROLADOR DE SAÍDAS

O controlador de saídas possui um conjunto de pinos fixos ligados às portas do Arduino, fixando o firmware nesse mesmo conjunto de portas. A figura 20 a seguir mostra os pinos onde o Arduino é conectado.

Figura 21 - Placa do controlador de saída



Fonte: MF Máquinas.

O circuito de atuação conta com MOSFETs (Metal Oxide Semiconductor Field Effect Transistor) definido por BOYLESTAD como transistores controlados por tensão requerendo uma corrente de entrada muito pequena, por possuírem alta impedância de entrada (BOYLESTAD, 1984), de canal N. Na atual configuração, é preciso uma linha com tensão de 24 volts percorrendo a máquina para o acionamento das bobinas.

Sugere-se a formulação de um circuito novo, dimensionado pelo consumo real das bobinas e deste modo, selecionar o transistor ideal. A chave aparentemente é mudar o tipo de topologia (independente de ser um TBJ ou um FET) de acionamento, para o tipo P, aterrando-se as bobinas. Isto daria mais segurança ao funcionamento, evitando curtos-circuitos e queima de componentes.

5.2 CODIFICAÇÃO DE MENSAGENS

Criou-se um sistema de códigos para identificar qual tipo de função está sendo ativada no Arduino, a ideia é poder por meio de comandos simples configurar o controlador de saídas para diferentes máquinas. Esses códigos são indicados no quadro 2.

Quadro 2 – Códigos de comando para o Arduino

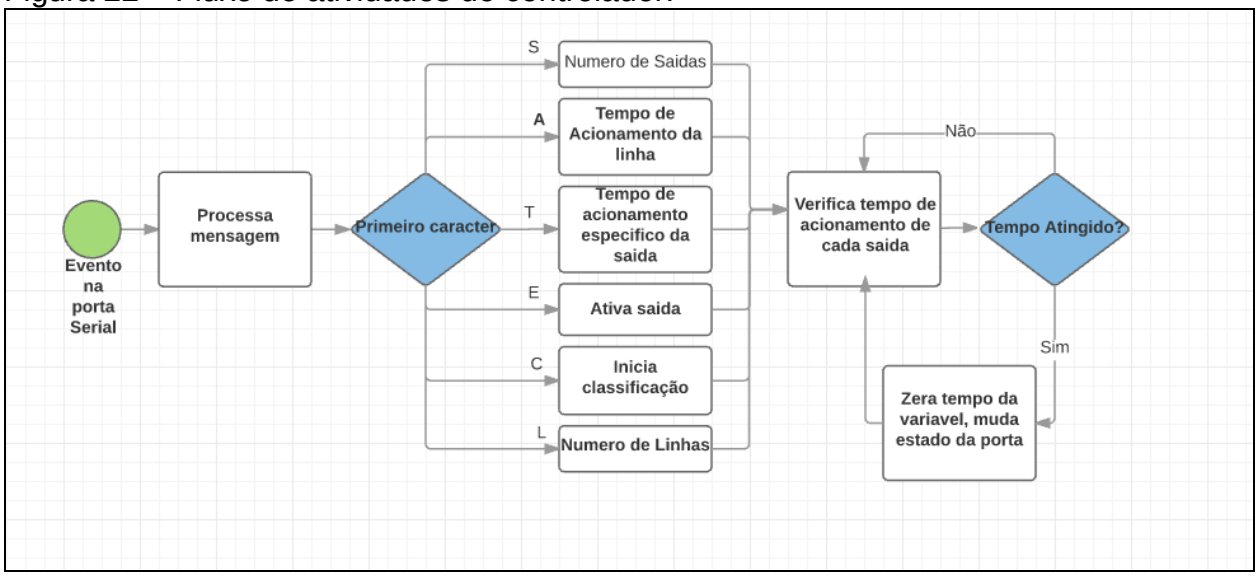
Comando	Função
L(...)	Número de linhas da máquina. Ex.: L2 (máquina de 2 linhas);
S(...)	Número de saídas da máquina. Ex.: S3 (máquina de 3 saídas);
A(...)	Tempo de acionamento definido para cada linha. Ex.: A10201 (tempo de acionamento da linha 1 de 201 milisegundos);
T(...)	Tempo de acionamento definido para cada linha. Ex.: T03021 (tempo de acionamento da saída 3 é de 21 milisegundos);
C	Início da classificação, este comando aciona a primeira saída de cada linha conforme define o RF 34 do quadro 1 do Apêndice A.
E(...)	Acionamento imediato da saída da linha definida. Ex.: E104 (Acionamento da saída 4 da linha 1).
F	Responsável pelo encerramento da classificação, fazendo o Arduino parar de atuar, desativando a primeira saída.

Fonte: Do Autor.

5.3 DIAGRAMA DO FIRMWARE

O diagrama visto na figura 22 representa o fluxo de informações e o comportamento do microcontrolador frente a diferentes comandos.

Figura 22 – Fluxo de atividades do controlador.



Fonte: Do Autor.

Com esse fluxograma, fica claro que a rotina principal é verificar o tempo de acionamento de saída, se superado o tempo retorna-se ao modo desativado.

5.4 CODIFICAÇÃO E VALIDAÇÃO

Tendo o fluxo de informação do programa figura 22, formulado os códigos de recepção (seção 5.2) e conhecendo as portas que podem ser usadas (seção 5.1) foi desenvolvido o *firmware*.

Após um evento na porta serial (chegada de um dado), a função serialEvent() (ARDUINO, 2015) é chamada fora do laço de repetição, possibilitando o microcontrolador de adotar uma das diferentes funções já citadas no quadro 2.

Outro ponto é a utilização da função wdt_enable() (WATCHDOG TIMER, 2014) que tem como função finalizar o controlador de saída quando recebe o comando F citado no quadro 2.

Para validar o *Firmware* desenvolvido, colocou-se a prova inicialmente em bancada com o protótipo de controlador de saída explorado na seção 4.3.2.1, visto na figura 16. Atendeu aos comandos, no entanto era difícil validar devido a velocidade de acionamento. Após esse teste, foi levado à máquina de duas linhas e dez saídas, apresentando resultados satisfatórios por atender os comandos previstos pelo *software*. Na máquina da Agrícola Catarinense atendeu a todos os comandos dentro dos tempos desejados.

6 CONCLUSÃO

Com este trabalho foi possível ter uma noção geral de como funciona uma máquina de classificação de frutos e/ou legumes, podendo observar sua importância frente ao mercado e a evolução dos mecanismos nesse segmento.

Foi explorado métodos de desenvolvimentos de *software*, sendo que é necessário um pré-entendimento do funcionamento de elementos como uma célula de carga ou um sensor fotoelétrico.

Até o presente momento o projeto não está concluído. É necessário ainda a validação dos testes em um classificador em operação real, como sugerido por Sr. Celso Kafer Marinês e conferir juntamente ao setor de qualidade (por meio de pesagem de lotes aleatórios) o resultado obtido e assim ingressar na fase de transição, apesar dos técnicos da MF já terem ciência do funcionamento do novo programa (fase transição).

É preciso alterar o circuito de atuação, como já citado, ele deixaria a máquina mais segura, evitando curto-circuitos, mau funcionamento do equipamento ou ainda danos maiores. Tal mudança tornaria a instalação e manutenção mais vantajosa por diminuir a quantidade de fios e simplificar – de maneira genérica – o sistema de atuação (sem cabo de retorno às calhas).

Um projeto promissor na área de beneficiamento de frutos, mas ainda distante, visa um sistema de classificação por imagem. Tais classificadores prometem menores máquinas com maior eficiência. Diversas empresas estrangeiras investem nessa ideia e conseguem em alguns casos dobrar a produção.

REFERÊNCIAS

ARDUINO. **Arduino MEGA 2560 e Genuino MEGA 2560**. Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>. Acesso em: 10 jun. 2016.

_____. **Download Arduino Software**. Disponível em: <https://www.arduino.cc/en/Main/Software>. Acesso em: 10 jun. 2016.

_____**Serial Event**. 2015. Disponível em: <https://www.arduino.cc/en/Tutorial/SerialEvent>. Acesso em: 12 jun. 2016.

ASTAH. Disponível em: < http://astah.net/download>. Acesso em: 10 jun. 2016.

BALSAMIQ. Web Demo Project. Disponível em: <https://webdemo.balsamiq.com/>. Acesso em: 10 jun. 2016.

BOYLESTAD, Robert L.; NASHEKSKY, Louis. **Dispositivos eletrônicos e teoria de circuitos.** v. 6. Prentice-Hall do Brasil, 1984.

CAELUM. Apostila Laboratório Java com testes, XML e Design Patterns. **Capítulo 5**: interfaces gráficas com Swing. Disponível em: <http://www.caelum.com.br/apostila-java-testes-xml-design-patterns/interfaces-graficas-com-swing/>. Acesso em: 12 jun. 2016.

DCA. Departamento de Engenharia de Computação e Automação Industrial. Faculdade de Engenharia Elétrica e de Computação da Universidade de Campinas. 2000. Disponível em: <http://www.dca.fee.unicamp.br/cursos/PooJava/classes/conceito.html>. Acesso em: 7 jul. 2016.

DEITEL. Harvey M. **Java**: como programar. 6. ed. São Paulo: Pearson Prentice Hall, 2009.

DENARDIN, Gustavo Weber. **Microcontroladores**. Joinville, 2014. Disponível em: <http://www.joinville.udesc.br/portal/professores/eduardo_henrique/materiais/apostila_micro_do_Gustavo_Weber.pdf>. Acesso em: 22 jun. 2016.

DEVMEDIA. **Utilizando a API RXTX para manipulação da serial – parte III**. 2007. Disponível em: <http://www.devmedia.com.br/utilizando-a-api-rxtx-para-manipulacao-da-serial-parte-iii/7171>. Acesso em: 22 jun. 2016.

DUNN, William C. **Fundamentos de instrumentação industrial e controle de processos**. Porto Alegre: Bookman, 2013.

ECLIPSE. Disponível em: < http://www.eclipse.org/downloads/>. Acesso em 10 jun. 2016.

FILIPEFLOP. **O que é arduino?**. Florianópolis, 2014. Disponível em: <http://blog.filipeflop.com/arduino/o-que-e-arduino.html>. Acesso em: 22 jun. 2016.

GUEDES, Gilleanes T. A. **UML 2**: uma abordagem prática. 2. ed. São Paulo: Novatec, 2011.

HBM. Measurement With Confidence**. AD 104**. Germany. Disponível em: <http://www.primaxbalancas.com.br/site200801a/manuais/AD104.pdf >. Acesso em: 7 jul. 2016.

_____. **Z6**. Germany. Disponível em: <http://www.hbm.com.pl/pdf/b1010.pdf>. Acesso em: 7 jul. 2016.

IEEE COMPUTER SOCIETY. **About SWEBOK**, 2004. Disponível em: <https://www.computer.org/web/swebok/>. Acesso em: 7 jul. 2016.

JAVA. Disponível em: <https://www.java.com/pt_BR/about/>. Acesso em: 22 jun. 2016.

LIBREOFFICE. The Document Foundation. Disponível em: <https://pt-br.libreoffice.org/>. Acesso em: 10 jun. 2016.

LUCIDCHART. Disponível em: <https://www.lucidchart.com/>. Acesso em: 10 jun. 2016.

MACORATTI, José Carlos. **Orientação a objetos**: conceitos básicos. Disponível em: <http://www.macoratti.net/net_oocb.htm>. Acesso em: 7 jul. 2016.

MARTIN NETO, Ladislau. **A automação agropecuária**. Brasília: SENAR: Agricultura de Precisão, 2013. Disponível em: <http://www.senar.org.br/agricultura-precisao/artigo-a-automacao-agropecuaria/>. Acesso em: 7 jul. 2016.

MF MÁQUINAS. **Home page**. Fraiburgo, 2016. Disponível em: <http://www.maquinasmf.com.br>. Acesso em: 07 set. 2016.

NATIONALS INSTRUMENTS. **Conceitos Gerais de Comunicação Serial**. Austin, 2015. Disponível em: <http://digital.ni.com/public.nsf/allkb/32679C566F4B9700862576A20051FE8F>. Acesso em: 22 jun. 2016.

OMRON. **Photoecletric sensors**. Germany. Disponível em: <https://www.fa.omron.com.cn/data_pdf/cat/e3fa_e3ra_e3fb_e3rb_e424-e1_1_11_csm1006569.pdf?id=3130)>. Acesso em: 7 jul. 2016.

OPENUP/BASIC. Disponível em: <http://epf.eclipse.org/wikis/openuppt/index.htm>. Acesso em: 30 jul. 2016.

PINHEIRO, José Mauricio Santos. **Sistemas de automação**, 2004. Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_sistemas_automacao.php>. Acesso em: 4 jul. 2016.

PINTA PROJECT. Disponível em: <https://pinta-project.com/pintaproject/pinta/>. Acesso em: 10 jun. 2016.

PINTO, Jhone Ricardo Cima; NUNES, Fabiano de Lima; VIERO, Carlos Frederico. Avaliação dos ganhos de produtividade e redução de custos gerados pela automação de processo em uma empresa calçadista: um estudo de caso. **Revista Espacios**, Caracas, v. 36, n. 16, 2015. Disponível em: <http://www.revistaespacios.com/a15v36n16/15361606.html>. Acesso em: 7 jul. 2016.

PRECISO ESTUDAR SEMPRE. **O que é um listener? Como fazer um?**. 2015. Disponível em: <http://precisoestudarsempre.blogspot.com.br/2015/06/o-que-e-um-listener-como-fazer-um.html>. Acesso em 7 jul. 2016.

RAMALHO, G.L.B. 2003. Descrição de um Sistema Experimental de Visão Artificial Aplicado à Inspeção Automática de Frutas na Pós-Colheita. 103p. **Monografia** (Especialização em Automação Industrial). Fortaleza: Universidade Estadual do Ceará

SILVEIRA, Leonardo; LIMA, Weldson Q.. **Um breve histórico conceitual da automação industrial e reder para automação industrial**. Natal: Universidade Federal do Rio Grande do Norte: Programa de Pós-Graduação em Engenharia Elétrica, 2003. Disponível em: <http://www.dca.ufrn.br/~affonso/FTP/DCA447/trabalho1/trabalho1_13.pdf>. Acesso em: 7 jul. 2016.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

TONSIG, Sérgio Luiz. **Engenharia de software**: análise e projeto de sistemas. 2. ed. Rio de Janeiro: Editora Ciência Moderna, 2008.

WAZLAWICK, Raul Sidney. **Engenharia de software**: conceitos e práticas. Rio de Janeiro: Elsevier, 2013.

WATCHDOG TIMER. **Arduino Watchdog Timer (WDT) exemple code**. 2014. Disponível em: <https://bigdanzblog.wordpress.com/2014/10/24/arduino-watchdog-timer-wdt-example-code/>. Acesso em: 9 jul. 2016.

W3SCHOOLS.COM. 2016. Disponível em: <http://www.w3schools.com/xml/xml_whatis.asp>. Acesso em: 22 jun. 2016.

APÊNDICE A – Requisitos Funcionais e Não Funcionais

Aqui são apresentados os quadros de requisitos funcionais e não funcionais do software.

Quadro 1 – Requisitos não funcionais

Requisitos não funcionais	
RNF_01	O gerenciador deve ser implementado para uma arquitetura PC, independente do S.O. usado (Win 7, Win 10 e Linux).
RNF_02	O gerenciador deve dar suporte a comunicação RS-232 e atender os protocolos do módulo AD104 da HBM.
RNF_03	O gerenciador não pode perder nenhuma mensagem vinda e/ou enviada pela porta serial.
RNF_05	O gerenciador deve se comunicar com um módulo de gerenciamento de acionamentos através do protocolo RS-232.
RNF_06	O gerenciador deve ter uma interface de fácil compreensão e intuitiva.
RNF_07	O gerenciador deve decidir em tempo qual saída o fruto/legume deve cair.
RNF_08	Os gerenciadores gerados em arquivos do programa devem seguir o padrão XML.

Fonte: Do Autor.

Quadro 2 – Requisitos funcionais

Requisitos funcionais	
RF_01	O sistema será responsável pela recepção de dados, processamento e atuação em uma máquina de classificar frutos/legumes.
RF_02	A interação entre usuário e máquina se dará através de uma tabela, onde nessa tabela deve conter as seguintes colunas: Calibre, peso mínimo, peso máximo, saídas ativas/desativas e o peso total do calibre.
RF_03	Referente à tabela citada no RF_02, os campos Calibre, peso mínimo e peso máximo: ● Devem ser números inteiros; ● Não podem ser nulos; ● O peso mínimo não pode ser maior que o peso máximo; ● Eles não podem se repetir dentro da tabela; ● O peso mínimo e máximo será dado em gramas; ● O usuário irá editar apenas o peso máximo e o calibre;

	● O peso mínimo se ajustará uma grama acima do peso máximo do calibre anterior. O primeiro calibre da tabela (primeira linha da tabela) terá peso mínimo igual à zero;
RF_04	Referente ao RF_02 os campos de saídas: ● Os campos serão editados pelo usuário; ● Podem adotar apenas dois valores: Ativo e Desativo;
RF_05	Referente ao RF_02 o campo peso total, irá começar com zero e será a soma total do peso daquele calibre.
RF_06	Referente ao RF_02 os campos da tabela podem sofrer 3 ações: ● Adicionar nova linha; ● Editar linha; ● Excluir linha;
RF_07	Referente ao RF_06 a opção *Adicionar nova linha*, irá colocar na última linha da tabela uma nova linha, onde os campos devem ser preenchidos de acordo com o RF_03, e RF_04.
RF_08	Referente ao RF_06 a opção *Editar linha* deve respeitar os requisitos dos RF_03 e RF_04.
RF_09	Referente ao RF_06 as opções *Editar linha* e *Excluir linha*, tem que ter uma linha selecionada.
RF_10	Deve-se ter a opção de salvar a tabela e carregar a tabela. Estas opções permitem a navegação por um gerenciador de arquivos.
RF_11	O formato gerado pelo arquivo da tabela deve ser do tipo "*mf*".
RF_12	O usuário deve ter a opção de: ● Iniciar o processo de classificação; ● Parar o processo de classificação; ● Zerar pesagem, equivalente a uma taragem.
RF_13	O usuário deve ter acesso a 4 informações: ● Peso total: peso total de frutas/vegetais passados na máquina; ● Velocidade de operação da máquina: velocidade dada em número de frutos/vegetais passados por minuto pela máquina; ● Ocupação da máquina: baseado nas pesagens é criado uma estimativa da ocupação da máquina; ● Peso por hora: estimativa de frutos/vegetais que serão classificados baseados na ocupação da máquina e pela sua velocidade usada no momento;
RF_14	O usuário deve visualizar: ● O peso medido no momento; ● O peso da tara (peso do suporte que carrega o fruto); ● O peso líquido (diferença entre o peso atual e o peso da tara (suporte));
RF_15	Deve-se ter um painel de configuração onde apenas o técnico responsável da empresa tenha acesso.
RF_16	Referente ao RF_15, nesse painel deve-se poder configurar o meio de acesso ao

	módulo de pesagem para cada linha.
RF_17	Referente ao RF_15, nesse painel deve-se poder configurar o meio de acesso ao módulo de atuação de cada linha.
RF_18	Referente ao RF_15, nesse painel deve-se poder configurar um valor que o sistema irá desconsiderar na pesagem de cada linha onde esse valor tem que ser um número inteiro e positivo.
RF_19	Referente ao RF_15, nesse painel deve-se poder configurar o peso para taragem (referente ao peso do copo que carrega o fruto) de cada linha esse valor tem que ser um número inteiro e positivo.
RF_20	Referente ao RF_15, nesse painel deve-se poder configurar um fator de correção para cada linha, que pode adotar números não inteiros positivos, esse valor será multiplicado pelo peso atual para que possa ser feito um ajuste fino.
RF_21	Referente ao RF_15 deve-se ter uma tabela para cada linha, com as colunas copos por minuto e tempo de acionamento sendo que esses valores tem que ser números inteiros e positivos.
RF_22	As colunas (copos por minutos e tempo de acionamento) referentes ao RF_21 serão responsáveis pelas referências para o cálculo do tempo de acionamento dos atuadores de cada linha, levando em conta a velocidade atual da máquina (RF_13).
RF_23	Referente ao RF_15, no painel deve conter uma tabela com as colunas: ● Saídas; ● Distância; ● Atraso;
RF_24	Referente ao RF_23, a coluna *Saída* terá uma linha para cada saída, esse campo será imutável e fornecido pelo sistema.
RF_25	Referente ao RF_23, a coluna *Distância* é responsável pelo número de intervalos até a saída propriamente dita sendo que: ● Só pode adotar números inteiros e positivos; ● O número da linha atual não pode ser menor que o da linha anterior;
RF_26	Referente ao RF_23 a coluna *Atraso* deve adotar apenas números inteiros positivos, sendo esse número responsável por um tempo fixo de acionamento que será somado ao tempo calculado através da velocidade da máquina (referente ao tempo de acionamento citado no RF_22).
RF_27	Deve existir na tela referente ao RF_15 um campo para o tempo de acionamento de cada linha (que será usado para as funções *Teste* e *Acionar*) que só aceita valores inteiros e representará o tempo em milisegundos.
RF_28	Deve existir na tela referente ao RF_15 uma janela que receberá o número de copos para a função teste que aceitará apenas números inteiros.
RF_29	Deve existir na tela referente ao RF_15 uma janela de seleção de saídas, onde essas saídas serão geradas pelo sistema.
RF_30	Existirá um indicador de copos por minuto para facilitar a validação da função teste.

RF_31	Em relação ao RF_15 ainda deve-se existir 4 opções de ação: ● Salvar: Gera/Edita o arquivo Conf. a partir das opções setadas na tela de configuração (RF_15) e valida os campos, caso os campos não atendam os itens referentes aos requisitos funcionais: RF_16, RF_17, RF_18, RF_19, RF_20, RF_21, RF_22, RF_23, RF_24, RF_25, RF_26, RF_27 e RF_28. Caso algum requisito não esteja de acordo indicará o erro; ● Fechar: fecha a tela de configuração; ● Teste: Responsável pela função teste; ● Acionar: Responsável pela função acionar;
RF_32	Referente ao RF_30 a opção *Teste* irá pegar o número de copos (RF_28) e manterá o atuador acionado para cada copo por um tempo pré-definido (RF_27) para saída (RF_29) cada linha e ainda indicará a velocidade de copos por minutos.
RF_33	Referente ao RF_30 a opção *Acionar* irá manter acionada a saída selecionada (RF_29) por um tempo definido (RF_27).
RF_34	A primeira saída deve ficar sempre acionada mudando para o estado desacionada caso a saída selecionada não seja a primeira.

Fonte: Do Autor.

APÊNDICE B – Detalhamento dos Casos de uso

Neste apêndice serão apresentados os quadros detalhados de cada caso de uso.

Quadro 1 – Caso de uso Parar

Nome do Caso de Uso	**CDU01 - Parar**
Caso de uso Geral	
Ator Principal	Usuário
Atores Secundarios	
Resumo	Responsável por parar a classificação de frutos
Pré-Condições	O sistema tem que estar classificando
Pós-Condições	O sistema para de classificar
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Usuário solicita a parada o sistema.	
	2. Mensagem enviada ao Processamento solicitando parada de classificação.
	3. Sistema retorna que parou o processo de classificação.
	4. Indicativo de classificação fica desativado.
Restrições/Validações	1. O sistema não pode estar classificando (Iniciar).
Fluxo Alternativo - Sistema já está Parado	
Ações do Ator	Ações do Sistema
	1. Mensagem avisando que o sistema não está classificando (o sistema já está parado).
Restrições/Validações	
Fluxo de Exceção - Erro comunicação	
Ações do Ator	Ações do Sistema
	1. Mensagem avisando que ocorreu um erro

	no envio da mensagem de parada.
	2. Sistema fecha comunicação com módulos.
Restrições/Validações	

Fonte: Do Autor.

Quadro 2 – Caso de uso Zerar

Nome do Caso de Uso	**CDU02 - Zerar**
Caso de uso Geral	
Ator Principal	Usuário
Atores Secundários	
Resumo	Responsável pelo zeramento (taragem) da pesagem do sistema.
Pré-Condições	O sistema tem que estar parado.
Pós-Condições	O sistema tem a pesagem tarada.
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Usuário solicita a zerar (tarar) a pesagem.	
	2. Mensagem enviada ao Processamento solicitando zerar (tarar) a pesagem.
	3. Sistema retorna que zerou (tarou) a pesagem.
Restrições/Validações	1. O sistema não pode estar classificando (Iniciar).
Fluxo Alternativo - Sistema não está Parado	
Ações do Ator	Ações do Sistema
	1. Mensagem avisando que o sistema não está parado.
Restrições/Validações	
Fluxo de Exceção - Erro comunicação	
Ações do Ator	Ações do Sistema
	1. Mensagem avisando que ocorreu um erro no envio da mensagem de Zeramento.

	2. Sistema fecha comunicação com módulos.
Restrições/Validações	

Fonte: Do Autor.

Quadro 3 – Caso de uso Iniciar

Nome do Caso de Uso	**CDU03 - Iniciar**
Caso de uso Geral	
Ator Principal	Usuário
Atores Secundários	
Resumo	Inicia o processo de classificação
Pré-Condições	O sistema tem que estar parado e a tabela de dados não pode estar vazia.
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Usuário solicita a inicialização da classificação.	
	2. Mensagem enviada ao Processamento solicitando classificação.
	3. Sistema inicia a classificação de frutos.
	4. Sistema retona idicativo de estar classificando.
Restrições/Validações	1. O sistema não pode estar classificando (Iniciar).
Fluxo Alternativo - Sistema já Iniciado	
Ações do Ator	Ações do Sistema
	1. Mensagem avisando que o sistema já está classificando.
Restrições/Validações	
Fluxo de Exceção - Erro comunicação	
Ações do Ator	Ações do Sistema
	1. Mensagem avisando que ocorreu um erro

	no envio da mensagem de Iniciar.
	2. Sistema fecha comunicação com módulos.
Restrições/Validações	

Fonte: Do Autor.

Quadro 4 – Caso de uso Editar Tabela

Nome do Caso de Uso	**CDU04 - Editar Tabela**
Caso de uso Geral	
Ator Principal	Usuário
Atores Secundários	
Resumo	Esse caso de uso é responsável por adicionar linha, editar linha, excluir linha e nova tabela.
Pré-Condições	
Pós-Condições	A tabela será editada.
Fluxo Alternativo - Adicionar Linha	
Ações do Ator	Ações do Sistema
1. Acionar a opção de Incluir linha	
	2. Sistema habilita campo de calibre, de peso máximo e de saídas.
	3. Adicionar nova linha na tabela.
Restrições/Validações	1. O peso máximo não pode ser menor ou igual ao peso mínimo. 2. O calibre não pode se repetir na tabela. 3. Esses campos não podem ser nulos.
Fluxo Alternativo - Editar Linha da Tabela	
Ações do Ator	Ações do Sistema
1. Selecionar uma linha.	
2. Acionar a opção de Editar linha da tabela.	
	3. Sistema habilita campo de calibre, de peso máximo e saídas.
	4. Editar a linha nova linha na tabela.
Restrições/Validações	1. Uma linha da tabela tem que estar

	selecionada. 2. O peso máximo não pode ser menor ou igual ao peso mínimo. 3. O calibre não pode se repetir na tabela. 4. Esses campos não podem ser nulos.
Fluxo Alternativo - Excluir Linha	
Ações do Ator	Ações do Sistema
1. Selecionar uma linha.	
2. Acionar a opção de Excluir linha.	
	3. Sistema elimina a linha selecionada.
Restrições/Validações	1. Uma linha da tabela tem que estar selecionada.

Fonte: Do Autor.

Quadro 5 – Caso de uso Tabela

Nome do Caso de Uso	**CDU05 - Tabela**
Caso de uso Geral	
Ator Principal	Usuário
Atores Secundários	
Resumo	Armazenar o calibre, peso mínimo, peso máximo e quais saídas devem ser acionadas.
Pré-Condições	
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
	1. Receber os dados referentes ao calibre, peso mínimo, peso máximo e saídas.
Restrições/Validações	

Fonte: Do Autor.

Quadro 6 – Caso de uso Salvar Tabela

Nome do Caso de Uso	**CDU06 - Salvar Tabela**
Caso de uso Geral	
Ator Principal	Usuário

Atores Secundários	
Resumo	Responsável por guardar os dados da tabela em um arquivo externo.
Pré-Condições	A tabela não pode estar vazia.
Pós-Condições	A tabela é salva no formato “.mf”
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Solicitação para guardar os dados da tabela.	
	2. Verificar se a tabela não está vazia.
	3. Solicitar o local para salvar a tabela.
4. Indica local de gravação do arquivo	
	5. Salvar a tabela em um arquivo do tipo “.mf”
Restrições/Validações	1. A tabela não pode estar vazia.

Fonte: Do Autor.

Quadro 7 – Caso de uso Carregar Tabela

Nome do Caso de Uso	**CDU07 - Carregar Tabela**
Caso de uso Geral	
Ator Principal	Usuário
Atores Secundários	
Resumo	Carrega a tabela.
Pré-Condições	
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Solicitação para carregar uma tabela.	
2. Localizar o arquivo da tabela.	
	3. Carregar os dados do arquivo para a tabela.

Restrições/Validações	1. O caminho para o arquivo da tabela tem que ser valido.

Fonte: Do Autor.

Quadro 8 – Caso de uso Salvar Configurações

Nome do Caso de Uso	**CDU08 - Salvar Configurações**
Caso de uso Geral	
Ator Principal	Técnico
Atores Secundários	
Resumo	Salvar as configurações da maquina.
Pré-Condições	As configurações não podem ser nulas
Pós-Condições	Guardar as informações em um arquivo do tipo “.conf”
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Solicitação para guardar os dados de configuração.	
	2. Verificar se os dados não validos.
	3. Salvar os dados em um arquivo no diretório base com o nome “conf.conf”
Restrições/Validações	1. Os dados tem que ser válidos.

Fonte: Do Autor.

Quadro 9 – Caso de uso Editar Linha

Nome do Caso de Uso	**CDU09 - Editar Linha**
Caso de uso Geral	
Ator Principal	Técnico
Atores Secundários	
Resumo	Edição dos dados de uma linha da máquina.
Pré-Condições	
Pós-Condições	
Fluxo Alternativo - Porta de comunicação de entrada	

Ações do Ator	Ações do Sistema
	1. Fornecer as portas de comunicação do modulo AD_104.
2. Selecionar uma porta de comunicação de entrada.	
	3. Mandar mensagem para Configuração dizendo qual porta foi selecionada.
Restrições/Validações	1. O módulo AD_104 tem que estar conectado ao computador.
Fluxo Alternativo - Porta de comunicação de saída	
Ações do Ator	Ações do Sistema
	1. Fornecer as portas de comunicação dos atuadores.
2. Selecionar uma porta de comunicação de saída.	
	3. Mandar mensagem para Configuração dizendo qual porta foi selecionada.
Restrições/Validações	1. O módulo de atuadores tem que estar conectado ao sistema.
Fluxo Alternativo - Tara	
Ações do Ator	Ações do Sistema
1. Informar o valor da tara.	
	2. Validar o valor da tara.
	3. Mandar mensagem para Configuração dizendo qual valor da tara.
Restrições/Validações	1. A tara tem que ser um número inteiro igual ou maior que zero.
Fluxo Alternativo - Desconsiderar	
Ações do Ator	Ações do Sistema
1. Informar o valor a ser desconsiderado.	
	2. Validar o valor a ser desconsiderado.
	3. Mandar mensagem para Configuração dizendo qual valor da a ser desconsiderado.
Restrições/Validações	1. O valor tem que ser um número inteiro

	igual ou maior que zero.
Fluxo Alternativo - Fator de correção	
Ações do Ator	Ações do Sistema
1. Informar o valor para a correção.	
	2. Validar o valor de correção.
	3. Mandar mensagem para Configuração dizendo qual valor de correção.
Restrições/Validações	1. O valor tem que ser um número real diferente de zero.
Fluxo Alternativo - Velocidade por tempo de acionamento	
Ações do Ator	Ações do Sistema
1. Selecionar uma velocidade de operação.	
2. Selecionar o tempo de acionamento para aquela velocidade.	
	3. Validar o valor da velocidade e do tempo de acionamento.
	4. Mandar mensagem para Configuração dizendo qual valor o valor da velocidade e do tempo de acionamento.
Restrições/Validações	1. Os valores (velocidade e do tempo de acionamento) tem que ser números inteiros maiores que zero.

Fonte: Do Autor

Quadro 10 – Caso de uso Configuração

Nome do Caso de Uso	**CDU10 - Configuração**
Caso de uso Geral	
Ator Principal	Técnico
Atores Secundários	
Resumo	Armazenar os dados referentes a configuração da máquina: Portas de comunicação, distância dos copos, tempo de acionamento, tara, valor a ser desconsiderado e fator de correção.
Pré-Condições	

Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
	1. Receber os dados referentes as portas de comunicação, distância dos copos, tempo de acionamento, tara, valor a ser desconsiderado e fator de correção.
Restrições/Validações	

Fonte: Do Autor.

Quadro 11 – Caso de uso Editar Saída

Nome do Caso de Uso	**CDU11 - Editar Saida**
Caso de uso Geral	
Ator Principal	Técnico
Atores Secundários	
Resumo	Edição das saídas da máquina, como a distância de cada copo e o tempo mínimo de acionamento da saída.
Pré-Condições	
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Selecionar o número de copos a partir da base da célula de carga até a saída.	
	2. Validar o valor da distância de copos.
	3. Mandar mensagem para Configuração dizendo qual valor da distância de copos.
Restrições/Validações	1. O número de copos tem que ser um número inteiro maior que zero.

Fonte: Do Autor.

Quadro 12 – Caso de uso Testar

Nome do Caso de Uso	**CDU12 - Testar**
Caso de uso Geral	

Ator Principal	Técnico
Atores Secundários	
Resumo	Responsável pela parte de testes da máquina, para facilitar a dedução do tempo de acionamento com a velocidade
Pré-Condições	
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema
1. Selecionar a saída a ser testada.	
2. Selecionar o número de copos para teste.	
	3. Validar o número de copos.
	4. Mandar mensagem para Processamento dizendo qual saída e qual o número de copos.
Restrições/Validações	1. O número de copos tem que ser um número inteiro maior que zero.

Fonte: Do Autor.

Quadro 13 – Caso de uso Processamento

Nome do Caso de Uso	**CDU13 - Processamento**
Caso de uso Geral	
Ator Principal	AD_104
Atores Secundários	Atuadores e Usuário
Resumo	Responsável pelo recebimento e envio de dados ao AD_104, verificação na tabela qual é a saída apropriada, e o casamento das configurações para o envio de mensagem para os atuadores.
Pré-Condições	Comunicação estabelecida com o AD_104 e com os Atuadores, tabela não nula e configurações não nula.
Pós-Condições	
Fluxo Principal	
Ações do Ator	Ações do Sistema

	1. Receber mensagem de Iniciar.
	2. Carregar os dados da Configuração.
	3. Carregar os dados da Tabela.
	4. Iniciar Comunicação com AD_104.
	5. Iniciar comunicação com Atuadores.
6. Manda dado da pesagem.	
	7. Processar dado vindo, cruzando com as informações da configuração.
	8. Mandar mensagem para Atuadores dizendo qual saída deve ser acionada.
Restrições/Validações	1. O módulo AD_104 tem que estar conectado ao sistema. 2. Atuadores devem estar ligados ao sistema.
Fluxo Alternativo - Parar	
Ações do Ator	Ações do Sistema
	1. Receber mensagem de Parar.
	2. Enviar mensagem de parada ao AD_104.
3. Enviar confirmação de parada.	
	4. Receber mensagem, e encerra a comunicação com AD_104.
	5. Encerrar comunicação com atuadores.
	5. Mudar estado para parado.
Restrições/Validações	1. O módulo AD_104 tem que estar conectado ao computador. 2. Atuadores devem estar ligados ao sistema.
Fluxo Alternativo - Zerar	
Ações do Ator	Ações do Sistema
	1. Receber mensagem de Zerar.
	2. Iniciar comunicação com AD_104.
	3. Enviar mensagem para Zerar ao AD_104.
	4. Mudar estado para Zerado.

Restrições/Validações	1. O módulo AD_104 tem que estar conectado ao computador.
Fluxo Alternativo - Testar	
Ações do Ator	Ações do Sistema
	1. Receber mensagem de Testar.
	2. Carregar os dados da Configuração.
	3. Iniciar comunicação com Atuadores.
	4. Iniciar Comunicação com AD_104.
5. Manda dado da pesagem.	
	6. Processar dado vindo.
	7. Mandar mensagem para Atuadores dizendo qual saída deve ser acionada.
Restrições/Validações	1. O módulo AD_104 tem que estar conectado ao sistema. 2. Atuadores devem estar ligados ao sistema.
Fluxo de Exceção - Erro comunicação	
Ações do Ator	Ações do Sistema
	1. Mensagem avisando que ocorreu um erro no envio de mensagens entre Atuador e/ou AD_104.
	2. Sistema fecha comunicação com módulos.
Restrições/Validações	

Fonte: Do Autor.

APÊNDICE C – Detalhamento das Classes

Neste apêndice estão os quadros de classe do software, sendo que para facilitar a compreensão do diagrama de classe os atributos estão alocados aqui.

Quadro 1 – Classe Interface

Nome da Classe	**Interface**
Aqui a interface representa um conjunto de outras classes, onde cada uma representa uma tela do sistema, para facilitar a visualização do diagrama de telas, e tendo em mente que seus métodos são apenas para enviar mensagens para a Classe Supervisor, não é necessário a descrição dessas classes. Apenas para esclarecimento segue o nome das classes que a Interface representa: ● TelaConfiguracão; ● TelaPrincipal; ● TelaAddEditCalibre; ● TelaNumLNumS;	

Fonte: Do Autor.

Quadro 2 – Classe Supervisor

Nome da Classe	**Supervisor**
Classe responsável por fazer a ligação entre os Usuários (Usuário e/ou Técnico) com as outras classes (motivo pelo qual ela está conectada com todos as classes do sistema), é uma classe intermediaria cuja sua função principal é agrupar os comandos e evitar o acumulo de métodos nas classes de interface.	

Fonte: Do Autor.

Quadro 3 – Classe Dados

Nome da Classe	**Dados**
Resumo	
Classe responsável por receber o dado do peso bruto vindo da classe Protocolo através de um Observador, decidir em qual saída ideal para aquele dado (no caso comparar na sua base de dados o peso do fruto com o da tabela de calibres e ver em qual saída ele deve ir), e enviar uma mensagem para o controlador avisando quando acionar a saída.	
Atributos	
Nome do Atributo	Tipo do Atributo
Serial	SerialRXTX
dadoRecb	Observable
Tara	int
Desconsiderar	int

vetSaidas	int[ ]
Linha	int
Ocupação	int
vetPeso	Int[ ]
Matriz	Int[ ]
posSaida	Int[ ]

Métodos	
Nome do Método	Função
Update	Responsável por receber o dado vindo da classe Protocolo através de um observador;
Compara	Responsável por comparar o dado recebido com a base de dados e decidir qual saída é apropriada para aquele fruto;
arrumaVet	Cada vez que chega um novo dado e ele é tratado e definida a saída que ele deve ir, essa função coloca esse dado na primeira posição do vetor de saídas e consequentemente faz os outros dados andarem uma casa a frente (o vetor posSaida é o responsável por guardar a saída que deve ser acionada pelo sistema);
Analisa	Função responsável por ver quais saídas devem ser acionadas, comparando o vetor de saídas (uma vez que quando um fruto é pesado ele manda pela porta serial incrementando assim o vetor posSaida através do método arrumaVet) com o vetor que guarda a posição das saídas (usada como referência, que guarda a distância da célula de carga para cada saída, que seria o vetSaidas);
setSerialSaida	Define a porta serial de saída, ou seja, a porta que se deve enviar a mensagem para o controlador;
setMatriz	Define matriz de comparação para o método compara;
setTara	Define o valor da tara;
setCorrecao	Define o valor da correção;
setDesconsiderar	Define o valor a ser desconsiderado pela pesagem;
setSaidas	Define o vetor vetSaidas;
setLinha	Define a linha de pesagem;
getOcupacao	Retorna o percentual de ocupação naquela linha;
getVetPeso	Retorna o peso acumulado em cada calibre que foi passado

	naquela linha;

Fonte: Do Autor.

Quadro 4 – Classe ManipulaTabXML

Nome da Classe	**ManipulaTabXML**
Resumo	
Classe responsável por transformar os dados da tabela de calibres em um formato contendo apenas inteiros, facilitando assim o trabalho do método compara da classe Dados	
Atributos	
Nome do Atributo	Tipo do Atributo
Tabela	TabelaXML
Matriz	int[ ][ ]
Nome do Método	Função
setTabela	Define a TabelaXML, usa o método criaMatriz, para criar uma matriz compatível com TabelaXML, e povoa essa matriz que sera usada no método compara da classe Dados;
criaMatriz	Cria uma matriz de inteiros da mesma proporção da tabela TabelaXML;
getMatriz	Retorna a matriz de inteiros matriz;

Fonte: Do Autor.

Quadro 5 – Classe ConfControlador

Nome da Classe	**ConfControlador**
Resumo	
Classe responsável por configurar o microcontrolador (Arduino Mega2560) das saidas.	
Atributos	
Nome do Atributo	Tipo do Atributo
Serial	SerialRXTX
serialUsada	SerialRXTX[ ]
Métodos	
Nome do Método	Função
configPortArd	Define a porta serial que o microcontrolador irá usar e estabelece a comunicação entre eles;

setDelay	Define com base no dado enviado o tempo de acionamento de cada linha e envia essa mensagem ao microcontrolador;
setDelayEspecifico	Define com base no dado enviado o tempo de acionamento de determinada saída e envia essa mensagem ao microcontrolador;
setLinhas	Define o número de linhas e envia essa mensagem ao microcontrolador;
setSaidas	Define o número de saídas de cada linha e envia essa mensagem ao microcontrolador;
getSerialArd	Retorna as seriais usadas para configurar o arduino;
tempAcion	Função que calcula o tempo de acionamento, com base nos dados contidos na configuração da máquina;

Fonte: Do Autor.

Quadro 6 – Classe TesteSaídas

Nome da Classe	**TesteSaidas**
Resumo	
Classe responsável pela função de teste das saídas.	
Atributos	
Nome do Atributo	Tipo do Atributo
Serial	SerialRXTX
dadoRecb	Observable
numDeEspaco	int
flagOcupacao	boolean
Saída	int
Métodos	
Nome do Método	Função
Update	Responsável por receber o dado vindo da classe Protocolo através de um observador, esse método serve para indicar quando ocorre uma nova entrada de peso e com base no valor que a variável numDeEspaco e no de flagOcupacao decide se aciona ou não a saída desejada;
setNumDeEspaco	Responsável por definir o número de espaço entre cada período de acionamento (numDeEspaco);
setSaida	Responsável por setar a saída que será testada (saída);

Fonte: Do Autor.

Quadro 7 – Classe SalvaCarregaXML

Nome da Classe	**SalvaCarregaXML**
Resumo	
Classe responsável por salvar objetos do tipo DadosConfXML (configurações do sistema) e/ou do tipo TabelaXML (tabela de calibres) em formato XML, e por transformar formatos XML em objetos do tipo DadosConfXML (configurações do sistema) e/ou do tipo TabelaXML (tabela de calibres)	
Atributos	
Nome do Atributo	Tipo do Atributo
Métodos	
Nome do Método	Função
Salva	Salva um objeto em um formato XML, em um determinado local do Computador;
Abrir	Carrega de um determinado local do Computador um arquivo XML e transforma em um objeto;

Fonte: Do autor.

Quadro 8 – Classe DadosConfXML

Nome da Classe	**DadosConfXML**
Resumo	
Classe que responsável por gerar objetos com os dados de configuração do sistema.	
Atributos	
Nome do Atributo	Tipo do Atributo
numeroLinhas	int
numeroSaidas	int
Atraso	int[ ]
distanciaDosCopos	int[ ]
nomeSaidas	String[ ]
nomeEntrada	String[ ]
Tara	int[ ]

Desconsiderar	int[ ]
Correção	double[ ]
tabTempAcion	int[ ][ ][ ]

Métodos	
Nome do Método	Função
setNumeroLinhas	Responsável por setar a variável numeroLinhas, que define o número de linhas da máquina;
setNumeroSaidas	Responsável por setar a variável numeroSaidas, que define o número de saídas que cada linha da máquina;
setAtraso	Responsável por setar o vetor atraso, este que define o tempo de atraso usado na função de teste onde cada posição do vetor equivale ao atraso para cada linha;
setDistanciaDosCopos	Responsável por setar o vetor deisatanciaDosCopos, este que define o espaço de cada saída contado a partir da célula de carga;
setNomeSaidas	Define o caminho para as portas de saída, ou seja a porta que o controlador (es) irá (m) usar;
setNomeEntrada	Define o caminho para as portas de entrada, ou seja, as portas que irá comunicar com o modulo AD_104;
setTara	Responsável por setar o vetor tara, este que define a tara para a pesagem de cada linha;
setCorrecao	Responsável por setar o vetor correção, este que define a correção para cada linha;
setDesconsiderar	Responsável por setar o vetor desconsiderar, este que define o peso que deve ser desconsiderado pelo sistema;
setTabTempAcion	Responsável por setar o vetor tabTempAcion, este que define a velocidade de copos por minuto e o tempo de acionamento para aquela velocidade, é importante falar que esse vetor é tridimensional, devido a primeira coluna ser a velocidade, a segunda o tempo de acionamento e o vetor referente a profundidade representar cada um uma linha;
getNumeroLinhas	Responsável por retornar a variável numeroLinhas, que define o número de linhas da máquina;
getNumeroSaidas	Responsável por retornar a variável numeroSaidas, que define o número de saídas que cada linha da máquina;
getAtraso	Responsável por retornar o vetor atraso, este que define o tempo de atraso usado na função de teste;

getDistanciaDosCopos	Responsável por retornar o vetor deisatanciaDosCopos, este que define o espaço de cada saída contado a partir da célula de carga;
getNomeSaidas	Retorna o caminho para as portas de saída, ou seja a porta que o controlador(es) irá(m) usar;
getNomeEntrada	Retorna o caminho para as portas de entrada, ou seja, as portas que irá comunicar com o modulo AD_104;
getTara	Responsável por retornar o vetor tara, este que define a tara para a pesagem de cada linha;
getCorrecao	Responsável por retornar o vetor correção, este que define a correção para cada linha;
getDesconsiderar	Responsável por retornar o vetor desconsiderar, este que define o peso que deve ser desconsiderado pelo sistema;
getTabTempAcion	Responsável por retornar o vetor tabTempAcion, este que define a velocidade de copos por minuto e o tempo de acionamento para aquela velocidade de cada linha;

Fonte: Do Autor.

Quadro 9 – Classe TabelaXML

Nome da Classe	**TabelaXML**
Resumo	
Classe que responsável por gerar objetos com os dados do calibre, peso mínimo, peso máximo e saídas ativas/desativas, e o peso acumulado naquele calibre;	
Atributos	
Nome do Atributo	Tipo do Atributo
Calibre	int[ ]
pesoMinimo	int[ ]
pesoMaximo	int[ ]
Saída	String[ ][ ]
Peso	int[ ]
Métodos	
Nome do Método	Função
setCalibre	Responsável por definir o vetor calibre, que define o calibre dos frutos;
setPesoMinimo	Responsável por definir o vetor pesoMinimo, que define o peso

	mínimo para cada calibre;
setPesoMaximo	Responsável por definir o vetor pesoMaximo, que define o peso máximo para cada calibre;
setSaida	Responsável por definir a matriz saída, que define quais saídas de quais calibres estão ativas/desativas;
setPeso	Responsável por definir o vetor peso, que define o peso total de frutos que foi registrado para aquele calibre;
getCalibre	Responsável por retornar o vetor calibre, que define o calibre dos frutos;
getPesoMinimo	Responsável por retornar o vetor pesoMinimo, que define o peso mínimo para cada calibre;
getPesoMaximo	Responsável por retornar o vetor pesoMaximo, que define o peso máximo para cada calibre;
getSaida	Responsável por retornar a matriz saída, que define quais saídas de quais calibres estão ativas/desativas;
getPeso	Responsável por retornar o vetor peso, que define o peso total de frutos que foi registrado para aquele calibre;

Fonte: Do Autor.

Quadro 10 – Classe Protocolo

Nome da Classe	**Protocolo**
Resumo	
Classe responsável por tratar os dados vindos da classe SerialRXTX, cujo sua função é ter um argumento observável, ou seja, disparar mensagens para deve recebê-la.	
Atributos	
Nome do Atributo	Tipo do Atributo
leituraComando	String
Métodos	
Nome do Método	Função
interpretaComando	Responsável por tratar o dado vindo da classe SerialRXTX, transformá-lo em um inteiro e enviar para os observadores;
setLeituraComando	Seta a variável leituraComando, e chama o método interpretaComando;
setEnvio	Responsável por setar os dados para os observadores;

Fonte: Do autor.

Quadro 11 – Classe SerialRXTX

Nome da Classe	**SerialRXTX**
Resumo	
Classe responsável por criar o canal de comunicação com os periférios, ou seja, classe que cria o caminho tanto para o envio quando para o recebimento dos dados pela porta serial, essa classe implementa um Listener para “ouvir” sempre o está ocorrendo e disparar uma sequência de eventos, ela trabalha no formato de Threads possibilitando o paralelismo.	
Atributos	
Nome do Atributo	Tipo do Atributo
Protocolo	Protocolo
serialPort	SerialPort
Spe	SerialPortEvent
Input	BufferedReader
Output	OutputStream
serialPortName	String
Métodos	
Nome do Método	Função
iniciaSerial	Responsável por iniciar a comunicação serial com uma porta pre definida;
listaPortas	Responsável por listar as portas seriais disponíveis;
Serial	Recebe o dado vindo da porta serial;
sendData	Responsável por enviar um dado pela porta serial para um periférico;
Close	Fecha a comunicação serial com aquela porta setada pelo método iniciaSerial, ou seja, finaliza comunicação para o objeto criado a partir da classe SerialRXTX;
setSerialPortName	Define o caminho, através do nome da porta serial, para que possa abrir a comunicação ou fechar a comunicação, é usada pelo método iniciaSerial;

Fonte: Do Autor.