UNIVERSIDADE DE PASSO FUNDO

Luís Guilherme Cataneo

CONTADOR ELETRÔNICO DE OVOS ATRAVÉS DE IMAGENS

Passo Fundo

2017

Luís Guilherme Cataneo

CONTADOR ELETRÔNICO DE OVOS ATRAVÉS DE IMAGENS

Trabalho apresentado ao curso de Engenharia Elétrica, da Faculdade de Engenharia e Arquitetura, da Universidade de Passo Fundo, como requisito parcial para obtenção do grau de Engenheiro Eletricista, sob orientação do professor Dr. Carlos Alberto Ramirez Behaine.

Passo Fundo

2017

Luís Guilherme Cataneo

Contador Eletrônico de Ovos através de Imagens

Trabalho apresentado ao curso de Engenharia Elétrica, da Faculdade de Engenharia e Arquitetura, da Universidade de Passo Fundo, como requisito parcial para obtenção do grau de Engenheiro Eletricista, sob orientação do professor Dr. Carlos Alberto Ramirez Behaine.

Aprovado em ____ de _____ de_____.

BANCA EXAMINADORA

_____

Prof. Dr. Orientador Carlos Alberto Ramirez Behaine - UPF

_____

Prof. Dra. Blanca Rosa Maquera Sosa - UPF

_____

Prof. Dr. Fernando Passold - UPF

Este trabalho é dedicado a meus pais Juvir e Dirce Cataneo, a minha irmã Letícia Cataneo, a minha namorada Melânia Paula Pavoni, aos colegas Marcos Dal Moro e Rodrigo Donzelli, ao professor Carlos Alberto Ramirez Behaine e a todos que contribuíram para o sucesso deste trabalho.

AGRADECIMENTOS

Agradeço a Deus, a meus pais pelo incentivo e apoio financeiro, a minha namorada pela paciência e compreensão, ao colega Marcos Dal Moro por me inteirar no mundo da avicultura, ao colega Rodrigo Donzelli pelo chimarrão e café que nos mantiveram acordados até altas horas, ao professor Carlos Alberto Ramirez Behaine por todo o empenho e dedicação que teve como professor orientador.

“A persistência é o caminho do êxito.”

Charles Chaplin

RESUMO

A contagem precisa de ovos é de grande relevância no controle de qualidade de produção em núcleos de produção de ovos férteis. Na grande maioria das situações, isso faz com que seja percebida rapidamente alguma anomalia no lote, reduzindo assim custos futuros indesejados. A partir deste contexto, originou-se a ideia de desenvolver um dispositivo, que seja capaz de realizar a contagem dos ovos produzidos em um galpão, através do processamento digital de uma sequência de imagens e posteriormente apresentar em um *display* o valor total da produção. Para tal desafio, foi utilizado um microcomputador *Raspberry Pi* em conjunto com uma câmera, para realizar a captura das imagens, retirar as informações e representar o resultado no *display*. Tal dispositivo possui um preço acessível e uma capacidade de processamento compatível com a necessidade do projeto. Para a aquisição das imagens, foi estabelecido que a mesma só pode ser adquirida, posteriormente ao recebimento de um sinal de sincronismo vindo da esteira coletora de ovos. A detecção dos ovos é realizada através de um algoritmo baseado na estatística dos *pixels* brancos em uma imagem binária, obtida em um ambiente com luminosidade controlada, sendo imune a variações da forma de distribuição dos mesmos sobre a esteira.

Palavras-Chave: *Raspberry Pi*, Contagem de ovos, Imagem.

ABSTRACT

A precise count of eggs is of great importance in quality control in the production of fertile eggs production centers. In the vast majority of situations, this causes some anomaly to be quickly perceived in the batch, thus reducing unwanted future costs. From this context, was originated the idea of developing a device, which is capable of counting the eggs produced in a shed, through the digital processing of a sequence of images and subsequently present on a display the total value of the production. For this challenge, a Raspberry Pi microcomputer was used in conjunction with a camera, to capture the images, to extract the information and to represent the result on the display. This device has an affordable price and a processing capacity commensurate with the need of the project. For the acquisition of the images, it was established that it can only be acquired, later the reception of a signal of synchronism coming from the egg collecting belt. The detection of the eggs is performed through an algorithm based on the statistics of the white pixels in a binary image, obtained in an environment with controlled luminosity, being immune to variations in the way of distribution of the eggs on the belt.

Keywords: Raspberry Pi, Egg counting, Image.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ALU – Arithmetic Logic Unit – Unidade Lógica Aritmética.

BLE - Bluetooth Low Energy.

CCD – Charged Coupled Device – Dispositivo de Carga Acoplada.

CD-ROM – Compact Disc Read-Only Memory. – Disco Compacto – Memória Somente de Leitura.

CLP – Programmable Logic Controller – Controlador Lógico Programável.

CMOS – Complementary Metal Oxide Semiconducto – Semicondutor de Metal-óxido Complementar.

CPU – Central Processing Unit – Unidade Central de Processamento.

CSI – Câmera Serial Interface – Interface Serial Câmera

DSI – Display Serial Interface – Interface Serial Display

HDMI – High-Definition Multimedia Interface – Interface Multimídia de Alta Definição.

LED – Light Emitter Diode – Diodo Emissor de Luz.

Pixel – Picture Element – Elemento de Imagem.

RGB – Red, Green, Blue, – Vermelho, Verde, Azul.

RPM – Revolutions per minute – Rotações Por Minuto.

GND – GrouND – Terra.

RISC – Reduced Instruction Set Computer – Computador com um conjunto reduzido de instruções.

SUMÁRIO

1 INTRODUÇÃO

A produção de ovos férteis no Brasil vem crescendo muito nos últimos anos, devido ao constante crescimento da produção de frango de corte sendo destaque nos índices de produtividade, tendo indicadores de produção iguais ou frequentemente melhores aos encontrados em qualquer outro país do mundo. Este crescimento vem trazendo consigo a necessidade de implantação de diferentes tecnologias de controle nestes núcleos de produção, com a globalização dos mercados, a margem de lucro tem diminuído ao longo do tempo, o que tem conduzido o setor a uma busca constante da máxima eficiência no processo produtivo. Para tanto, a incorporação constante de novas tecnologias em todas as áreas, tem se tornado fundamental para a sobrevivência da atividade.

Ao analisar o sistema de produção de ovos percebe-se outro fator que merece atenção, que diz respeito à mão de obra para a execução dos trabalhos nas granjas. Percebe-se que há pouca oferta de mão de obra devido à grande demanda ocasionada pelas operações de coleta, manejo dos ovos, busca de melhores salários e tarefas menos exaustivas faz com que a mão de obra migre para outros setores da economia.

Estes fatores levam avicultores e empresas a busca constante de equipamentos que facilitem os trabalhos através de tecnologia e automação. Pelo fato de a atividade de contagem de ovos na sua grande maioria ser uma atividade manual e existir a necessidade de se ter conhecimento da quantidade de ovos, que está sendo produzida diariamente em cada galpão, surgiu a ideia de desenvolver um contador eletrônico de ovos.

1.1 OBJETIVOS

Desenvolver um equipamento de baixo custo, que realize a contagem de ovos através de imagens para ser potencialmente adaptado a uma esteira, e mostrar em um *display* a contagem total.

1.2 JUSTIFICATIVA

Saber a quantidade de ovos produzidos em cada galpão é de extrema importância, pois com esse dado são gerados outros índices, como o de ovos de cama que são ovos postos na cama do aviário, ovos com trincas que são ovos com pequenas trincas em sua casca e eficiência de produção. Com a implantação deste sistema em núcleos de produção de ovos é possível obter estes índices mesmo em núcleos que possuem esteira transportadora de ovos, afinal hoje

isso não é possível de ser realizado de forma manual, havendo assim um descontrole dos índices individuais de cada galpão.

1.3 ESTRUTURA DO TRABALHO

No capítulo 2 é apresentada a importância de ter um controle de produção em granjas de produção de ovos, as diferentes formas de realizar a contagem de ovos, conceitos fundamentais do processamento de imagens, conceitos básicos sobre sensores matriciais, etapas e formas para realizar o reconhecimento de objetos. O capítulo 3 apresenta às características mecânicas do sistema, a descrição detalhada dos componentes que compõem o sistema e a descrição detalhada do algoritmo. No capitulo 4 são apresentados os resultados obtidos neste desenvolvimento. No capitulo 5 são apresentadas as considerações deste trabalho.

2 REVISÃO DE LITERATURA

2.1 INTRODUÇÃO

A avicultura de postura chegou ao Brasil com a vinda dos primeiros japoneses. Nas décadas de 1940 e 1950 instalaram-se no estado de Santa Catarina unidades de abate da Sadia e "Perdigão" (atual BRF Food's). Em seguida, a produção avícola se expandiu para a região Sul e recentemente para o Centro-Oeste e Norte do país, relatando uma trajetória de inovação tecnológica (BELUSSO; HESPANHOL, 2010).

As inovações tecnológicas na avicultura de postura que ocorreram recentemente podem ser consideradas como sendo consequência de fatores de competitividade. Estas inovações foram radicais e graduais em formas de criação, manejo, nutrição e genética das aves. A aquisição de novas tecnologias com o objetivo de reduzir custos, melhorar a quantidade, tempo de processamento, mão de obra, logística interna e externa do núcleo o que ocasiona a atividade a ser mais competitiva (PIZZOLANTE et al., 2011).

Na década de 60 a 70, os avicultores exerceram papel fundamental na implantação de novas tecnologias, pois neste período os fabricantes e as empresas se ajustavam a suas necessidades tecnológicas. Hoje, ocorre uma inversão onde as empresas e fabricantes criam novas tecnologias e lançam no mercado seus produtos, que poderão ser adquiridas pelos produtores (PIZZOLANTE et al., 2011).

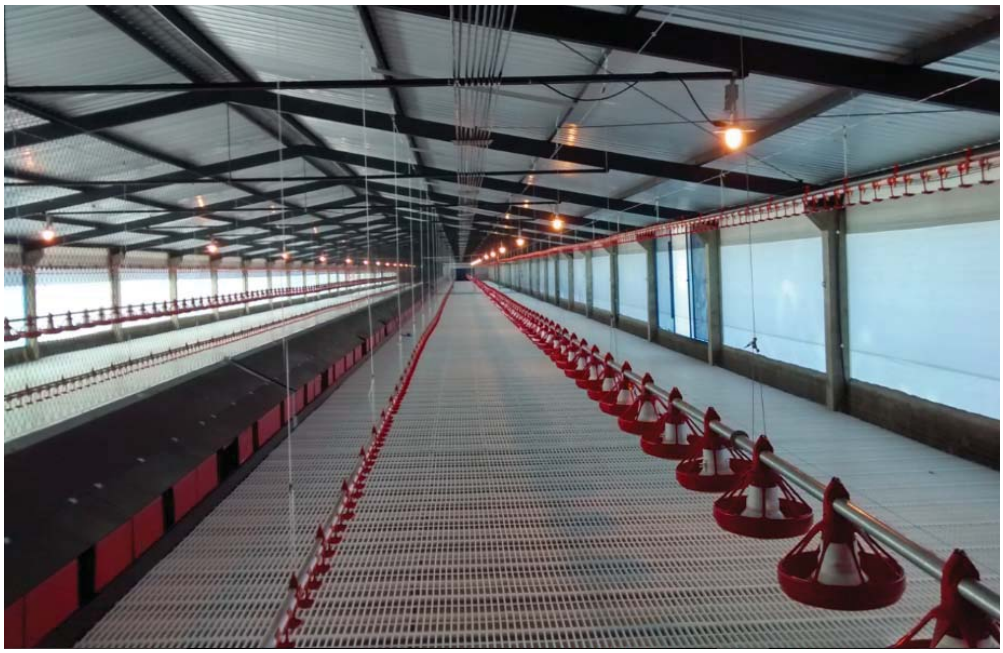
Na Figura 1 é apresentado um modelo de núcleo de produção de ovos férteis em construção. Este núcleo possui um sistema de coleta automática dos ovos que é realizada através de uma esteira coletora (interna ao galpão) e uma esteira transportadora que passa por todos os galpões, transportando os ovos produzidos dos mesmos.

Figura 1 – Granja de produção de ovos férteis em construção.

Fonte: GSI Agromarau.

Na Figura 2 é apresentada uma imagem do interior de um galpão de produção de ovos férteis. No interior do ninho (casinha no corredor central do galpão) é onde ocorre à postura, e no interior deste encontra-se a esteira coletora de ovos que faz o transporte dos ovos até a esteira transportadora.

Figura 2 – Vista interna galpão de produção de ovos sem aves.



Fonte: GSI Agromarau.

Na Figura 3 é exposta uma imagem do interior de um galpão de produção de ovos férteis em operação.

Figura 3 – Vista interna galpão de produção de ovos com aves.

Fonte: GSI Agromarau.

Na Figura 4 é mostrada como é realizada a transição dos ovos entre a esteira coletora e a esteira transportadora de ovos.

Figura 4 – Esteira coletora (branca), esteira transportadora (amarela).

Fonte: GSI Agromarau.

Na Figura 5 é mostrado um modelo de instalação da esteira transportadora de ovos em fase de montagem.

Figura 5 – Esteira transportadoura de ovos em construção.

Fonte: GSI Agromarau.

2.2 CONTADOR DE OVOS

O histórico da quantidade diária produzida, peso do ovo e índice de ovos trincados e ovos sujos têm grande influência na adoção de definições a respeito do manejo das aves e dos ovos (ROCHA et al., 2010). Mais relevante do que possuir registros é assegurar que estes sejam fiáveis. Para obter isso, as técnicas de coleta de informações necessitam ser arquitetadas, retratadas e rigorosamente seguidas (ROCHA et al., 2010).

Da mesma forma que nos processos industriais, que fazem uso de matérias-primas de origem animal ou vegetal, há um desconforto em detectar variação de padrões. Esses problemas se manifestam de diferentes maneiras, sendo na principal os diferentes tamanhos de ovos (BOARETTO et al., 2005).

Ainda que a postura prematura resulte em uma quantidade maior de ovos, a maturidade prematura resulta em grande quantidade de ovos pequenos (Manual de Segurança e Qualidade para Avilcultura de Postura, 2004). Técnicas como a de visão artificial são ferramentas com larga aplicação nos processos industriais, especialmente por aceitarem uma alta velocidade de processamento, excluindo totalmente ações humanas diretas. Além disso, por operar

ininterruptamente e ser de fácil implementação, em comparação com as práticas tradicionais que, geralmente, necessitam de mecanismos sofisticados para a alocação dos sensores (BOARETTO et al., 2005).

2.3 DIFERENTES FORMAS DE REALIZAR A CONTAGEM DE OVOS

2.3.1 Contagem Manual

A partir do momento em que o ovo é posto, alterações físicas e químicas são iniciadas e começam a modificar o seu frescor. Temperaturas elevadas contribuem para essas alterações, por isso, os ovos devem ser coletados com frequência e refrigerados rapidamente. O ideal é que a coleta não seja manual, mas sim automatizada com a utilização de esteiras (POSTURA, 2004).

Desta forma, a contagem manual Figura 6 também não é apropriada por submeter os ovos a possíveis contaminações. Além disso, pode provocar um pequeno aumento da temperatura do mesmo o que pode acabar acelerando as alterações químicas e físicas.

Figura 6 – Contagem manual de ovos.

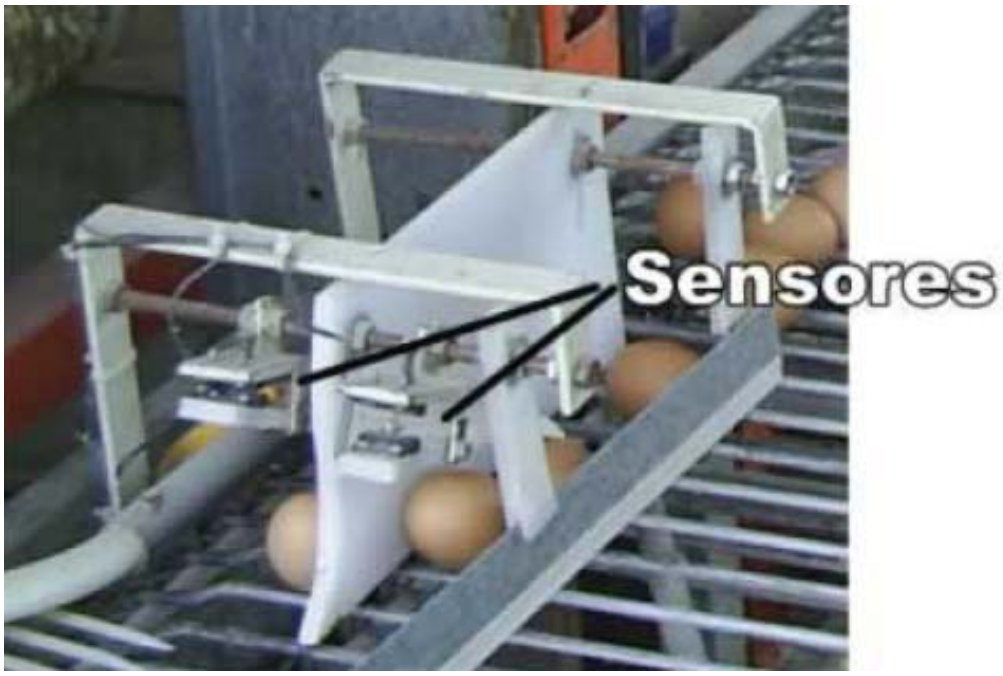
Fonte: GSI Agromarau.

2.3.2 Contagem Automática

2.3.2.1 Utilizando CLP e micros mecânicos

Este método de contagem de ovos consiste em um *Programmable Logic Controller* (CLP) programado em Ladder, um dispositivo de alinhamento dos ovos, e um software supervisório SCADA, através da ferramenta visual VB. O programa percebe a existência dos ovos mediante uma variação de estado lógico nas entradas do CLP, conforme Figura 7 representa a instalação dos sensores e os dispositivos para alinhamento dos ovos (BOARETTO et al., 2005).

Figura 7 – Contador de ovos utilizando CLP e micros mecânicos.



Fonte: Adaptado de (BOARETTO et al., 2005).

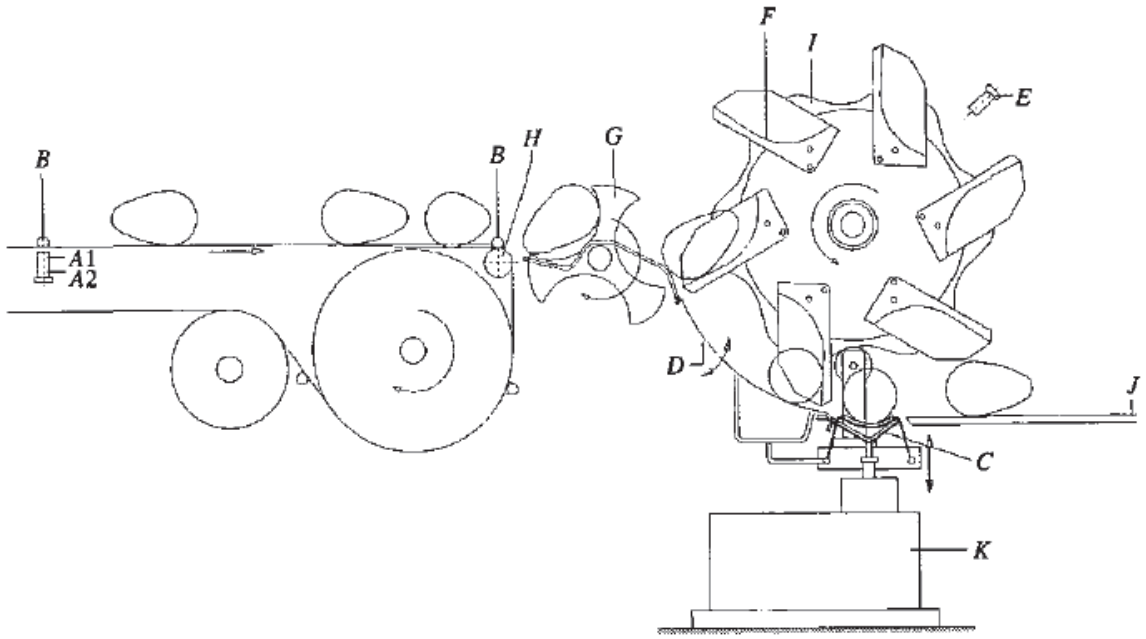
Essa forma de contagem exclui qualquer ação humana no interior do aviário diminuindo, assim, grandes prejuízos causados por contaminações. Porém, este sistema é sensível a variações de padrões dos ovos, sendo necessário ajuste de sensibilidade dos sensores ao longo do lote e não é sensível o suficiente para realizar a contagem de ovos frágeis sem danificá-los (BOARETTO et al., 2005).

2.3.2.2 Contagem e pesagem de ovos

Este sistema de contagem e pesagem de ovos é instalado no final da esteira coletora de ovos. Nesta forma de contagem é necessária a instalação de pequenas cristas Figura 8 (B) ao longo da esteira que facilitam o transporte dos ovos. Quando os ovos chegam ao cilindro Figura 8 (H) em que ocorre uma suave inclinação fazendo com que os ovos rolem para o separador de ovos Figura 8 (G). Os ovos são separados por meio de três linhas paralelas e o separador de ovos transporta os ovos para o carrossel, que consiste em seis bandejas paralelas, conforme Figura 8 (F) (LOKHORST; VOST, 1994).

Ainda a bandeja guia os ovos para o quadro Figura 8 (D). O quadro é movido para cima e para baixo por um eixo de comando Figura 8 (I). Quando o quadro esta para baixo os ovos encontram-se na balança Figura 8 (C). Após a pesagem o ovo é empurrado pela bandeja para a mesa de coleta Figura 8 (J). A chave magnética Figura 8 (E) fornece um sinal de interrupção para pesar o ovo. Quando todos os ovos são pesados e contados a correia deve retornar à sua posição de partida. A contagem e pesagem são combinadas e o sistema necessita de limpeza periódica de uma bandeja que está instalada embaixo do separador de ovos e do carrossel (LOKHORST; VOST, 1994).

Figura 8 – Sistema de contagem e pesagem de ovos automático.



Fonte: Adaptado de LOKHORST e VOST (1994).

2.3.2.3 Contagem de ovos individual em instalações com produção em gaiolas

A contagem de ovos em instalações com produção em gaiolas pode ser realizada individualmente. Utilizando um sistema de contagem que consiste em um eixo com dedos metálicos e este eixo ligado a um disco enumerado que indica o numero de ovos produzidos pela galinha na gaiola (BAUMSTARK, 1958).

O processo de contagem é realizado de forma que as gaiolas possuem uma inclinação, que permite que o ovo que é posto role até o mecanismo de contagem. Quando o ovo entra em contato com o sistema de contagem e provoca um giro de um quarto de volta no sistema, faz com que seja incrementada uma unidade na contagem. Este modelo possui também um sistema que não permite o movimento contrário ao movimento normal de modo a não ser possível a contagem no sentido contrário (BAUMSTARK, 1958).

Este sistema deixa os ovos sujeitos a uma fricção considerável que prejudica repetidamente o movimento de rotação livre das partes de modo que a quebra de um ovo é uma possibilidade. Além disso, o operador não tem facilidade em ler a quantidade de ovos que o dispositivo registrou (BAUMSTARK, 1958).

2.4 PROCESSAMENTO DIGITAL DE IMAGEM

A imagem é definida como uma função bidirecional *f(x,y)*, onde *x* e *y* são coordenadas do plano, e a magnitude de *f* em qualquer par de coordenadas *(x,y)* é definida como intensidade da imagem neste ponto. Quando *x, y* e os valores de intensidade de *f* são valores finitos e discretos chamamos de *imagem digital* (GONZALEZ; WOODS, 2010).

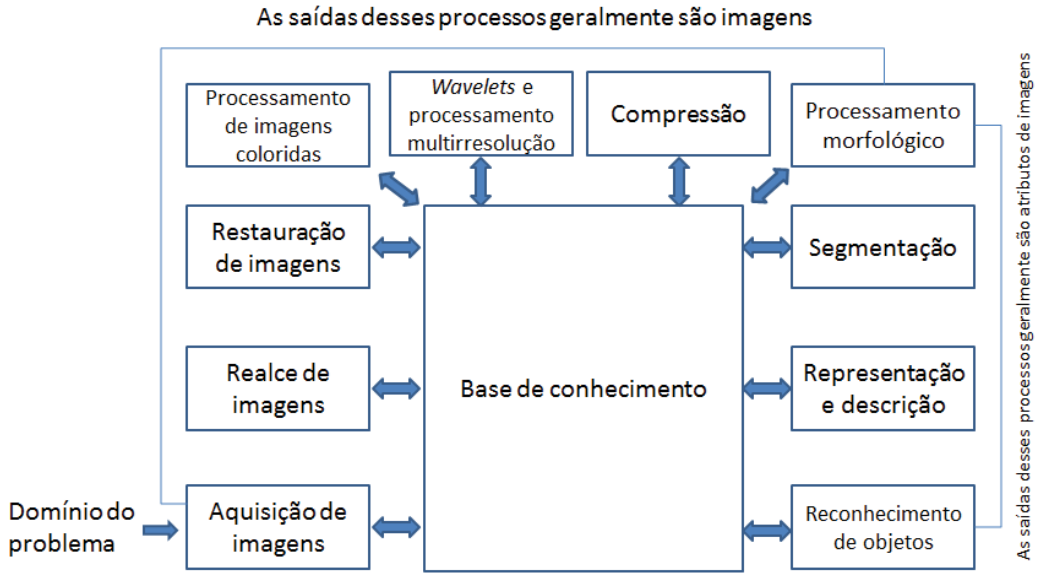
Percebe-se que uma imagem digital é definida por uma matriz com um número finito de elementos, cada um com determinada posição e valor específico. Esses elementos são intitulados como *elemento de imagem* ou, da abreviatura do inglês, *pixels* (**Pic**ture **El**ement), e são o menor elemento de uma imagem. Atribuindo-se um determinado valor de intensidade para cada um dos *pixels* de uma matriz é possível observar que uma imagem é formada a partir do posicionamento desses *pixels* (GONZALEZ; WOODS, 2010).

2.4.1 Principais passos em Processamento Digital de Imagens

O processamento digital de imagens pode ser dividido em dois grupos. Método este, cuja entrada e saída são imagens e método no qual as entradas podem ser imagens, mas as saídas são

atributos extraídos desta. Esta organização está representada no diagrama da Figura 9. Este diagrama tem como princípio nos dar uma ideia de todos os métodos que podem ser aplicados a uma imagem para diferentes finalidades, o que não quer dizer que todos os processos se aplicam a uma imagem (GONZALEZ; WOODS, 2010).

Figura 9 – Etapas fundamentais em processamento digital de imagens.



Fonte: Adaptado de Gonzales e Woods (2010, p. 15).

A *aquisição de imagens* é o primeiro processo da Figura 9 a ser realizado e consiste em adquirir a imagem, realizar o pré-processamento desta, exemplo realizar o redimensionamento. O *realce da imagem* é o processo de manusear uma imagem de modo que o resultado seja mais apropriado que o original para uma aplicação própria (GONZALEZ; WOODS, 2010).

A *restauração da imagem* é a técnica que tem por objetivo também fornecer uma melhoria visual da imagem, este método é baseado em modelos matemáticos ou probabilísticos de degradação das imagens. As *wavelets* representam a base para a representação de imagens em diversos níveis de resolução. A *compressão* é a técnica que tem como objetivo reduzir o armazenamento necessário para salvar uma imagem, ou a largura de banda exigida para transmiti-la (GONZALEZ; WOODS, 2010).

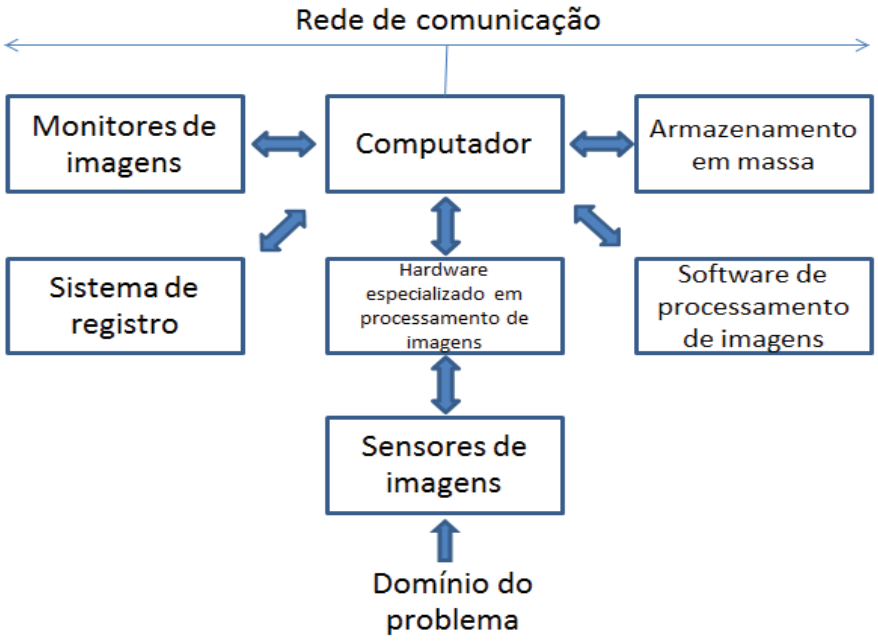
A *segmentação* consiste na divisão de uma imagem em suas partes constituintes. Em geral a segmentação autônoma é considerada uma das tarefas mais árduas do processamento digital de imagens. O *processamento morfológico* trata de componentes de imagem, importantes na representação e descrição da forma (GONZALEZ; WOODS, 2010).

A *representação e descrição* na grande maioria das vezes iniciam do resultado de um estágio de segmentação, que de modo geral são dados primários em forma de *pixels*, associados tanto a uma fronteira de uma região da imagem como todos os pontos dela. Nos dois casos é necessário transformar os dados em um formato adequado para o processamento computacional. A representação por fronteira é recomendada quando o interesse está focado nas características externas de forma, como vértices e formas de inflexão. A representação por região é recomendada quando o interesse se concentra nas propriedades internas do objeto, como textura ou a forma do esqueleto. O *reconhecimento* é o passo em que consiste identificar uma característica especial dentro de varias classes com base em *features* ou descritores (GONZALEZ; WOODS, 2010).

2.4.2 Componentes de um Sistema de Processamento de Imagem

Embora os sistemas de processamento de imagens em grande escala ainda serem vendidos para aplicações de maior porte envolvendo imagens, exemplo processamento de imagens de satélite, a orientação continua sendo a miniaturização e associação de pequenos computadores de uso geral com *hardware* para o processamento de imagens. A Figura 10 ilustra os componentes básicos que constituem um sistema típico de processamento digital de imagens (GONZALEZ; WOODS, 2010).

Figura 10 – Componentes de um sistema de processamento de imagens de uso geral.



Fonte: Adaptado de Gonzales e Woods (2010, p. 18).

A seguir, estão descritos a função de cada componente. O primeiro deles, o *sensoriamento* nesta etapa dois elementos são indispensáveis na aquisição de imagens digitais. O primeiro elemento é o dispositivo físico sensível à energia irradiante pelo objeto o qual desejamos captar a imagem. O segundo elemento é o digitalizador, é um dispositivo utilizado para converter o sinal da saída do dispositivo físico em formato digital (GONZALEZ; WOODS, 2010).

O *hardware especializado em processamento de imagens* na grande maioria consiste em um digitalizador, além de uma *Arithmetic Logic Unit* (ALU), que opera em paralelo realizando operações aritméticas em toda a imagem. De modo geral, essa unidade efetua funções que exigem um rápido processamento de dados, como por exemplo, a digitalização e o calculo da média em vídeos a 30 quadros/s (GONZALEZ; WOODS, 2010).

O *computador* em um sistema de processamento digital de imagem é um computador de uso geral que pode variar de um computador pessoal a um supercomputador dependendo da capacidade de processamento exigida pela aplicação. O *software* para o processamento de imagens compõe-se em módulos especializados em realizar tarefas exclusivas. Pacotes computacionais bons podem fornecer para o usuário a possibilidade de o usuário escrever códigos que utilizem estes módulos especializados. Pacotes mais sofisticados possibilitam a inserção destes módulos e de comandos gerais de *software* com base em uma linguagem computacional (GONZALEZ; WOODS, 2010).

A capacidade de *armazenamento em massa* é importante em aplicações em imagens. O armazenamento digital é decomposto em três categorias principais que são descritas como armazenamento de curto prazo, online e em arquivo. Armazenamento de curto prazo é utilizado durante o processamento, podendo armazenar os dados na memória do computador ou em placas de vídeo especializadas, chamadas de *frame buffers*. Armazenamento online para acesso relativamente rápido utiliza discos magnéticos ou mídias óticas para armazenar os dados. Armazenamento em arquivo para acessos pouco frequentes que armazena os dados em fitas magnéticas e discos óticos alojados em “jukeboxes” (GONZALEZ; WOODS, 2010).

Os *monitores de imagem* utilizados em grande escala atualmente são monitores de TV em cores (preferivelmente de tela plana). Estes monitores são controlados pelas placas de vídeo (gráficas ou de imagem), que são integrantes do sistema computacional. Em casos em que se faz necessário a visualização estereoscópica (3-D), se faz necessária à utilização de um “capacete” que é constituída de dois pequenos monitores de vídeo acoplados em um óculos que deve ser utilizado pelo usuário (GONZALEZ; WOODS, 2010).

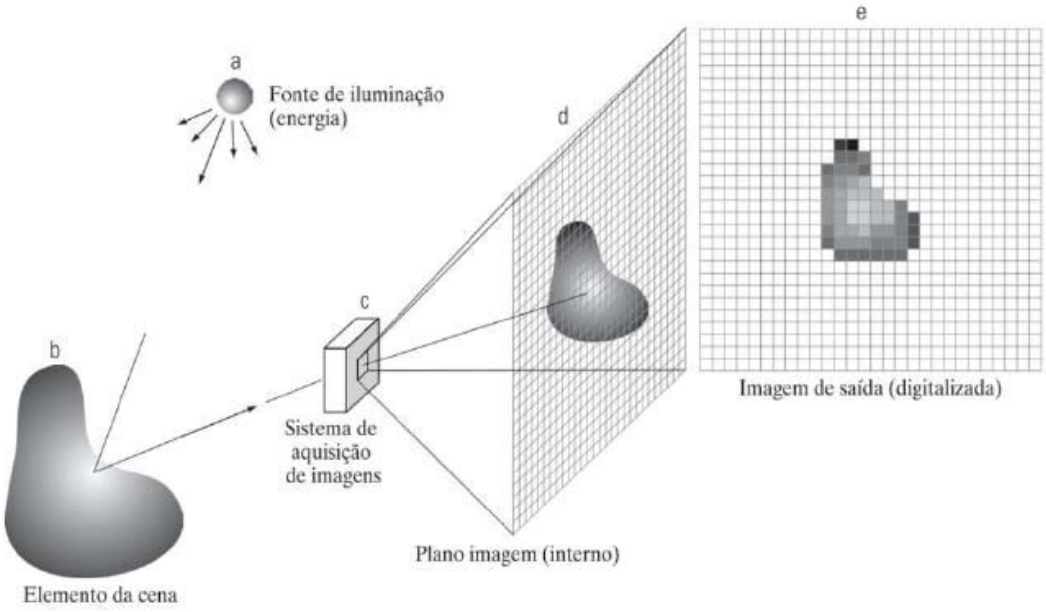
Os *sistemas de registro* para imagens incluem filmes fotográficos, impressoras térmica, jato de tinta, laser e mídias digitais, como os discos óticos e de *Compact Disc Read-Only Memory* (CD-ROM). Filmes podem proporcionam a mais elevada resolução possível, mas o papel é o meio favorito para o material escrito (GONZALEZ; WOODS, 2010). A *rede de comunicação* é praticamente um componente padrão de qualquer sistema computacional atual. Pelo grande volume de dados característicos de aplicações de processamento de imagem, a maior preocupação é a largura de banda essencial para a transmissão de imagens. Em redes dedicadas esta preocupação não representa um problema, mas em comunicações com sites remotos pela internet nem sempre são efetivas. Com a implantação da fibra ótica e de outras tecnologias de banda larga isso vem melhorando velozmente (GONZALEZ; WOODS, 2010).

2.4.3 Aquisição de Imagens utilizando Sensores Matriciais

Inúmeros dispositivos sensores eletromagnéticos e alguns ultrassônicos são repentinamente utilizados de forma matricial. Essa organização também é encontrada nas câmeras digitais. Sensores típicos para essas câmeras é uma matriz *Charged Coupled Device* (CCD), que pode ser fabricada com uma grande variedade de propriedades sensoras que podem estar dispostas em arranjos matriciais de 4000 X 4000 elementos ou mais. Sensores CCD têm ampla aplicação em câmeras digitais e outros dispositivos sensores de luz. A resposta de cada sensor é proporcional à integral da energia luminosa projetada sobre sua superfície (GONZALEZ; WOODS, 2010).

A forma crucial na qual os sensores matriciais são utilizados é indicada na Figura 11 (c). Nesta figura é mostrada a energia de uma fonte de iluminação sendo refletida de um elemento de cena. A primeira atribuição realizada pelo sistema de aquisição de imagens da Figura 11 (c) é coletar a energia de entrada e projetá-la num plano de imagem. Se a iluminação for luz, a entrada frontal do sistema de aquisição de imagens é constituída por uma lente ótica que projeta a cena vista sobre o plano focal da lente, como é indicado na Figura 11 (d). O conjunto de sensores, que corresponde com o plano focal, produz saídas integrais da luz recebida em cada sensor. Circuitos digitais e analógicos realizam uma varredura nas saídas de cada sensor e as convertem em um sinal analógico, que em sequência é digitalizado por outro componente do sistema de aquisição de imagem (GONZALEZ; WOODS, 2010).

Figura 11 – Processo de aquisição de uma imagem digital (a) Fonte de energia (“iluminação”). (b) Um elemento de uma cena. (c) Sistema de aquisição de imagem. (d) Projeção da cena no plano imagem. (e) Imagem digitalizada.



Fonte: Adaptado de Gonzales e Woods (2010, p. 32).

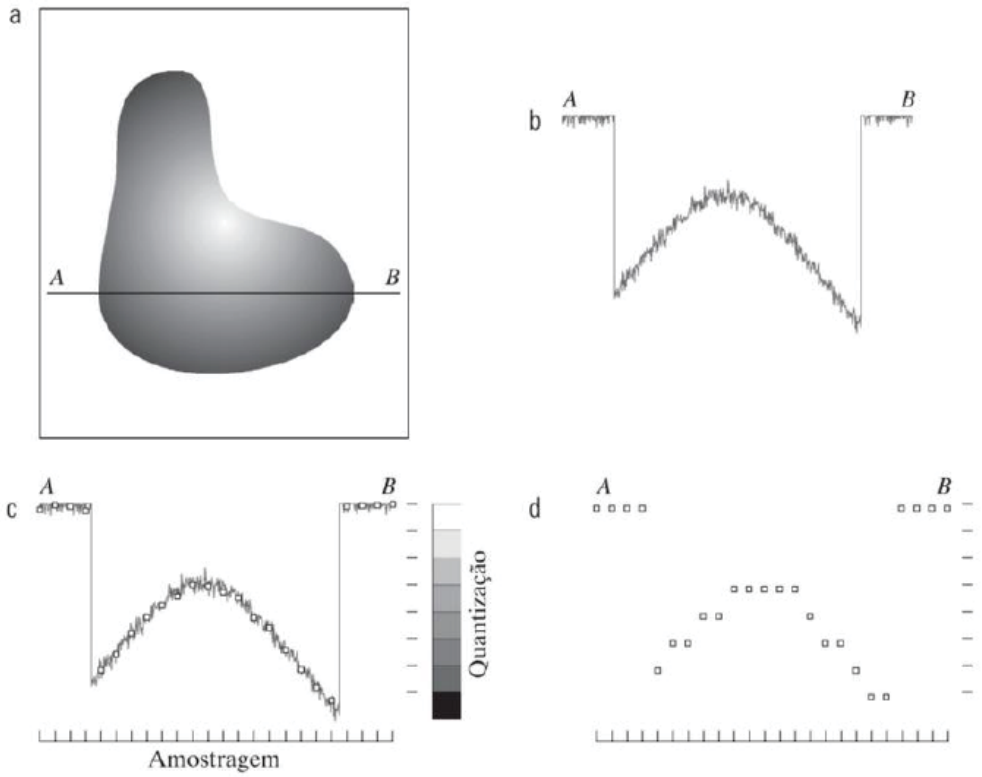
2.4.4 Amostragem e Quantização de Imagens

O conceito fundamental da amostragem e da quantização é representado na Figura 12. A Figura 12 (a) representa uma imagem continua *f* que desejamos converter em formato digital. Uma imagem pode ser continua tanto nas coordenas *x* e *y* quanto em sua amplitude. Para realizar a conversão de uma imagem para o formato digital devemos fazer a amostragem da função em ambas às coordenadas e na amplitude. *Amostragem* é chamada a digitalização dos valores das coordenada. *Quantização* é chamada a digitalização dos valores de amplitude (GONZALEZ; WOODS, 2010).

A função unidirecional da Figura 12 (b) é um gráfico que demonstra os valores de amplitude (níveis de intensidade) da imagem contínua entre a reta *AB* na Figura 12 (a). Ao lado direito da Figura 12 (c) é indicado à escala de intensidade dividida em oito intervalos discretos variando do preto ao branco. Na vertical da Figura 12 (c) são indicados os valores específicos atribuídos a cada um dos oito níveis de intensidade. Essa atribuição é realizada dependendo da proximidade vertical de uma amostra a uma marca indicadora. O resultado da amostragem e da quantização é indicado na Figura 12 (d). Está subentendido na Figura 12 (d) que, além do número discreto de níveis utilizados, a precisão atingida na quantização depende muito do conteúdo do ruído do sinal amostrado (GONZALEZ; WOODS, 2010).

Quando são utilizados sensores por varredura em linha para realizar a aquisição da imagem, quem define o as limitações da amostragem na direção da imagem é o número se sensores da linha. É possível controlar o movimento mecânico na outra direção com mais precisão, mas faz pouca lógica tentar atingir uma densidade de amostragem em uma direção que ultrapasse os limites de amostragem definidos pelo número de sensores da outra. Com a quantização das saídas dos sensores se completa o processo de formação de uma imagem digital (GONZALEZ; WOODS, 2010).

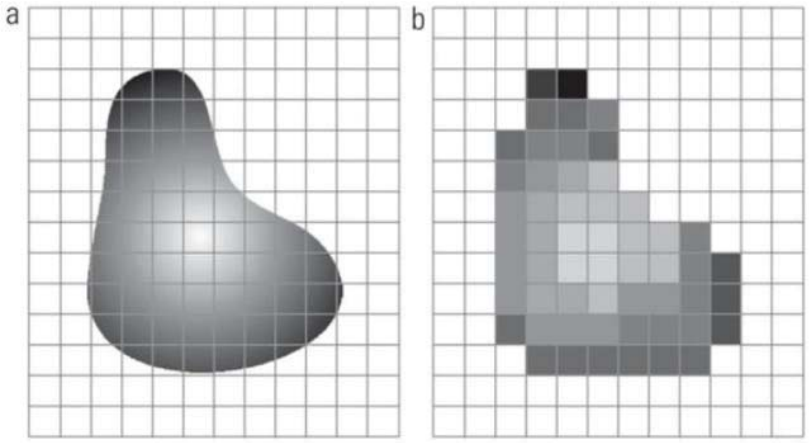
Figura 12 – (a) Imagem contínua. (b) Linha de varedura A e B na imagem contínua. (c) Amotragem e quantização. (d) Linha de varredura digital.



Fonte: Adaptado de Gonzales e Woods (2010, p. 34).

Quando é utilizada uma matriz de sensores para se realizar a aquisição de imagens, não existe movimento e os limites das amostragens em ambas as direções é determinado pelo número de sensores da matriz sensora. A quantização das saídas do sensor é realizada da mesma forma que é realizada quando utilizamos sensores de varredura em linha. A Figura 13 representa este conceito. A Figura 13 (b) ilustra a imagem após a amostragem e a quantização. A qualidade da imagem é determinada, na maior parte, pelo número de amostras e de níveis discretos de intensidade utilizados na amostragem e na quantização (GONZALEZ; WOODS, 2010).

Figura 13 – (a) Imagem contínua projetada em uma matriz de sensores. (b) Resultado da amostragem e quantização da imagem.



Fonte: Adaptado de Gonzales e Woods (2010, p. 35).

2.4.5 Passos para Realizar a Contagem de Objetos

Após a aquisição da imagem é necessário realizar o pré-processamento desta imagem a fim de extrair suas características. No pré-processamento é realizada a conversão da imagem colorida para uma imagem em escala de cinza, limiarização e a detecção de borda com o intuito de obter o contorno do objeto (MANASA et al., 2015).

Para realizar a c*onversão da imagem colorida para uma imagem em escala de cinza* o primeiro passo é obter as intensidades das corres primitivas vermelho, verde e azul (escala RGB). Após obter os valores de intensidade de cada cor basta adicionar as seguintes quantidades de cada cor 30% vermelha mais 59% verde mais 11% da azul. Estas porcentagens estão relacionadas com a sensibilidade do olho humano. Desta forma chegamos à equação (1) que deve ser realizada para cada *pixel* da imagem. Para transformar a imagem em tons de cinza, coloca-se o valor de $f_C(x,y)$ para as componentes RGB (CAVINATO, 2009).

$$f_C(x,y) = 0{,}299 f_R(x,y) + 0{,}587 f_G(x,y) + 0{,}114 f_B(x,y) \quad (1)$$

Onde:

$f_C(x,y)$ é a função bidimensional de nível de cinza em determinado ponto.

$f_R(x,y)$ é a função bidimensional de nível de vermelho em determinado ponto.

$f_G(x,y)$ é a função bidimensional de nível de verde em determinado ponto.

$f_B(x,y)$ é a função bidimensional de nível de azul em determinado ponto.

A *Limiarização* é utilizada para extração de objetos do fundo de uma imagem. Supondo uma imagem *f(x,y)* composta de um fundo escuro e objetos claros, que tenham a intensidade de cada *pixel* agrupadas em dois grupos majoritários (modos), desta forma podemos simplesmente selecionar um limiar T, que separará estes modos. Logo, qualquer ponto (x,y) da imagem em que *f(x,y)>T* é chamada de ponto do objeto caso contrario o ponto é chamado de ponto de fundo. De forma matemática esse processo é dado por:

$$g(x,y)=\begin{cases}1 & \text{se } f(x,y) > T \\ 0 & \text{se } f(x,y) \leq T\end{cases} \tag{2}$$

Onde:

g(x,y) é a função bidimensional da imagem limiarizada.

Embora seja seguida a convenção de usar os valores de 1 para a intensidade do objeto e 0 para a intensidade do fundo, esses valores podem ser alterados para quaisquer valores distintos (GONZALEZ; WOODS, 2010).

Detecção de borda é utilizada para detectar os *pixels* da borda dos objetos da imagem. Para detectar a borda dos objetos contidos em uma imagem se faz uso da primeira e segunda derivada de *f(x,y)*, existem diferentes aproximações para a primeira derivada, mas é necessário que a aproximação escolhida cumpra os seguintes pré-requisitos (1) seja igual a zero nas áreas de intensidade constante; (2) seja diferente de zero no inicio de uma rampa ou degrau; e (3) seja diferente de zero por toda extensão de uma rampa de intensidade. A primeira derivada pode ser obtida a partir da expansão da função *f(x + Δx)* em uma série de Taylor sobre x, onde $\Delta x = 1$ e mantendo-se apenas os termos lineares obtemos a equação (3) (GONZALEZ; WOODS, 2010).

$$\frac{\delta f}{\partial x} = f'(x) = f(x+1) - f(x) \tag{3}$$

Aplicando a derivada parcial na equação (3) obtemos a expressão para a segunda derivada indicada na equação (7), como *f* é uma função somente de *x* então *∂f/∂x =df/dx*.

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial f'(x)}{\partial x} = f'(x+1) \; - \; f'(x) \tag{4}$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+2) - f(x+1) - f(x+1) + f(x) \tag{5}$$

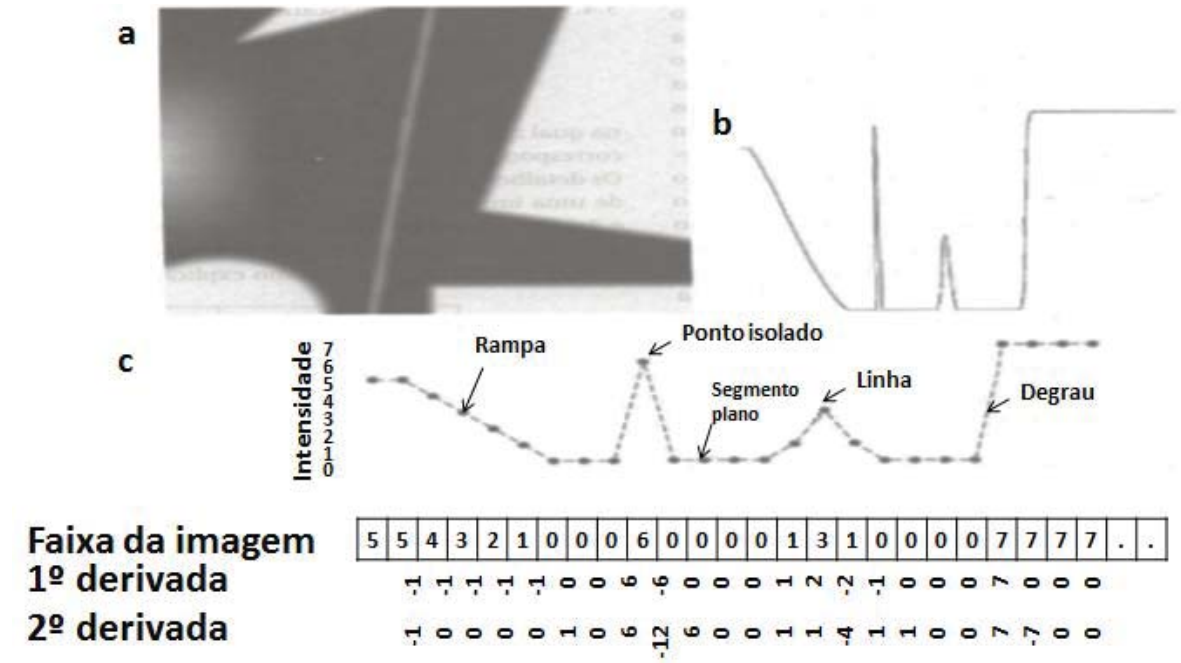
$$\frac{\partial^2 f}{\partial x^2} = f(x+2) - 2f(x+1) + f(x) \tag{6}$$

$$\frac{\partial^2 f}{\partial x^2} = f^n(x) = f(x+1) + f(x-1) - 2f(x) \tag{7}$$

O resultado da aplicação da derivada de primeira ordem é uma borda mais espessa e a aplicação da derivada de segunda ordem tem como resultado bordas muito mais finas.

Na Figura 14(a) e apresentada uma imagem formada por vários objetos sólidos, uma linha e um ponto de ruído. Na Figura 14(b) é representado um perfil de intensidade horizontal da imagem, próximo ao centro cruzando pelo ponto de ruído. As variações de intensidade entre os objetos sólidos e o fundo no comprimento da linha de digitalização representam dois tipos de bordas: *bordas em rampa* (à esquerda) e *bordas em degrau* (à direita) (GONZALEZ; WOODS, 2010).

Figura 14 – (a) imagem (b) perfil de intensidade horizontal (c) perfil simplificado.



Fonte: Adaptado de Gonzales e Woods (2010, p. 457).

2.4.5.1 Métodos para reconhecimento de objetos

- *Casamento (template matching)* é uma técnica de reconhecimento de objetos que representam cada grupo utilizando um vetor de características próprio. Atribui-se um padrão desconhecido ao grupo mais próximo em forma de uma métrica predeterminada, existem vários critérios para realizar este casamento, como o classificador da mínima distância e os classificadores estatísticos ótimos (GONZALEZ; WOODS, 2010).
- *O classificador de distância mínima* que utiliza a distância (euclidiana) entre o padrão desconhecido e cada um dos eventos protótipos para tomar uma decisão (GONZALEZ; WOODS, 2010).
- *Classificadores estatísticos ótimos* é uma técnica que utiliza de modelos probabilísticos para o reconhecimento de padrões que tem como objetivo obter o menor erro de classificação. Estes modelos probabilísticos geralmente utilizam classificadores *bayesianos* baseados no pressuposto de uma expressão analítica para as diversas funções densidade, seguidas da estimativa dos parâmetros das expressões a partir de amostras de cada grupo (GONZALEZ; WOODS, 2010).
- *Que utilizam transformações* são modelos que são baseados em transformadas como, por exemplo, a transformada de *hough,* estes modelos operam em diferentes espaços vetoriais.
- *Híbridos* são modelos que fazem uma junção dos modelos acima apresentados. Dando-lhes a os métodos escolhidos diferentes valores de importância para cada um dos métodos.

3 PROJETO DESENVOLVIDO

Neste capítulo será apresentada uma descrição detalhada dos componentes que compõem o sistema, assim como a maneira como estes se relacionam entre si para o funcionamento adequado do sistema e a descrição de funcionamento do algoritmo de contagem dos ovos.

3.1 ESTRUTURA MECÂNICA

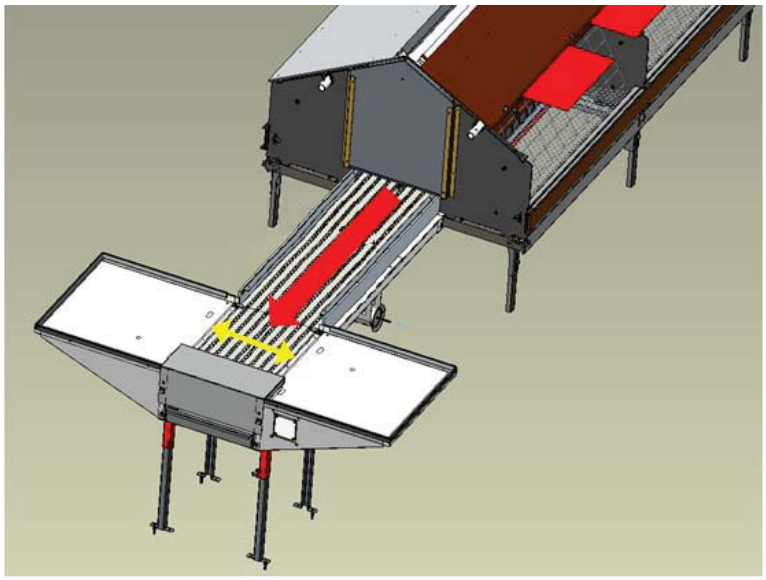
Neste item serão descritas as características do sistema onde este contador eletrônico de ovos possivelmente poderá ser instalado e as características do sistema desenvolvido para simular uma aplicação real.

3.1.1 Características da esteira

As esteiras são utilizadas para transportar objetos, e em um caso especial são utilizadas para transportar ovos, estas esteiras consistem em uma fita plástica perfurada com largura de aproximadamente 40 cm conforme indicado pela flecha amarela na Figura 15, que percorre todo o comprimento do ninho. Esta é posta em movimento a partir de um motor que é controlado por um inversor de frequência permitindo assim uma velocidade variável da esteira, que pode ter uma velocidade mínima de 0 m/mim e máxima de aproximadamente 8 m/min com sentido indicado pela flecha vermelha na Figura 15.

Para a aplicação deste sistema em uma esteira se faz necessário a utilização de uma esteira com cor diferente de branca, pois como os ovos também são brancos não haveria contraste, tornando-os imperceptíveis. Neste projeto, optamos por uma esteira de cor preta com isso obteremos um contraste elevado.

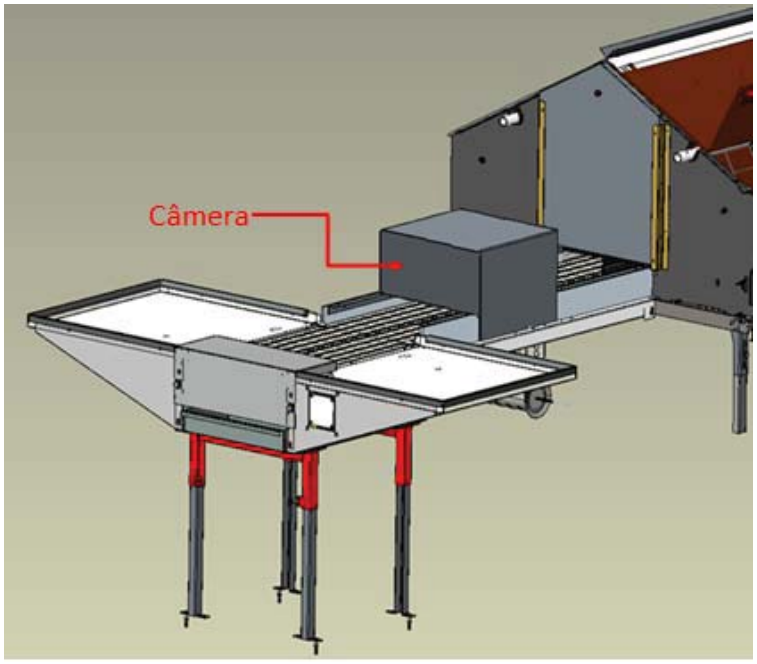
Figura 15 – Imagem esteira e indicação de direção de movimento.



Fonte: Própio autor.

Em uma aplicação real o sistema seria montado no final da esteira coletora de ovos. Acima desta, seria montada uma estrutura fechada onde internamente a esta estrutura, a câmera poderia ser fixada. O sistema ficará capturando imagens da esteira e realizando a contagem. Esta captura obedece a um sincronismo gerado por um sensor instalado na região de movimento da esteira. Na Figura 16 é apresentada uma imagem de como poderia ser a instalação real da câmera em uma esteira coletora de ovos.

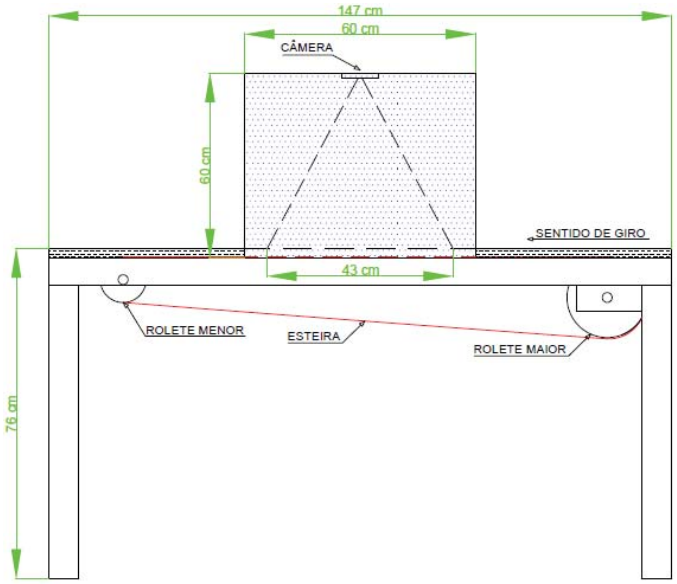
Figura 16 – Ilustração do sistema de contagem de ovos automático.



Fonte: Própio autor.

Na Figura 17 é apresentado o esquemático do protótipo deste sistema.

Figura 17 – Esquemático do sistema.

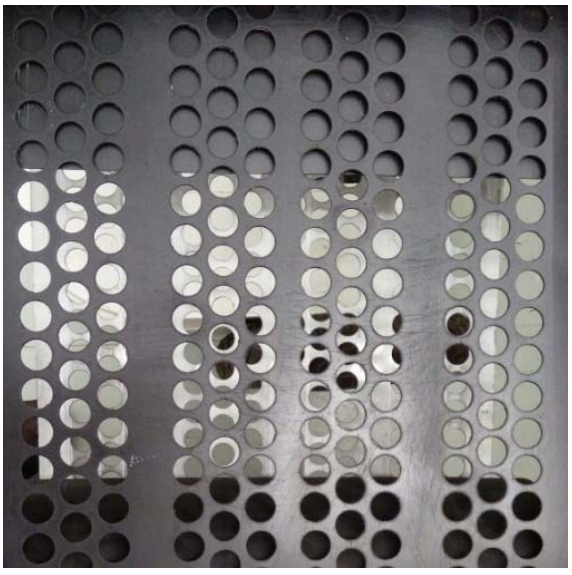


Fonte: Própio autor.

Para a realização de testes e levantamento de dados, foi desenvolvido um protótipo que simula uma condição possível de aplicação do projeto desenvolvido. Este se baseia em uma estrutura metálica e um compartimento de madeira.

A estrutura metálica foi desenvolvida para acomodar uma pequena esteira plástica perfurada de largura igual a 40 cm de cor preta conforme indicado na Figura 18 e a fixação do caixote utilizado na captura das imagens.

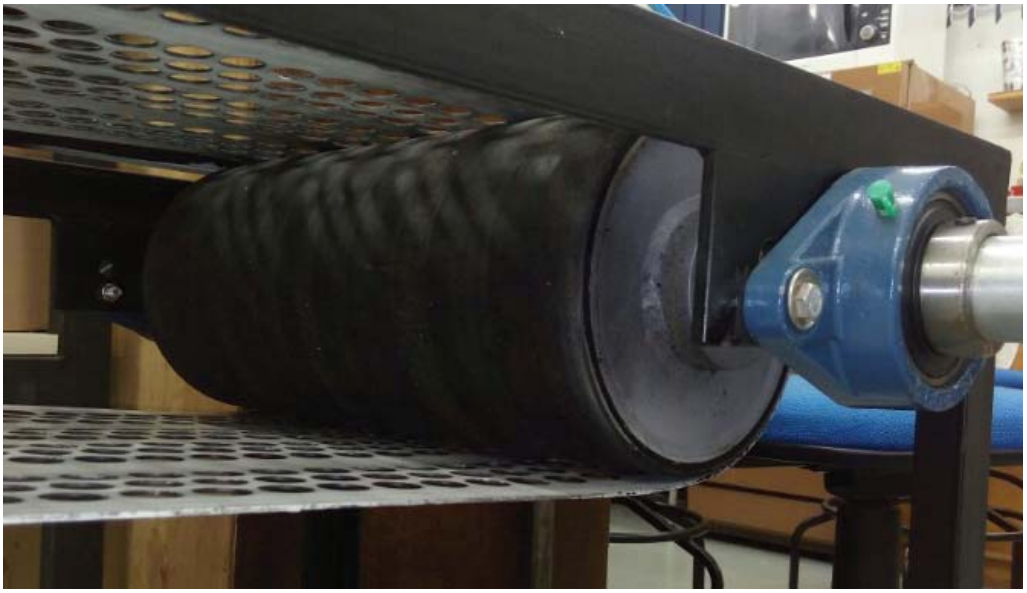
Figura 18 – Imagem esteira.



Fonte: Próprio autor.

A esteira roda sobre dois roletes sendo o de diâmetro maior utilizado pra proporcionar torque conforme indicado na Figura 19.

Figura 19 – Rolete maior.



Fonte: Próprio autor.

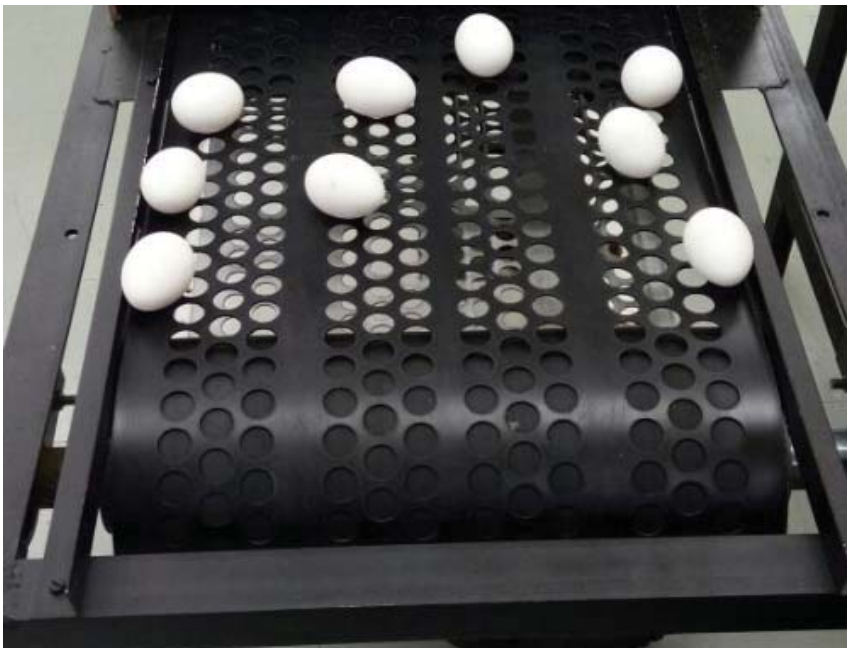
O de diâmetro menor juntamente com um sistema de molas é utilizado para tencionar e alinhar a esteira conforme representado na Figura 20.

Figura 20 – Rolete menor e tencionador.

Fonte: Próprio autor.

Nas laterais da esteira foram adicionadas duas cantoneiras conforme Figura 21, utilizada para alinhar os ovos sobre a esteira garantindo que os ovos postos sobre esta continuem por todo o processo de contagem.

Figura 21 – Cantoneiras de contensão.



Fonte: Próprio autor.

Ao final da esteira foi montada uma espécie de barreira metálica que foi revestida por uma espuma evitando que os ovos sejam arremessados para fora do sistema ou quebrados, o mecanismo montado está indicado na Figura 22.

Figura 22 – Barreira metálica.

Fonte: Próprio autor.

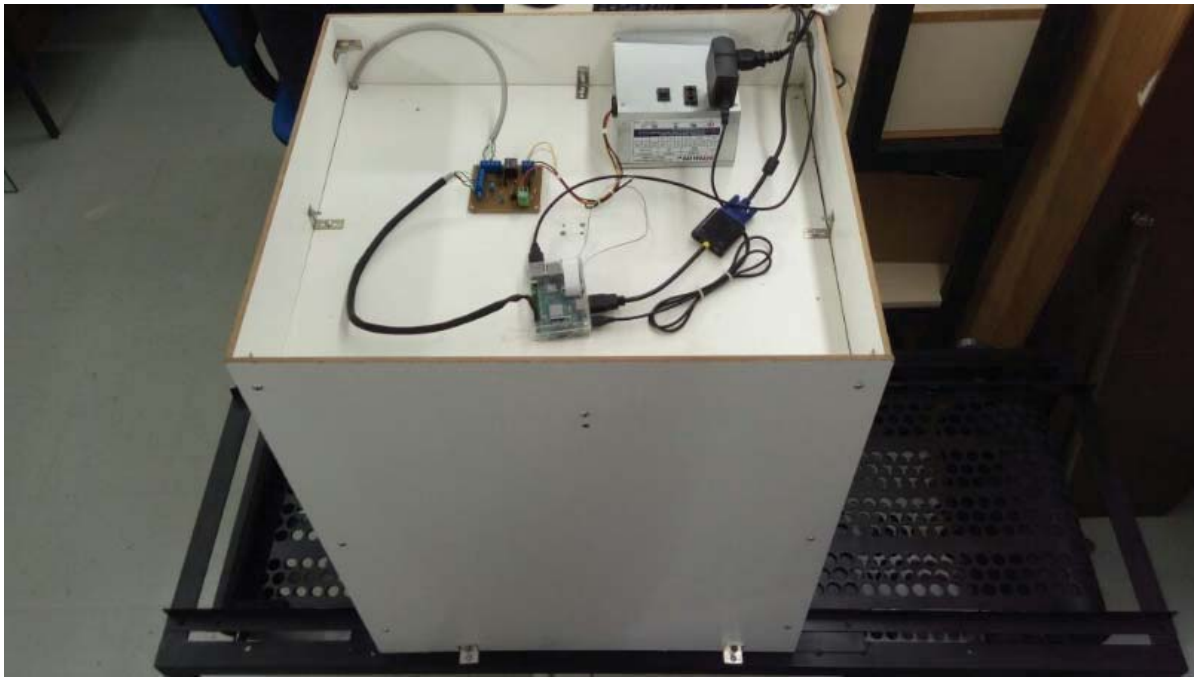
Próximo ao final da esteira foi construído dois pequenos suportes para realizar a fixação das partes do sensor de sincronismo, a forma de fixação está indicada na Figura 23.

Figura 23 – Suporte fixação sensor de sincronismo.

Fonte: Próprio autor.

Para acomodar a câmera, o sistema de iluminação, placa de circuito impresso, fonte de alimentação e o *hardware* de processamento, foi desenvolvido compartimento de madeira retangular com dimensões 0,6 X 0,53 X 0,6 metros que foi fixada sobre a estrutura metálica conforme mostrado na Figura 24.

Figura 24 – Compartimento de madeira.

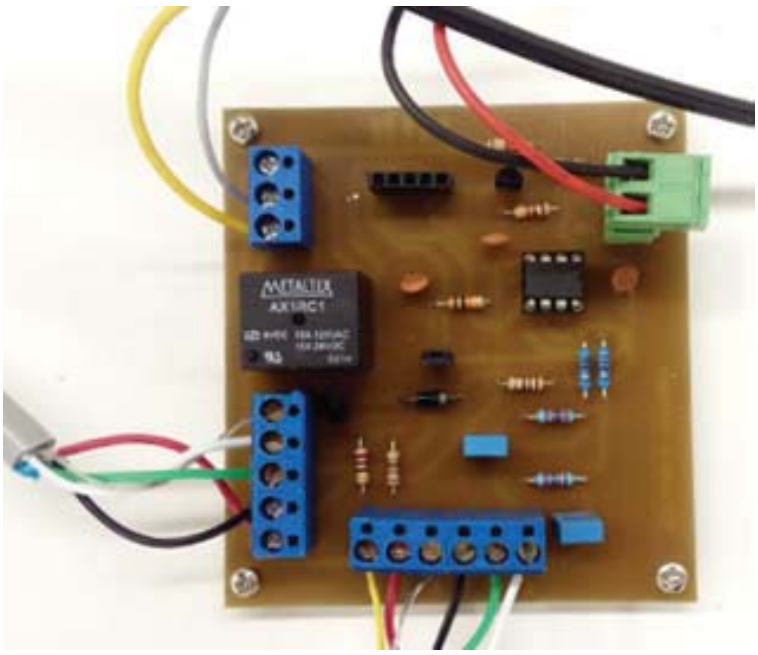


Fonte: Próprio autor.

3.2 SISTEMA ELETRÔNICO DE SINCRONISMO E ILUMINAÇÃO

Para o projeto em questão se fez necessário o desenvolvimento de uma placa de circuito impresso representada na Figura 25, responsável por ativar a iluminação da caixa de captura de imagens e gerar o sinal de sincronismo.

Figura 25 – Placa sinal de sincronismo e iluminação.

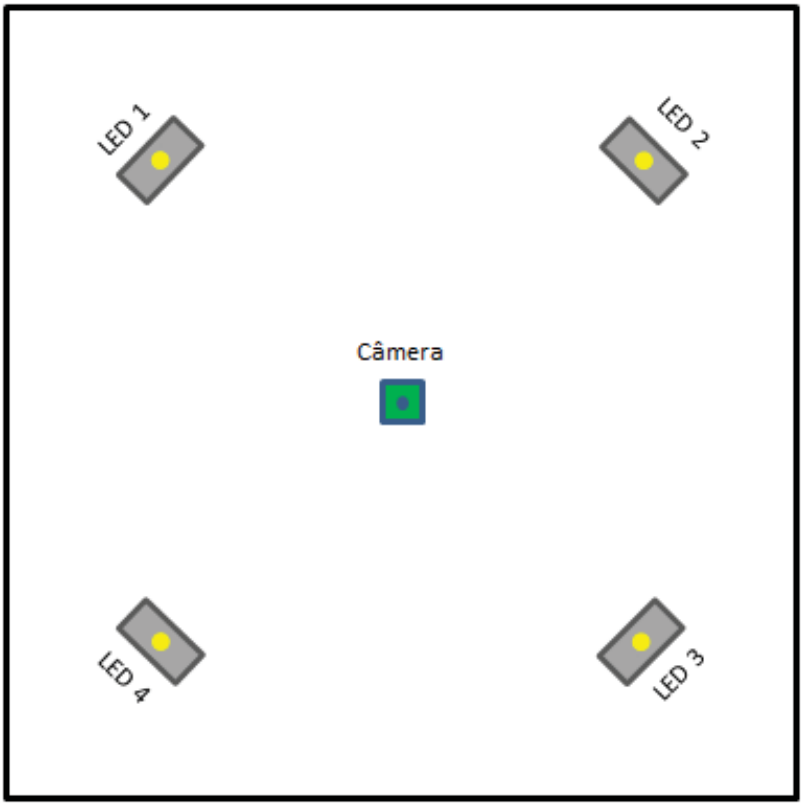


Fonte: Próprio autor.

3.2.1 Sistema Eletrônico de Iluminação

Para suprir a necessidade de uma iluminação constante no interior da câmera de captura, adotou-se uma configuração de iluminação que consiste em quatro *Light Emitter Diode* (LEDs), que estão instalados internamente ao compartimento de madeira na parte superior deste, sua disposição de instalação está representada na Figura 26, esta foi estabelecida com o objetivo de não ter pontos escuros ou muito claros no interior da câmera de captura. Facilitando o processamento da imagem, não sendo necessária a utilização de filtros no processamento da imagem capturada.

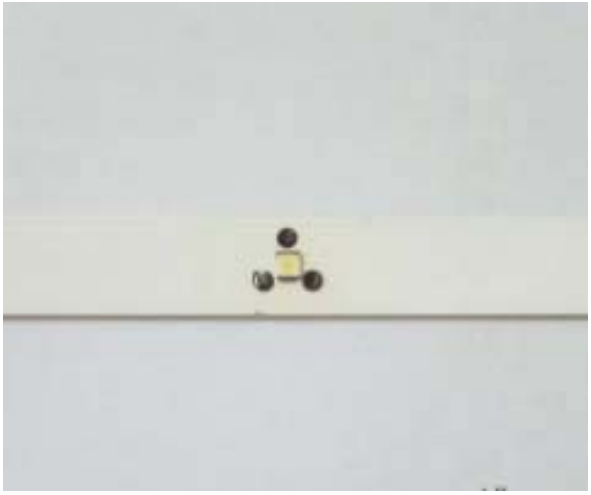
Figura 26 – Montagem mecânica LEDs.



Fonte: Próprio autor.

Para tal função foi optado por utilizar LED's utilizados nos *backlight* de TVs LEDs, este é indicado na Figura 27.

Figura 27 – Um led sistema de iluminação.



Fonte: Própio autor.

Na Tabela 1 são apresentadas as especificações técnicas dos LEDs utilizados.

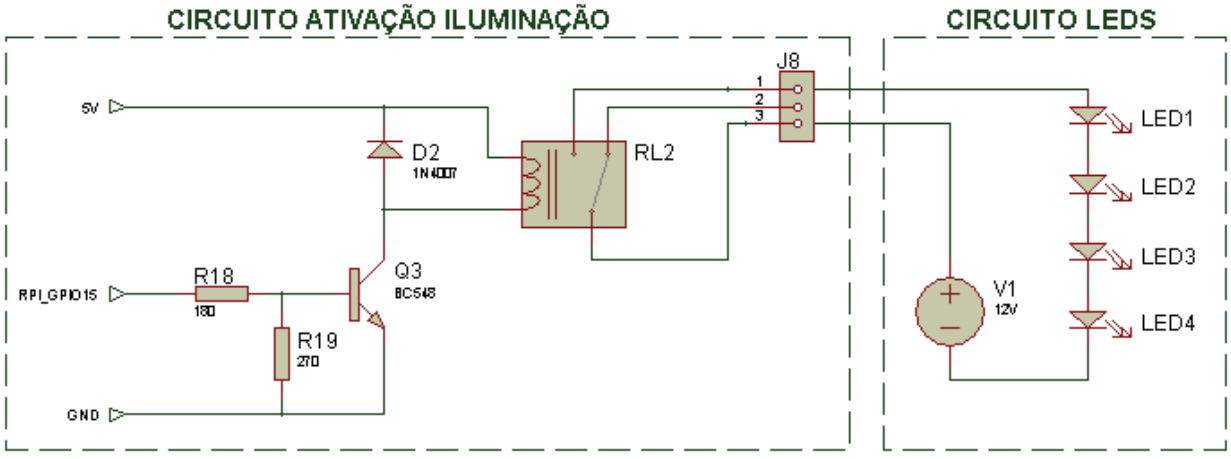
Tabela 1 – Especificações técnicas LEDs.

Fabricante	*Honglitronic*
Cor	Branco
Temperatura de cor	6000K
$v_{Fmáx}$ (tensão direta máx.)	3,6 V
v_{Fmin} (tensão direta min.)	2,8 V
I_F (corrente direta)	300 mA
I_R (corrente reversa)	10 µA
P_D (potência dissipada)	1000 mW
T_{OPR} (temperatura de operação)	-40 ~ +80 ºC

Fonte: Própio autor.

O circuito de ativação da iluminação está representado na Figura 28, este circuito é comandado pelo *Raspberry Pi* através da GPIO 15, que está configurada como saída, quando esta é colocada em nível lógico alto (3,3V) o transistor (Q3) é chaveado ativando o relé (RL2), este quando ativado conecta o +12 V proveniente da fonte de alimentação externa no ânodo do LED1 que está conectado em série com os LEDs 2,3 e 4, sendo o catodo do LED4 conectado ao 0 V da fonte externa fechando a malha, desta forma cada *LED* está submetido a 3 V.

Figura 28 – Diagrama elétrico circuito iluminação.



Fonte: Próprio autor.

3.2.2 Sistema Eletrônico de Sincronismo

Como o sistema tem como base de funcionamento a análise de uma sequência de imagens adquiridas, de uma esteira com velocidade variável, para garantirmos que as imagens não sejam sobrepostas, necessitamos a instalação de um sensor para realizar o sincronismo de aquisição de imagens.

Para o sensoriamento da esteira foi utilizado um sensor óptico modelo PHCT204 conforme indicado na Figura 29.

Figura 29 – Sensor optico PHCT204.

Fonte: Própio autor.

O sensor utilizado consiste em LED emissor infravermelho alinhado frente a frente a um foto transistor, este componente possui as características técnicas descritas na Tabela 2.

Tabela 2 – Especificações técnicas sensor PHCT204.

Fabricante	*Photonic*
EMISSOR	
v_F (tensão direta)	I_F = 30mA @ 1,5 V
I_F (corrente direta)	V_R = 4V @ 10 µA
DETECTOR	
I_{CEO} (fuga coletor-emissor)	(V_{CE} = 10V, I_F = 0) @ 100 nA
t_r (tempo de subida)	(I_C = 2mA, V_{CE} = 5V, R_L = 100 Ω) @ 5 µs
t_f (tempo de descida)	(I_C = 2mA, V_{CE} = 5V, R_L = 100 Ω) @ 5 µs
Dimensões	24,5 x 6,3 x 10,8 mm

Fonte: Próprio autor.

Em conjunto com o sensor óptico foi utilizado um microcontrolador da família PIC modelo 12F675 indicado na Figura 30.

Figura 30 – Microcontrolador PIC 12F675.

Fonte: (MICROCHIP, 2017)

O microcontrolador utilizado é um microcontrolador de baixa complexibilidade, que possui uma arquitetura do tipo *Reduced Instruction Set Computer* (RISC), que possibilita desenvolver um sincronismo muito mais preciso, pelo fato deste gastar a mesma quantidade de tempo para executar diferentes instruções e também pelo seu tamanho e preço reduzido, suas características técnicas são descritas na Tabela 3. Este simplesmente conta um numero pré-determinado de pulsos provenientes do sensor óptico e gera um pulso em um de seus pinos de I/O digital que está conectado ao *PI*, este quando recebe o pulso habilita a capturar a imagem e realizar a contagem dos ovos.

Tabela 3 – Especificações técnicas microcontrolador PIC 16F675.

Fabricante	*Microchip*
Arquitetura	PIC
Bits	8
Pinos	8
ADC	4 canais de 10 bits
Comparador	1
EEPROM	128 Bytes
Memória de programa	Flash 1,75 Kb
RAM	64 Bytes
Timers	1 x 8-bit, 1 x 16-bit
Faixa de temperatura (ºC)	- 40 a 125
Faixa de tensão de operação (V)	2 a 5.5

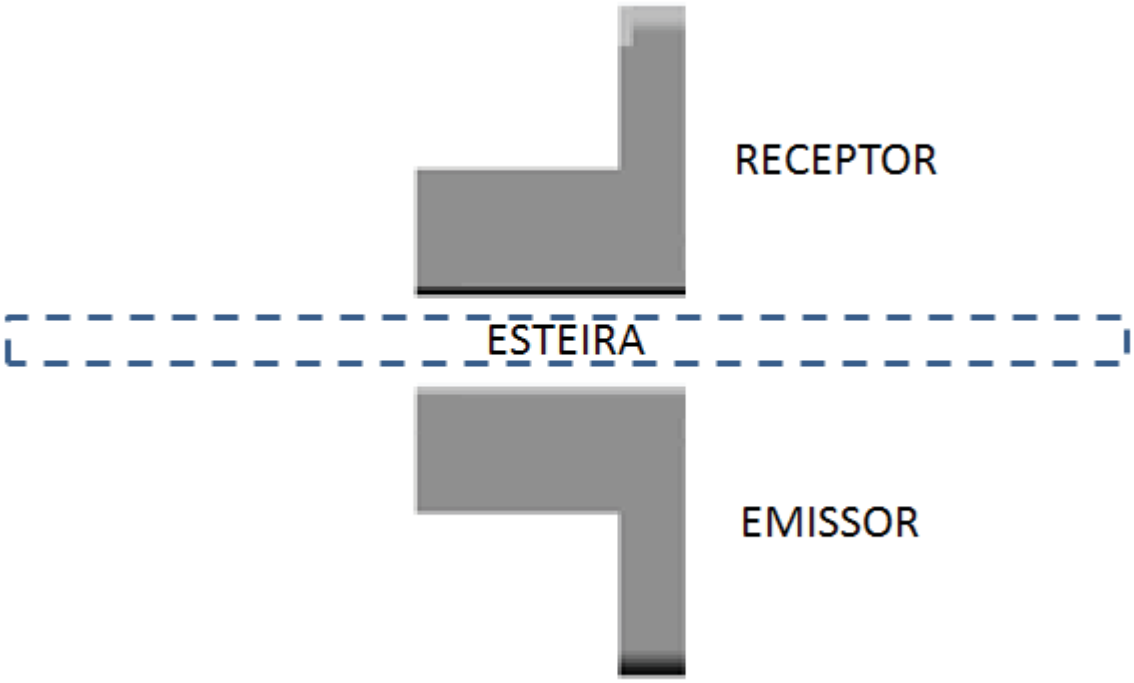
Fonte: Próprio autor.

O sensor óptico tem como base de funcionamento a emissão de um feixe de luz pelo emissor, o qual é recebido por outro elemento foto-sensível chamado de receptor. A forma de sensoriamento utilizada neste projeto é o sistema por barreira, este consiste no emissor

instalado de forma alinhada com o receptor, deste modo toda vez que algum objeto interromper o feixe de luz emitido pelo emissor o receptor alterará o estado lógico de sua saída.

A instalação do sensor esta organizada da seguinte forma, instalamos o receptor sobre a esteira e o emissor na parte inferior da esteira conforme Figura 31.

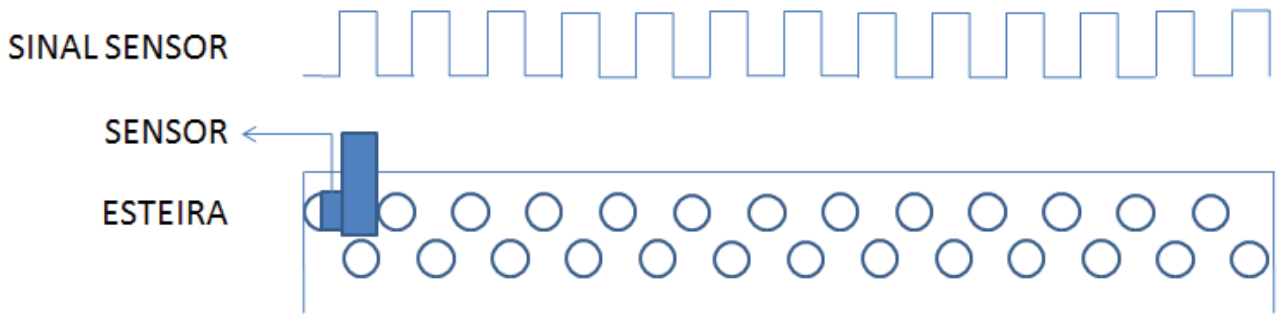
Figura 31 – Instalação sensor óptico.



Fonte: Próprio autor.

Com isso a esteira cruza entre o emissor e o receptor. Como a esteira é perfurada entre um orifício e outro é gerado um sinal conforme ilustrado na Figura 32 que é interpretado pelo microcontrolador que realiza a contagem dos orifícios.

Figura 32 – Sinal sensor óptico.

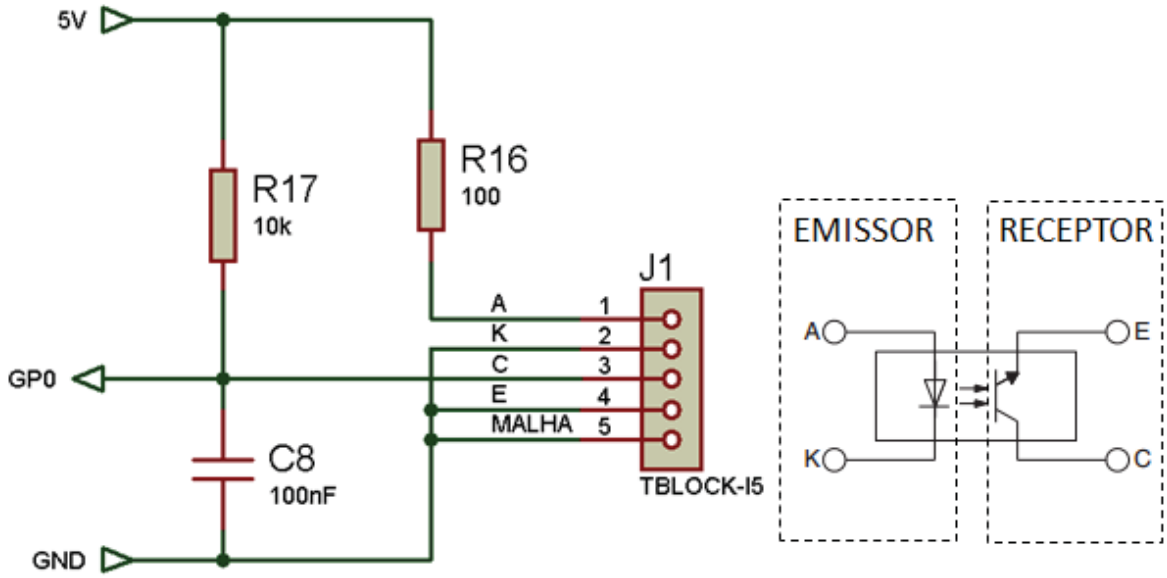


Fonte: Próprio autor.

A interligação do sensor com a placa de sincronismo é realizada através de um cabo 4 X 18 AWG com malha aterrada em uma de suas extremidades. Foi necessária a utilização deste cabo para atenuarmos a interferência de ruído e tornarmos este sistema mais robusto.

O circuito utilizado para interligar o sensor ao microcontrolador está indicado na Figura 33, onde R16 é utilizado para limitar a corrente de alimentação do circuito emissor do sensor, R17 é utilizado para polarizar o fototransistor do circuito receptor, em paralelo com o fototransistor é ligado o capacitor C8, que tem como função limitar o ruído.

Figura 33 – Circuito de interligação sensor com o microcontrolador PIC.



Fonte: Próprio autor.

A cada 17 pulsos contados o microcontrolador gera um pulso de 1 milisegundo em um de seus pinos de I/O digital, 17 pulsos correspondem ao comprimento da área capturada pela câmera, esta quantidade de pulsos é obtida a partir da equação (8),

$$N = \frac{C}{L_f + D} \tag{8}$$

Onde:

N é o número de pulsos.

C é o comprimento da área de captura em cm.

L_f é o diâmetro do furo em cm.

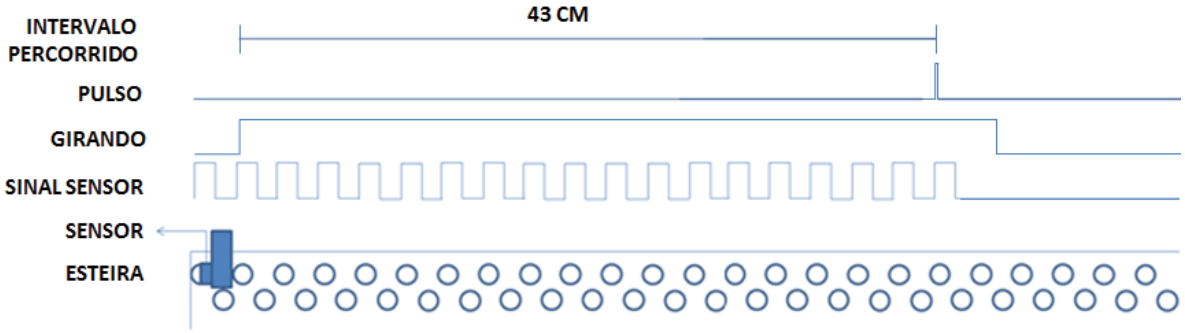
D é a distância entre os furos em cm.

$$N = \frac{43}{2 + 0{,}53} \tag{9}$$

$$N = 16{,}99 \cong 17 \tag{10}$$

O pulso é interpretado pelo *Raspberry* como sendo o momento para a captura da imagem. Em paralelo com a contagem o 12F675 está programado de modo que enquanto a esteira esta girando o microcontrolador mantem nível lógico alto em outro pino, isso é realizado através do mecanismo de interrupção do *TIMER0*, que a cada pulso é realizado o *reset* do registrador do *TIMER0*. Caso a esteira esteja parada, ou seja, o *TIMER0* não é resetado é mantido nível lógico alto no pino responsável por indicar-se a esteira esta girando por mais um curto período de tempo, percorrido esse tempo de atraso, o pino é colocada em nível lógico baixo, passando a indicar que a esteira está parada. Na Figura 34 é apresentado o diagrama de estados do circuito em questão.

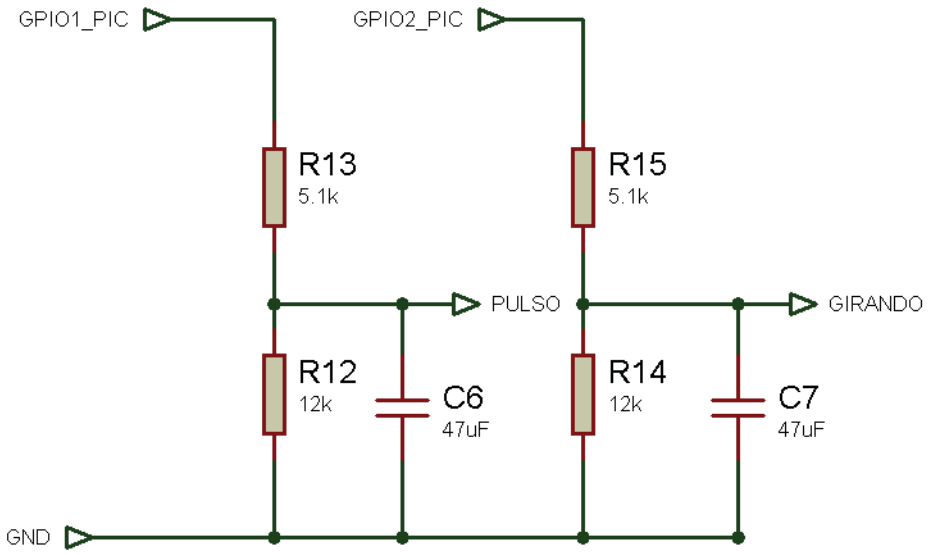
Figura 34 – Diagrama de estados sinal de sincronismo.



Fonte: Próprio autor.

Como o *Raspberry* não pode receber em suas entradas sinais com tensão superior a 3,3 V se faz necessário a utilização de um divisor de tensão para garantirmos que esta tensão não seja ultrapassada, em paralelo com os resistores que estão conectados ao GND é colocado um capacitor de 47 uF que tem como função filtrar o sinal limitando o ruído.

Figura 35 – Circuito de saída sinal de sincronismo e esteira girando.



Fonte: Próprio autor.

3.3 SISTEMA ELETRÔNICO DE PROCESSAMENTO

Nesta seção será descrito as características do *hardware*, câmera, linguagem de programação que foram utilizadas e as configurações necessárias para o sistema operar.

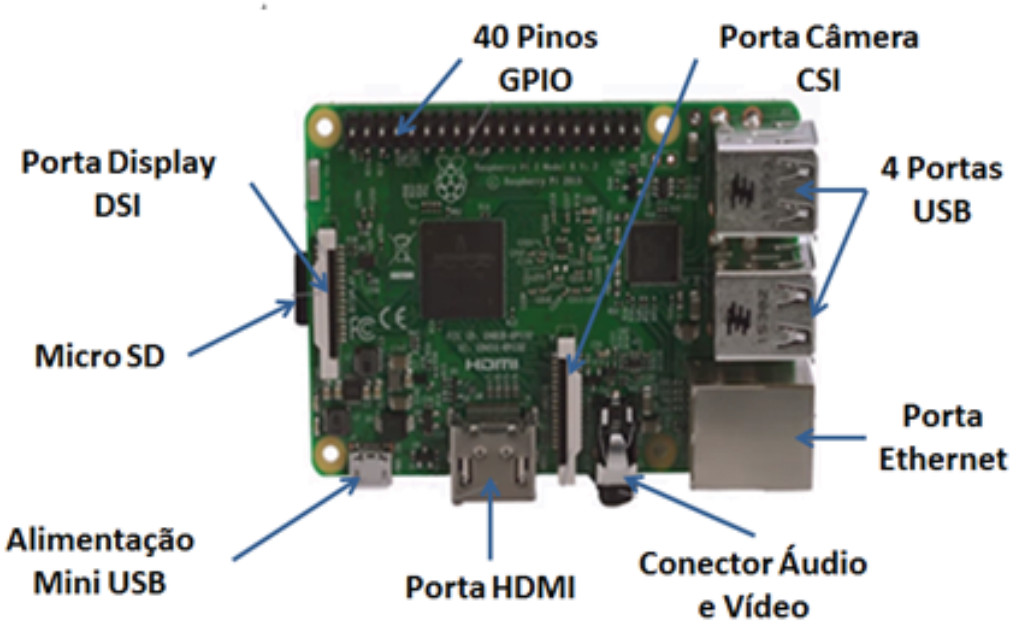
3.3.1 HARDWARE DE PROCESSAMENTO

Para realizar a captura, processamento da imagem e detecção dos ovos foi utilizado um microcomputador *Raspberriy Pi* 3 modelo B em conjunto com uma Câmera *Raspberry Pi*. Pelo fato deste possuir uma capacidade de processamento elevada, que é uma necessidade do projeto em questão, possuir periféricos integrados em sua estrutura, compatível um sistema operacional, interface gráfica, periféricos comuns como mouse e teclado. Além disso, não possui um custo beneficio elevado e não ser necessário à utilização de outro computador para realizar sua programação.

3.3.1.1 Raspberry Pi 3

Existem vários modelos de microcomputadores *Raspberry Pi*, porém no desenvolvimento deste projeto optamos por utilizado um microcomputador *Raspberry Pi 3* modelo B que possui a estrutura ilustrada na Figura 36.

Figura 36 – Estrutura Raspberry Pi 3.



Fonte: Adaptado de KHANDELWAL (2016).

As características técnicas do dispositivo estão descritas na Tabela 4.

Tabela 4 – Especificações técnicas Raspberry Pi 3.

Marca	*Raspberry Pi*
Versão	3
Modelo	B
CPU	*Quad-core* 64 bits ARMv8
Clock	1.2 GHz
Wireless	LAN 802.11n
Bluetooth	4.1 BLE
Memória RAM	1 GB
Portas USB 2.0	4
GPIO	40
Porta Ethernet	1
Porta HDMI	1
Conector Áudio e Vídeo	1
Interface para Câmera	1
Interface para *Display*	1
Micro SD Card	1
Corrente de Alimentação	2.5 A
Tensão de Alimentação	5 V
Dimensões	85 x 56 x 17 mm
Peso	42 g

Fonte: Próprio autor.

Na Tabela 5 é apresentado o descritivo de cada pino da GPIO, sendo indicadas para cada pino quais suas funções possíveis.

Tabela 5 – Funções dos 40 pinos da GPIO

Pino	Função	Pino	Função
1	3V3	2	+5 V
3	GPIO2 / SDA1	4	+5 V
5	GPIO3 / SCL1	6	GND
7	GPIO4	8	TXD0 / GPIO14
9	GND	10	RDX0 / GPIO15
11	GPIO17	12	GPIO18
13	GPIO27	14	GND
15	GPIO22	16	GPIO23
17	+3V3	18	GPIO24
19	GPIO10 / MOSI	20	GND
21	GPIO9 / MISO	22	GPIO25
23	GPIO11 /SCLK	24	CE0# / GPIO8
25	GND	26	CE1# /GPIO7
27	GPIO0 /ID_SD	28	ID_SC / GPIO1
29	GPIO5	30	GND
31	GPIO6	32	GPIO12
33	GPIO13	34	GND
35	GPIO19 / MISO	36	CE2# /GPIO16
37	GPIO26	38	MOSI / GPIO20
39	GND	40	SCLK / GPIO21

Fonte: Adaptado de EAMES et al., (2015).

O *Pi* foi conectado a uma câmera *Raspberry Pi* através da porta CSI e a uma placa de sincronismo através das GPIO15, GPIO18, GPIO23 e GPIO24, os quais são responsáveis por realizar a aquisição das imagens no momento certo, resetar o sistema de sincronismo e ativar o sistema de iluminação da caixa de captura. As imagens adquiridas são processadas pelo *Pi,* o qual informa o resultado do processamento da imagem em um monitor que está conectado na sua saída de *High-Definition Multimedia Interface* (HDMI).

Antes de iniciar com o desenvolvimento do algoritmo é necessário preparar o microcomputador, para isso se fez necessário à instalação do sistema operacional, ativação de periféricos, instalação de bibliotecas e atualização das bibliotecas que o acompanham.

O *Raspberry Pi*, como uma plataforma livre, possui diversos sistemas operacionais disponíveis para utilização, dependendo da necessidade do usuário. Neste projeto, foi utilizado

o *Raspbian* 1.9.2, sistema operacional em *Linux*, que tem como base o sistema operacional *Debian*, só que específico para o *Pi*. Sua instalação e simples e prática, apenas necessita de acesso à internet e de um computador auxiliar para fazer os preparativos. No próprio site da *Raspberry* (https://www.raspberrypi.org/downloads/ acessado em 16 de janeiro de 2017) é possível realizar o *download* do sistema operacional e neste estão disponíveis os documentos com as informações necessárias para realização desta etapa.

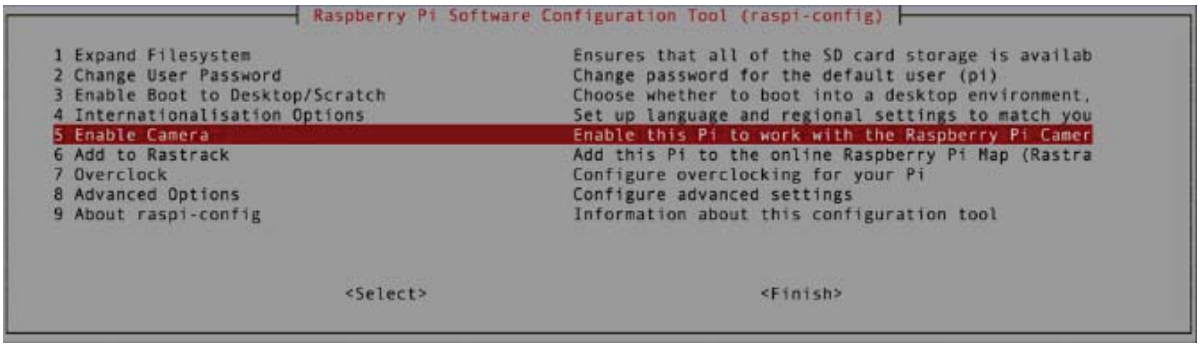
Além da instalação do sistema operacional é necessário habilitar os periféricos que serão utilizados, isso é realizado através da linha de comando do *Raspberry Pi*, que é chamada de *LXTerminal* observado na Figura 37 que pode ser acessado pelo menu principal. A partir do *LXTerminal* é possível configurar, atualizar programas instalar novos programas, bastando apenas saber a linha de comando adequada para cada situação.

Figura 37 – LXTerminal.

Fonte: Próprio autor.

Para acessar as configurações do *Pi* basta utilizar a linha de comando *sudo raspi-config* e é mostrada a tela de configuração Figura 38, onde é possível ajustar uma série de configurações, neste projeto apenas expandimos a memória do cartão SD através do comando *Expand Filesystem* no menu de configurações e habilitamos a câmera no menu *Enable Camera* os menus citados podem ser navegados através do teclado, após realizada as alterações necessária é somente utilizar o comando <*Finish*>. Assim o *Pi* irá reiniciar com as alterações realizadas.

Figura 38 – Menu configuração *Raspberry Pi*.



Fonte: Adaptado de RASPBERRY PI FOUNDATION (2016).

3.3.1.2 Raspberry Pi Câmera

Para a aquisição das imagens foi utilizado uma câmera *Raspberry Pi* que foi conectada ao *Raspberry Pi 3*, pois este já possuir uma interface para esta câmera, além desta possui uma velocidade de captura superior a uma câmera USB normal de mercado, tendo menor latência nas capturas, ter uma boa profundidade de cor, ter filtros *Anti-aliasing* que possibilitam capturas sem interferências entre as cores, utilizar um sensor de imagem *Complementary Metal Oxide Semiconducto* (CMOS) modelo OV5647 da OmniVision, capturar imagens com resolução de 1080p, com velocidade de até 30 quadros por segundo, sendo possível até 120 quadros por segundo utilizando uma resolução de 320 x 240. As características técnicas do dispositivo estão descritas na Tabela 6.

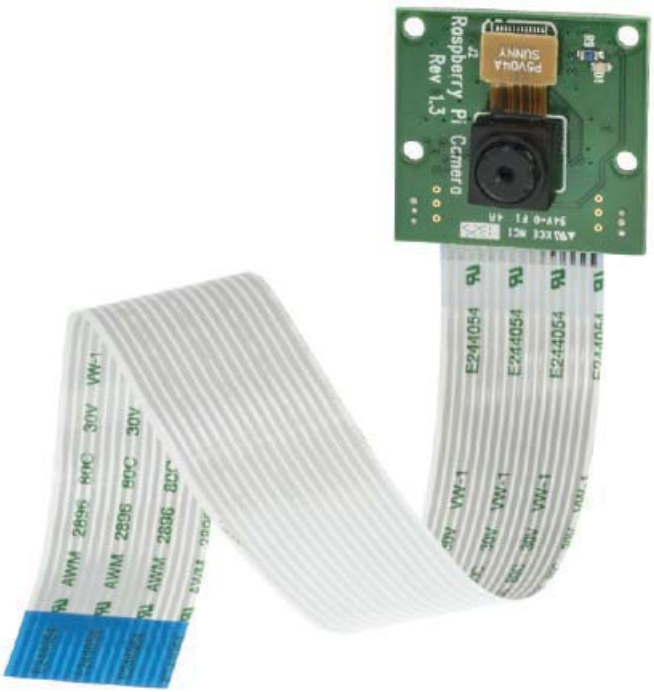
Tabela 6 – Especificações técnicas Camera Board.

Marca	*Raspberry Pi*
Versão	1
Modelo	*Camera Board*
Resolução	1920 x 1080
Profundidade de cor	8/10 bits
Quadros por segundo	30 Fms
Sensor de imagem	CMOS
Comunicação	CSI - 2
Foco	Fixo
Ângulo de visão	24º
Dimensões	20 x 25 x 10 mm
Peso	3g

Fonte: Próprio autor.

Uma imagem do dispositivo é apresentada na Figura 39.

Figura 39 – Câmera *Raspberry Pi*.



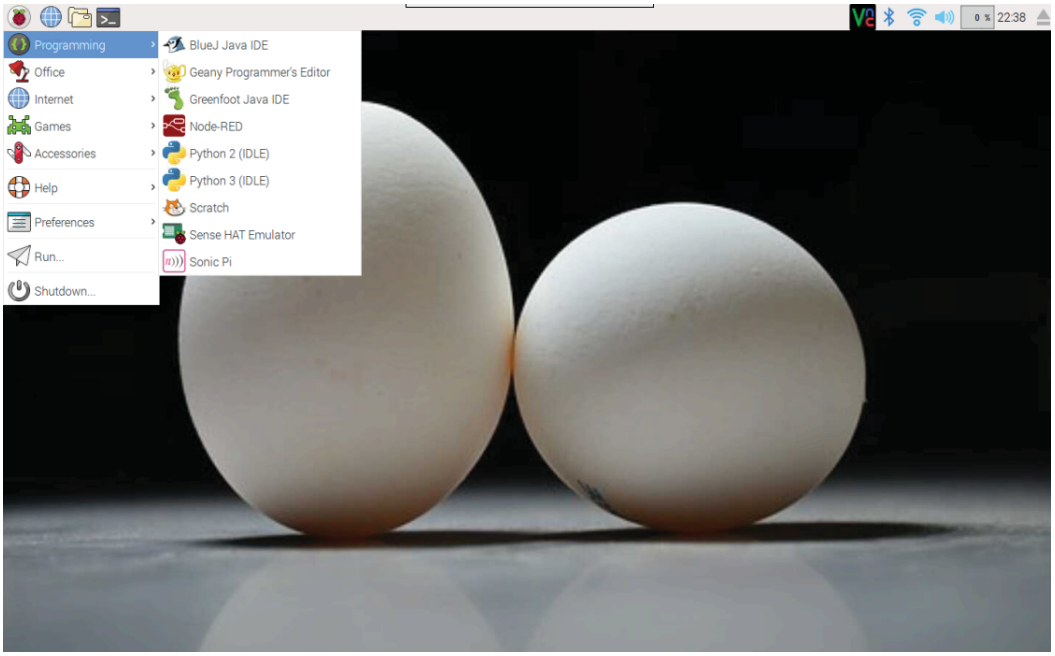
Fonte: Adaptado de RASPBERRY PI FOUNDATION (2016).

3.3.2 Linguagem de Programação *Python*

A linguagem de programação *Python* é considerada uma linguagem de alto nível que detém uma sintaxe clara e concisa que facilita a legibilidade do código-fonte transformando-a em uma linguagem mais produtiva. Além disso, possui uma grande quantidade de funções semelhante à linguagem de programação C, com pequenas variações de formatação, porém com tempo de execução de programa superior ao de um programa escrito em C.

Pelo fato de ser uma linguagem clara, disponível no *Pi* e atender a necessidade do projeto foi optado por utilizar a linguagem de programação *Python.* O compilador de *Python* do *Raspberry Pi* é o *IDLE*, ele possui duas versões, uma para *Python* 2 e outra para *Python* 3, como pode ser visto na Figura 40. A diferença entre as versões é baseada em diferenças de sintaxe. Para este projeto foi utilizado *Python* 3.

Figura 40 – Compiladores disponíveis no Pi.



Fonte: Próprio autor.

3.3.3 Bibliotecas *Python*

No desenvolvimento deste projeto se faz necessário a utilização de algumas bibliotecas que facilitam e simplificam o código fonte, na sequência são indicadas as bibliotecas utilizadas bem como os passos para obtenção destas e suas atualizações.

3.3.3.1 Bibliotecas padrões Raspberry Pi

Como neste projeto optamos por trabalhar com o sistema operacional *Raspbian* este já possui algumas bibliotecas incorporadas, que serão detalhadas neste capítulo, não sendo necessário um procedimento de instalação, somente um procedimento de atualização para garantir que estamos utilizando a versão mais atualizada de todas as bibliotecas padrão, para isso basta conectar o *Pi* a internet e utilizar os comandos indicados abaixo no *LXTerminal* e automaticamente é realizado o *download* e a instalação das atualização destas bibliotecas.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

3.3.3.1.1 Time

A biblioteca *Time* fornece varias funções relacionadas a tempo para utilizar esta biblioteca basta adicionar a linha de comando *import time* no *script* de programa através deste comando é possível desfrutar das funções pertinentes a esta biblioteca, abaixo é informado e descrito a função desta utilizada no desenvolvimento deste projeto.

Neste projeto utilizamos a função *sleep( )*, que é uma função utilizada para gastar tempo. Esta suspende a execução do *thread* atual. O argumento que é passado pra está função é o tempo em segundos que se deseja gastar, este pode ser um número de ponto flutuante para indicar um tempo de parada mais preciso.

3.3.3.1.2 Picamera

A *Picamera* é uma biblioteca que fornece uma interface *Python* pura para o módulo de câmera do *Raspberry Pi*. Que possibilita a capturar de imagem e gravação de vídeo bem como realizar algumas configurações, para que a captura tenha maior aproveitamento em cada diferente condição de captura.

Para utilizar esta biblioteca basta adicionar a linha de comando *from picamera import PiCamera* no *script* de programa através deste comando é possível desfrutar das funções pertinentes a esta biblioteca, abaixo são informados e descritos algumas destas que foram utilizadas no desenvolvimento deste projeto.

A especificação do arquivo de saída e realizada a partir da linha de comando.

with picamera.PiCamera() as camera:

A configuração da resolução da imagem é realizada através da função indicada abaixo, onde é necessário informar os argumentos X e Y, sendo X a quantidade de *pixels* na horizontal e Y a quantidade de *pixels* na vertical.

camera.resolution = (X, Y)

É possível pré-visualizar a imagem antes mesmo de realizar a captura, com o comando *camera.start_preview()*. Está pré-visualização é mostrada na tela do *Raspberry Pi* e tem grande utilidade no desenvolvimento do projeto no que diz respeito aos ajustes mecânicos para tornar a

captura mais precisa. A finalização dá pré-visualização é realizada utilizando o comando *camera.stop_preview()*.

A captura da imagem é realizada utilizando o comando *camera.capture()* o argumento que devemos informar deve ser colocado entre aspas simples e deve conter o nome do arquivo e sua extensão, como por exemplo *('foto.jpg')*, o resultado deste comando é um arquivo de imagem que pode ser dos seguintes formatos *jpg*, *png* e *gif*.

3.3.3.1.3 Picamera.array

A *Picamera.array* é uma biblioteca fornece um conjunto de classes que auxiliam na construção de *arrays numpy* n-dimensionais a partir da saída da câmera *Pi* este módulo não é automaticamente importado pelo pacote *picamera* principal e deve ser explicitamente importado.

Para utilizar esta biblioteca basta adicionar a linha de comando *import picamera.array* no *script* de programa através deste comando é possível desfrutar das funções pertinentes a esta biblioteca.

Através deste do módulo é possível realizar a captura de uma imagem, sendo que ao invés de obter um arquivo de imagem é possível armazenar esta imagem em formato de uma matriz que é armazenada no *buffer* do *Pi*, possibilitando que esta imagem seja processada de forma mais ágil e fácil o comando que ativa este módulo é indicado a seguir.

with picamera.array.PiRGBArray(camera) as stream:

Após o comando de ativação basta utilizar o comando *camera.capture(stream, format='bgr')*, que realiza a captura da imagem, este comando permite realizar a captura da imagem no formato RGB.

Com a imagem capturada é possível salvar esta em uma variável em formato de matriz através do comando *image = stream.array*, este faz com que a matriz seja salva na variável *image* e assim pode ser processada como uma matriz normalmente.

3.3.3.1.4 GPIO

Através da biblioteca GPIO é possível configurar e comandar os 26 pinos GPIO do *Pi*, estes pinos são a interface física do *Pi* para o mundo externo, quando configurados como saída

podem ser utilizados para dar um comando a um equipamento externo, quando configurados como entrada podem ser utilizados para realizar a leitura de um estado lógico proveniente de um sensor, bem como através destes é possível que o *Pi* se comunique com outros periféricos com comunicações do tipo I2C, SPI ou UART.

Para utilizar esta biblioteca basta adicionar a linha de comando *import RPi.GPIO as GPIO* no *script* de programa através deste comando é possível desfrutar das funções pertinentes a esta biblioteca, abaixo são informados e descritos algumas destas que foram utilizadas no desenvolvimento deste projeto.

A configuração dos pinos como entrada ou saída, ativação de resistor de *pull down* ou *pull up* é realizada através da linha de comando indicada abaixo.

GPIO.setup(16, GPIO.IN,GPIO.PUD_DOWN)

Sendo o argumento *GPIO.IN* que configura o pino como entrada e *GPIO.OUT* como saída, o argumento *GPIO.PUD_UP* configura os resistores de *pull up,* o argumento *GPIO.PUD_DOWN* realiza a configuração dos resistores de *pul downl* referente a GPIO indicada, no exemplo informado está sendo configurado o pino físico 16 que se refere a GPIO 23.

Quando o pino é configurado como saída é possível alterar o nível lógico deste pino, isso é realizado através da linha de comando *GPIO.output(22, 1)*, onde o primeiro argumento é o pino físico que se deseja setar e o segundo argumento é o nível lógico que se deseja colocar deste pino.

Quando o pino é configurado como entrada é possível ler o nível lógico deste pino, isso é realizado através da linha de comando *GPIO.input(18)*, onde o argumento é o pino físico que se deseja ler.

3.3.3.2 Bibliotecas adicionais Raspberry Pi

Além das bibliotecas disponíveis no *Pi* se fez necessário a instalação de algumas bibliotecas adicionais, neste item abordaremos algumas de suas características e quais de suas funções foram utilizadas neste projeto.

3.3.3.3 OpenCV

A biblioteca *OpenCV* é uma biblioteca muito poderosa, que é livre para uso acadêmico e comercial, possui interfaces *C ++, C, Python e Java, suporta Windows, Linux, Mac OS, iOS e Android*, os passos para instalação destas biblioteca estão indicados no site (http://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/ acessado em 16 de janeiro de 2017).

Neste projeto foi optado por utilizar a interface *Python3*, utilizamos apenas algumas funções básicas como transformação de cor e a função para guardar algumas imagens geradas no processo de detecção e contagem dos ovos.

Para utilizar esta biblioteca basta adicionar a linha de comando *import cv2* no *script* de programa através deste comando é possível desfrutar das funções pertinentes a esta biblioteca, abaixo são informados e descritos algumas destas que foram utilizadas no desenvolvimento deste projeto.

A transformação da imagem capturada no formato RGB em tons de cinza é realizada através da função *cv2.cvtColor(captura, cv2.COLOR_BGR2GRAY),* onde *captura* é a imagem a ser transformada em tons de cinza, desta forma o retorno desta função é uma matriz de 1 camada.

Para guardar as imagens capturadas ou obter imagens de determinadas etapas do processo foi utilizado a função *cv2.imwrite('imagem.png',img)*, onde o primeiro argumento é o caminho a partir da pasta do programa até onde se deseja guardar a imagem e o segundo argumento é a imagem a ser guardada, a função em questão permite guardar imagens em diferentes formatos como JPEG, PNG 2000 entre outros.

3.3.3.4 Tkinter

A biblioteca *Tkinter* oferece subsídios para o desenvolvimento de interfaces gráficas em *Python*. A partir deste módulo é possível criar janelas, botões, menus entrar com dados para o programa entre outras aplicações.

Esta biblioteca não faz parte do pacote de bibliotecas disponíveis no *Pi*, sua instalação é relativamente simples, é realizada através do *LXTerminal*, para realizar sua instalação basta saber qual é a biblioteca *Python* que está sendo utilizada, neste projeto foi utilizado *Python 3*, desta forma para realizar a instalação é apenas necessário digitar a linha de comando indicada abaixo no *LXTerminal*.

$ sudo apt-get install python3-tk

Para utilizar esta biblioteca basta adicionar as linhas de comando indicadas abaixo no *script* de programa.

```
from tkinter import *
from tkinter import ttk
```

Através destes comandos é possível desfrutar das funções pertinentes a esta biblioteca, abaixo são informados e descritos algumas destas que foram utilizadas no desenvolvimento deste projeto.

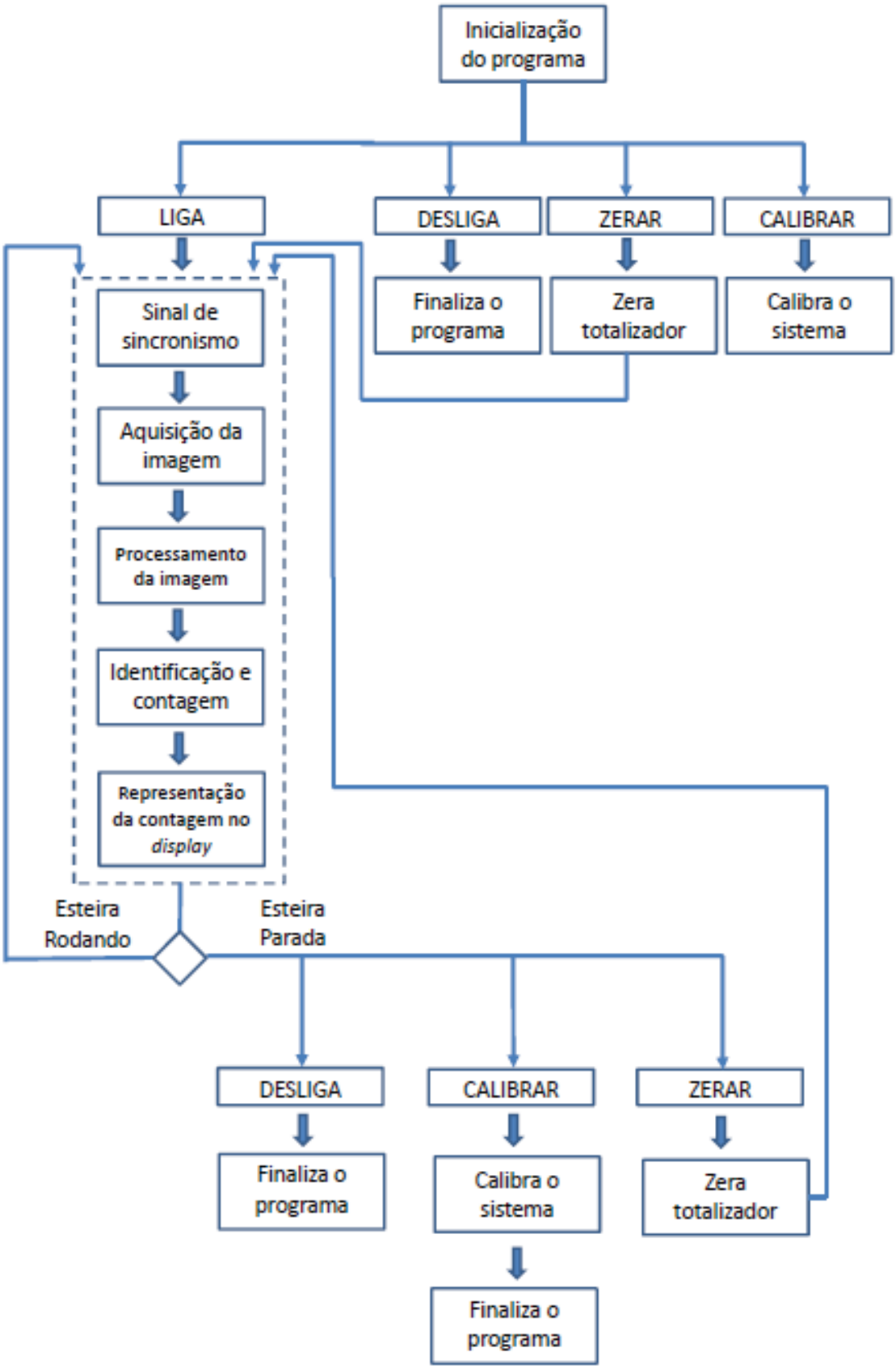
A criação de uma janela é realizada através do comando *janela = Tk()*, o nome testa janela é dado utilizando a função *janela.title()*, com a janela criada é possível criar botões, apresentar textos na interface, carregar imagens.

Neste projeto foi desenvolvido uma interface simples onde possui alguns botões com funções especificas que foram criados utilizando a função *ttk.Button()*, foram criadas caixas de textos e exibida a imagem de uma das etapas da detecção e contagem de ovos através da função *ttk.Label()*.

3.4 SISTEMA DE PROCESSAMENTO DE CONTAGEM DE OVOS

Neste capítulo será apresentada a lógica de funcionamento do algoritmo, como é realizado o acesso ao programa e a interface desenvolvida para este projeto, na Figura 41 é apresentado o diagrama funcional do algoritmo desenvolvido.

Figura 41 – Fluxograma do sistema.



Fonte: Próprio autor.

3.4.1 Acesso ao programa

O acesso ao programa é realizado através de um ícone criado na área de trabalho do *Raspberry* chamado de Contador de Ovos, o ícone pode ser visto na Figura 42.

Figura 42 – Ícone acesso Contador de Ovos.

Fonte: Próprio autor.

O procedimento de criação do ícone consiste na criação de uma pasta com todos os arquivos necessários para a aplicação, através do *LXTerminal* utilizando o comando *touch nomedoatalho.desktop* é criado um arquivo vazio com extensão *.desktop*, a configuração do arquivo é realizada através do comando *nano nomedoatalho.desktop*, a partir deste comando é possível configurar o arquivo para torna-lo um ícone para o executável ou seja escrever os comandos no arquivo em branco, os comandos a serem inseridos neste arquivo são:

Comando de identificação de lançadores no d*esktop* é o comando descrito abaixo, este permite que o sistema operacional interprete o arquivo como um lançador do *desktop*.

[Desktop Entry]

A formatação da escrita é realizada através do comando.

Encoding=UTF-8

O nome do atalho é dado utilizando o comando.

Name=Nomedoatalho

A inserção de um comentário do tipo qual a função do ícone criado é realizado através do comando.

Comment=Atalho para aplicação de contagem de ovos

O tipo do arquivo que se deseja criar o ícone é informado através do comando.

Type=Tipo do arquivo

A desativação da inicialização automática e realizada através do comando descrito abaixo.

Terminal=false

A informação de qual arquivo deve ser executado é configurada no comando.

Exec=Caminho do executável

O logotipo do ícone é inserido através do comando descrito abaixo, este ícone é uma pequena imagem que preferencialmente deve estar salva na pasta da aplicação.

Icon=Caminho do Ícone

O comando que permite que a área de trabalho use qualquer notificação de inicialização incorporada na aplicação é indicado abaixo.

StartupNotify=true

A categoria do executável é informada através do comando.

Categories=Development;IDE;

Após a inserção dos comandos pressione Ctrl+x → S → Enter.

Feito isso é necessário dar permissão de execução para que o arquivo possa aparecer e ser executado, isso é realizado através do comando *chmod -R 777 Nomedoatalho.desktop*.

3.4.2 Interface do sistema

A partir da biblioteca apresentada no item 3.3.3.4 foi desenvolvida uma interface para o processo de detecção e contagem dos ovos a tela inicial da interface criada é apresentada na Figura 43.

Figura 43 – Interface contador de ovos.

Fonte: Próprio autor.

Esta interface é composta de quatro botões com funções distintas, um totalizador da contagem de ovos, uma *flag* que indica o estado da esteira e nela também é apresenta uma imagem de inicialização da aplicação, que quando ativado o sistema esta é substituída pelas imagens capturadas após o processo de binarização.

O *flag* que indica quando a esteira esta girando ou não nos indica também quando é possível manusear a interface, desta forma o comando dos botões só é aceito com a esteira parada garantindo que o usuário não interfira no processo de contagem.

Nos próximos itens serão abordadas as formas de funcionamento de cada botão deste sistema.

3.4.2.1 Botão LIGA

O botão LIGA é responsável por ativar o sistema de contagem de ovos, quando este é pressionado o algoritmo entra no *loop* de contagem de ovos, a saída deste *loop* só é permitida após a esteira estar parada por um determinado tempo, desta forma sendo possível a execução de outros comandos.

3.4.2.2 Botão ZERAR

O botão ZERAR é responsável por zerar o totalizador de ovos produzidos, este comando só é aceito com a esteira parada. Para realizar este procedimento basta dar um *click* sobre o botão ZERAR e o total de ovos produzidos passa a ser zero sendo assim iniciando um novo ciclo de contagem.

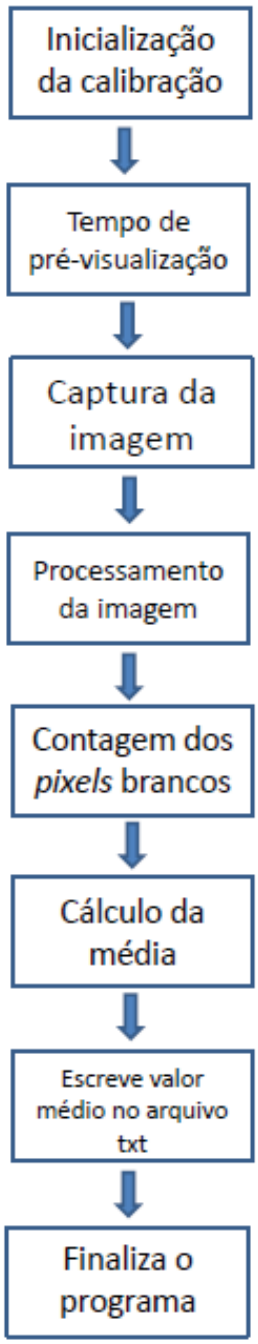
3.4.2.3 Botão DESLIGAR

O botão DESLIGAR é responsável por desligar e fechar o aplicativo, este comando só é aceito com a esteira parada. Para realizar este procedimento basta dar um *click* sobre o botão DESLIGAR e a interface será fechada e o sistema desativado.

3.4.2.4 Botão CALIBRAR

A calibração do sistema de contagem de ovos é realizada através do botão CALIBRAR, está é realizada utilizando seis ovos, estes devem ser posicionados sobre a esteira e no campo de visão da câmera, o processo de calibração é realizado para se obter a quantidade média de *pixels* brancos que representam um ovo, sendo este o valor base para realizar a contagem dos ovos, na Figura 44 é apresentado o diagrama de blocos deste processo.

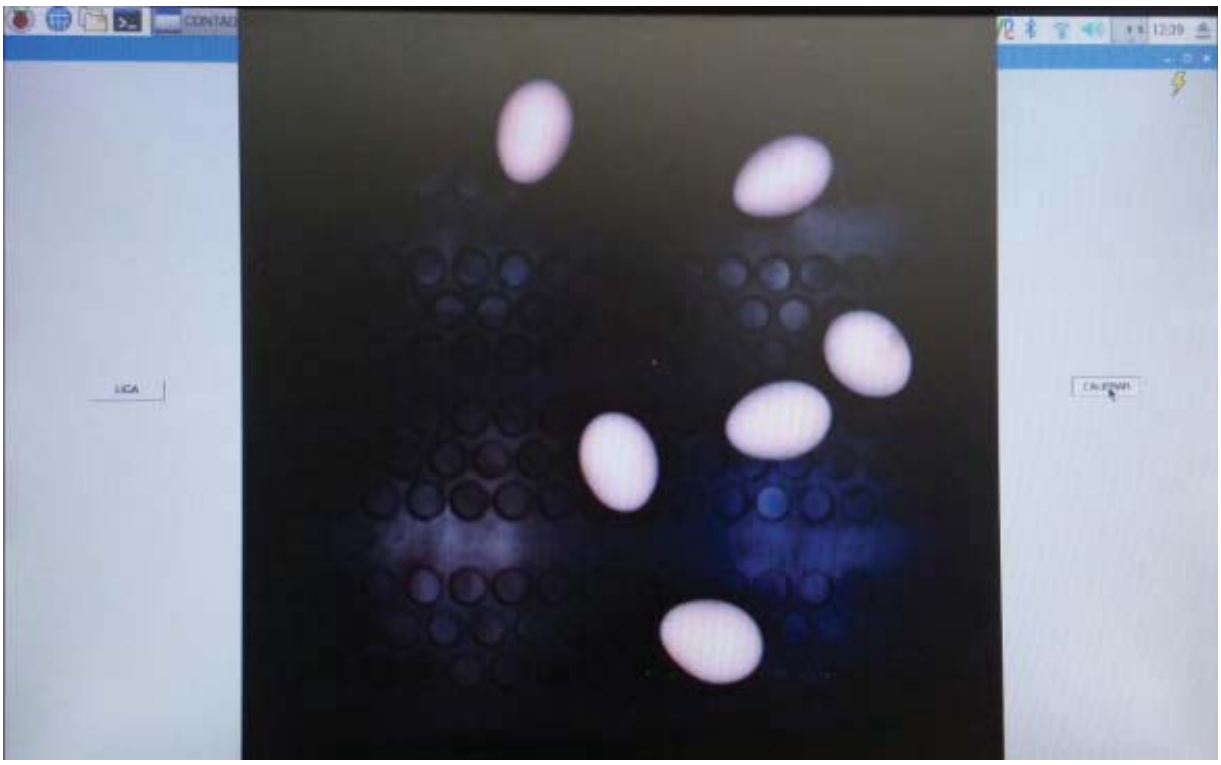
Figura 44 – Diagrama de blocos da calibração.



Fonte: Próprio autor.

A inicialização da calibração é realizada pressionando o botão CALIBRAR. O sistema entrará no modo calibração, estará sendo reproduzida no *display* do *Raspberry* por vinte segundos a imagem da esteira em tempo real fornecida pela câmara, para que seja visualizado se os ovos estão realmente posicionados no campo de visão da câmera conforme indicado na Figura 45, caso não estejam é possível realizar o juste neste intervalo de tempo.

Figura 45 – Imagem calibração do sistema



Fonte: Próprio autor.

Finalizado o intervalo de tempo, é capturada uma imagem e através do algoritmo desenvolvido são somadas as quantidades de *pixels* brancos da imagem. O total da soma é dividido por seis, desta forma temos a média de área dos ovos, este dado é salvo no arquivo *parametros.txt* que é o arquivo que possui todos os parâmetros do programa. Feito isso, a aplicação é fechada, sendo necessária inicializar o sistema novamente através do ícone *Contador de Ovos* para recarregar o novo parâmetro calibrado, o parâmetro de área média dos ovos é à base de todo o cálculo da quantidade de ovos presentes em uma imagem.

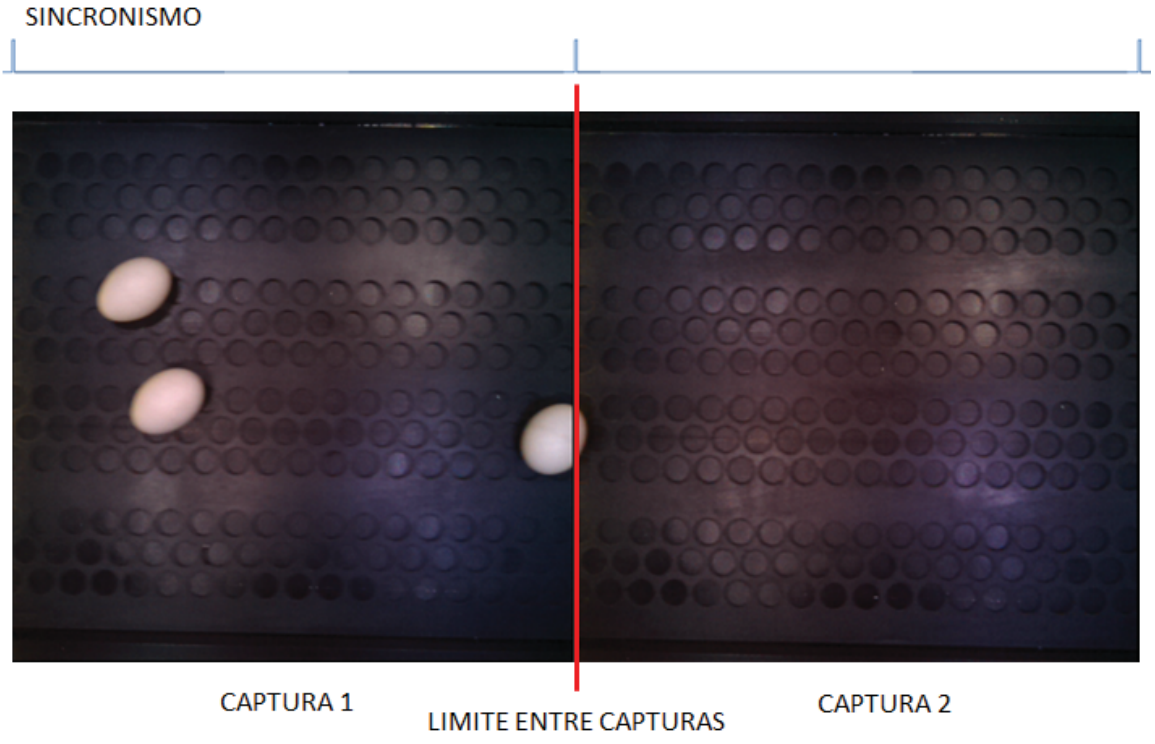
3.4.3 Algoritmo de detecção e contagem dos ovos

A partir do estudo teórico foram desenvolvidos vários algoritmos de maior complexidade que possibilitam diferentes formas de detecção, porém não obtiveram bom desempenho. Algoritmos estes baseados em transformadas como a de *Hough*, que é baseada na identificação do centro de circunferências ou elipses, desta forma em casos onde os ovos ficam fracionados entre capturas, não são possíveis de serem detectados.

Algoritmos baseados em casamento de padrões *(template matching),* também não conseguem detectar ovos fracionados entre capturas, além de demandar muito processamento tornando o processo de contagem lento. Algoritmos baseados na área delimitada dos ovos possibilita realizar a contagem dos ovos mesmo fracionados entre capturas, porém quando dois ou mais ovos estão encostados não é possível detectar a borda destes ovos, sendo este interpretados apenas como um ovo.

Sendo o mais indicado para a aplicação o algoritmo baseado na análise estatística dos *pixels* brancos de uma imagem binária. Afinal, este algoritmo permite contar os ovos que ficam com parte em uma captura e a outra parte na próxima captura conforme indicado na Figura 46, permite contar ovos encostados também, pois não é necessário reconhecer a borda do ovo para detecta-lo e possibilita a contagem de ovos em uma esteira "infinita".

Figura 46 – Sequência de captura.

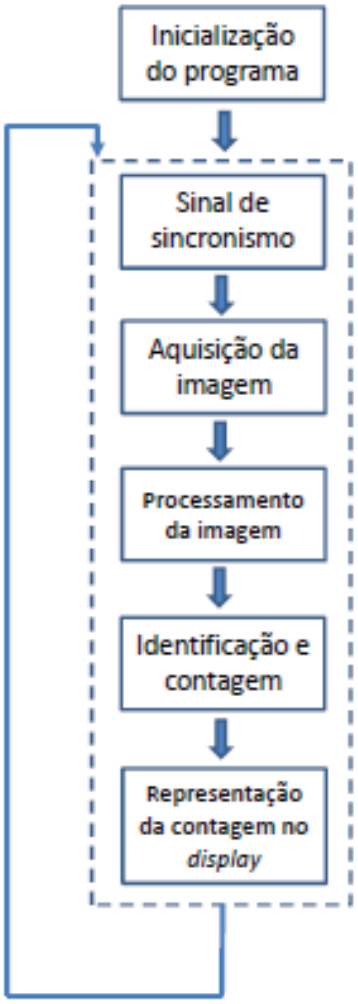


Fonte: Próprio autor.

Também este algoritmo possibilita contar ovos sem ter uma posição pré-determinada destes sobre a esteira. A contagem de ovos é realizada através de uma sequência de fotos capturadas da esteira em movimento.

Através de um algoritmo são detectados os ovos, contados e apresentado em um *display* o valor total de ovos contados. Na Figura 47 é apresentado um diagrama que resume o funcionamento do sistema.

Figura 47 – Diagrama de blocos sistema.



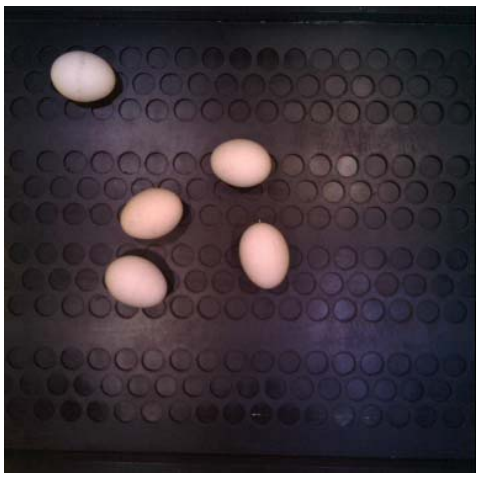
Fonte: Própio autor.

O algoritmo citado tem seu funcionamento assim descrito, ao inicializar a aplicação dando dois *clicks* sobre o ícone Contador de ovos na área do *Pi*. Todas as bibliotecas necessárias são carregadas, é realizada a leitura do arquivo *parametros.txt*. Nestes arquivos estão armazenados os valores de resolução, limiar de binarização e a área média dos ovos.

Com a interface inicializada ao pressionar o botão LIGA o sistema de contagem é ativado e fica aguardando um sinal gerado pelo sistema de sincronismo. Este sinal habilita a

captura da imagem. Recebido o sinal é realizada a captura de uma imagem da esteira no formato *RGB* conforme indicado na Figura 48, esta imagem é armazenada no *buffer* do *Raspberry*.

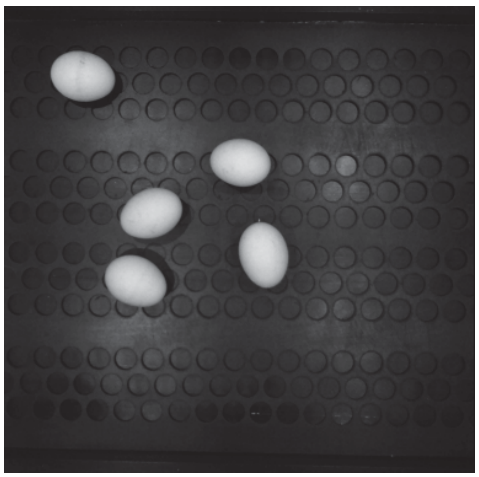
Figura 48 – Captura imagem da esteira formato RGB.



Fonte: Próprio autor.

A imagem armazenada no *buffer* passa pela etapa de processamento, que consiste em transformar a imagem RGB em tons de cinza, o resultado deste processo é mostrado na Figura 49.

Figura 49 – Imagem transformada em tons de cinza.

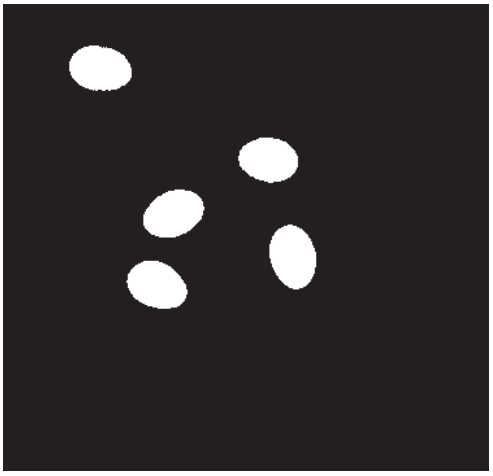


Fonte: Próprio autor.

Ainda na etapa de processamento é realizada a binarização da imagem, a partir do parâmetro PONTO DE CORTE BINARIZAÇÃO que é carregado na inicialização do aplicativo. Este parâmetro consiste no limiar de decisão da segmentação binaria, ou seja, para os *pixels* com valores maiores que esse valor o valor do *pixel* é alterado para 255 (branco) e

para os *pixels* com valores menores a este valor o valor do *pixel* é alterado para 0 (preto), resultando em uma imagem preta e branca conforme mostrado na Figura 50.

Figura 50 – Imagem binarizada.



Fonte: Próprio autor.

Finalizada a etapa de processamento, a imagem está pronta para passar pelo processo de contagem. Neste processo, é criada uma matriz *visitado*, que representa cada *pixel* da imagem como um *boolean*, sendo visitado (*True*) e não visitado (*False*). Todos os *pixels* da imagem binarizada com valor diferente de branco (255), são inicializados com *True*, e os *pixels* iguais a 255 são inicializados com *False.*

A partir da matriz *boolean,* o algoritmo percorre todos os *pixels* da imagem, verificando se o *pixel*, ainda não foi marcado como visitado. Quando for encontrado um *pixel* (x, y) não visitado, a variável *contadorpixel* é incrementada, na qual representa a quantidade de *pixels* brancos encontrados na imagem.

Após ter percorrido todos os *pixel* da imagem a soma total de *pixels* brancos é dividido pela quantidade de *pixels* média dos ovos, esta que é pré-determinada a partir da calibração do sistema. A quantidade de ovos contados na imagem é somada com a quantidade acumulada, e passando por um critério de arredondamento esta é representada na interface gráfica do sistema, após isso é representada a imagem binarizada na interface gráfica do sistema.

Realizado o processo de detecção e contagem o sistema volta a aguardar o sinal de sincronismo, para realizar o processo novamente em uma nova captura. Caso a esteira seja parada por um longo intervalo de tempo o *Raspberry* receberá um sinal proveniente da placa de sincronismo. Desta forma o usuário poderá navegar pelos menus CALIBRAR, DESLIGAR e ZERAR, se a esteira retomar o movimento automaticamente o sistema é habilitado e retornará a aguardar o sinal de sincronismo para uma nova captura.

4 RESULTADOS E DISCUSÃO

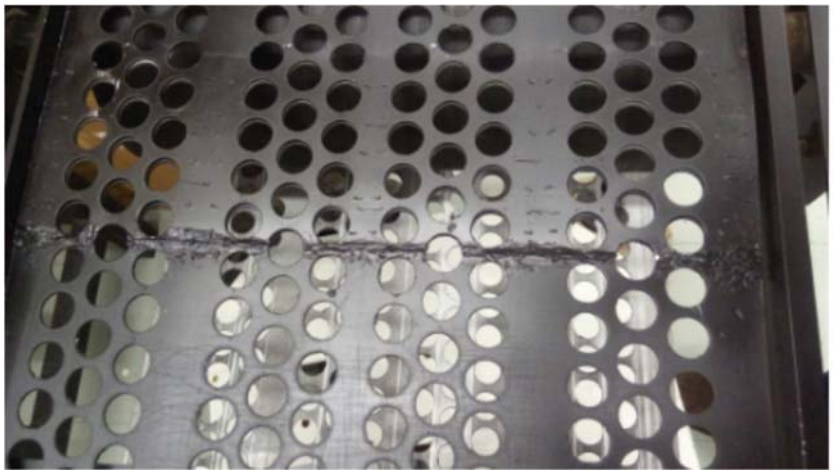
Neste capítulo será abordado o comportamento do projeto e seus resultados. Também serão discutidos os pontos fortes e fracos do projeto e seu desempenho durante os testes.

4.1 SISTEMA MECÂNICO

O sistema mecânico projetado auxiliou no desenvolvimento do algoritmo trazendo a situação mais próxima da realidade. Sendo de suma importância para o levantamento de resultados do projeto em questão.

A montagem da esteira neste sistema foi o item de maior dificuldade, devido à falta de equipamento para realizar a soldagem da emenda da esteira, e esta necessitar de um tensionamento adequado. A solução adotada está indicada na Figura 51. Ela é baseada em grampear as tuas extremidades da esteira, tendo como ponto negativo a sobreposição das duas esteiras. Isso faz com que quando a emenda passa pelo rolete menor acaba ocasionando uma variação no torque necessário pra girar a esteira, fazendo com que os ovos se movimentem sobre a esteira, sendo este efeito amenizado com o desenvolvimento de uma manivela com um braço de torque maior.

Figura 51 – Emenda esteira.



Fonte: Próprio autor.

O alinhamento da esteira sobre os roletes também não foi uma tarefa simples. Após a superação das dificuldades encontradas o sistema desenvolvido, se tornou apto e contribuiu positivamente nos resultados do projeto. Em Apêndice A, encontra-se uma imagem do sistema desenvolvido.

4.2 SISTEMA ELETRÔNICO DE ILUMINAÇÃO E CAMARA DE CAPTURA

O sistema de iluminação desenvolvido em conjunto com a câmera de captura tornou o sistema imune a variações da iluminação do ambiente externo, mantendo a luminosidade constante por todo o processo.

Neste sistema está fixada a câmera *Pi*, esta necessita de um ajuste muito preciso de sua altura em relação a esteira. A instalação e ajuste deste item necessitaram de um grande envolvimento para encontrar o ponto ideal, pelo fato deste opera em conjunto com o sistema de sincronismo e o ajuste inadequado ter grande influencia nos resultados finais do sistema.

Após a realização dos ajustes no sistema desenvolvido não foi mais necessário à intervenção durante as etapas de testes.

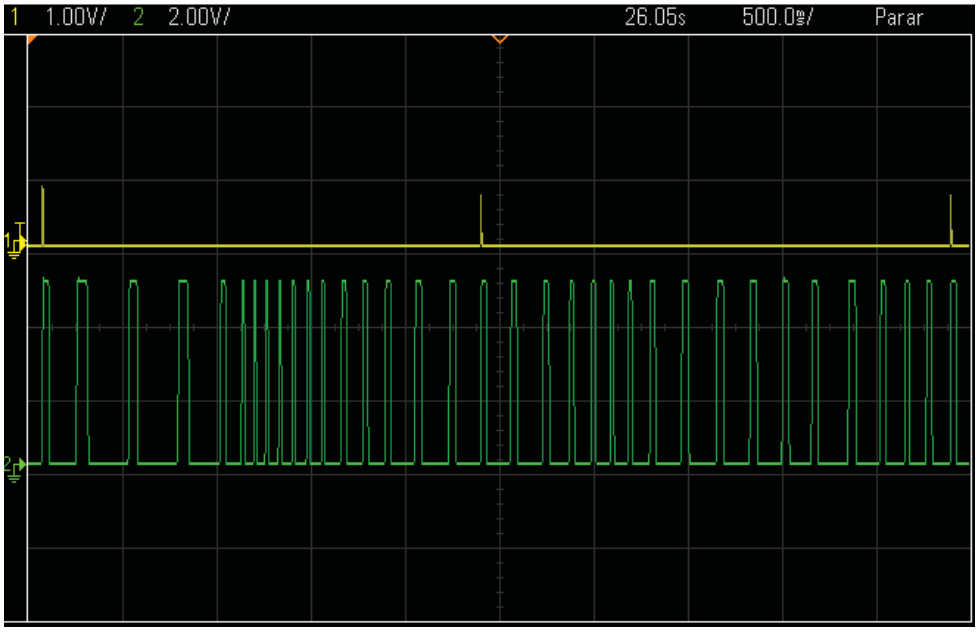
4.3 SISTEMA DE SINCRONISMO

O sistema de sincronismo desenvolvido se mostrou fundamental para o trabalho, além de possuir um baixo custo. Este foi construído a partir de componentes normais de mercado, na sua concepção foi utilizado um microcontrolador o qual possibilita o sistema ser ajustável. Em Apêndice B encontra-se o algoritmo desenvolvido para esta finalidade.

As limitações deste sistema estão ligadas no que diz respeito a um ajuste fino, pois o sensoriamento foi realizado através dos furos existentes na esteira. Desta forma, o ajuste da distância percorrida pela esteira entre uma foto e outra, pode ser ajustado alterando a quantidade de pulsos gerados por cada buraco. Sendo a distância percorrida de um furo a outro de aproximadamente 2,53 cm, mas como o sistema mecânico possibilita ajustar a altura da câmera em relação à esteira isso não se tornou um fator prejudicial aos resultados do projeto.

Na Figura 52 é apresentada uma imagem onde são apresentados os sinais gerados por este sistema, sendo o sinal verde o sinal gerado pelo sensor quando a esteira está em movimento, e o sinal amarelo o sinal de sincronismo para o *Raspberry* realizar a captura da imagem.

Figura 52 – Sinal de sincronismo.



Fonte: Próprio autor.

4.4 SISTEMA ELETRÔNICO DE CONTAGEM DE OVOS

O objetivo inicial do projeto foi alcançado com êxito, o qual consistia em desenvolver um equipamento de baixo custo, que realize a contagem de ovos através de imagens para ser potencialmente adaptado a uma esteira, e mostrar em um *display* a contagem total.

No geral, o *software* desenvolvido utilizando o *Raspberry Pi* funcionou com um bom desempenho, permitindo contar ovos brancos com tamanho semelhante em uma esteira em movimento em tempo real, obtendo uma porcentagem de erro menor que 1 % como ilustrado na Tabela 7. A linguagem de programação *Python* se mostrou muito eficiente e simples, permitindo realizar a captura da imagem, processar a imagem, detectar os ovos e realizar contagem dos ovos contidos na imagem capturada em menos de 3 segundos.

Se considerarmos um intervalo entre as capturas de 3 segundos e aplicando a equação (11) obteremos a velocidade máxima da esteira que este sistema pode contar os ovos.

$$V = \frac{d * 60}{t} \tag{11}$$

Onde:

d é o intervalo da captura em metros.

V é a velocidade da esteira em m/min.

t é o tempo entre as capturas em segundos.

$$V = \frac{0{,}43 * 60}{3} \tag{12}$$

$$V = 8{,}6\ m/min \tag{13}$$

Obteve-se que o sistema permite contar ovos em uma esteira com velocidade de 8,6 m/min, sendo superior ao valor de velocidade máxima de 8 m/mim da esteira em que este sistema poderá potencialmente ser instalado. Em Apêndice C encontrasse o algoritmo desenvolvido para esta finalidade.

Com o objetivo de levantar a porcentagem de erro por amostragem e acumulado do sistema foi realizada a contagem de 2000 ovos, divididas em 20 amostragens de 100 ovos cada, os resultados obtidos nestas amostragens está indicado na Tabela 7.

Tabela 7 – Resultados do sistema.

Amostragens	*Contagem total de ovos*	*Erro %* (e_n)
1º	100	*0*
2º	98	*-2*
3º	99	*-1*
4º	98	*-2*
5º	101	*1*
6º	100	*0*
7º	100	*0*
8º	102	*2*
9º	98	*-2*
10º	102	*2*
11º	100	*0*
12º	101	*1*
13º	100	*0*
14º	101	*1*
15º	101	*1*
16º	101	*1*
17º	99	*-1*
18º	102	*2*
19º	101	*1*
20º	100	*0*
Total	**2004**	**0,2**

Fonte: Próprio autor.

Na equação (14) está indicado à forma com que foi realizado o cálculo da porcentagem de erro do sistema por amostra, os resultados obtidos estão indicados na Tabela 7.

$$e_n = \left(\frac{C_n}{a_{padrão}} - 1\right) * 100 \tag{14}$$

Onde:

e_n é o erro em porcentagem do sistema em uma determinada amostragem.

C_n é o total de ovos contados pelo sistema eletrônico em uma determinada amostragem.

$a_{padrão}$ é o tamanho da amostragem.

Na equação (15) está indicado à forma com que foi realizado o cálculo da porcentagem de erro acumulado do sistema, onde o sistema contou 2004 ovos obtendo uma porcentagem de erro de 0,2 %.

$$e_{total} = \left(\frac{C_{total}}{a_{total}} - 1\right) * 100 \tag{15}$$

Onde:

e_{total} é o erro em porcentagem do sistema no total de amostragens realizadas.

C_{total} é o total de ovos acumulados contados pelo sistema eletrônico de contagem.

a_{total} é o total acumulado de todas as amostragens.

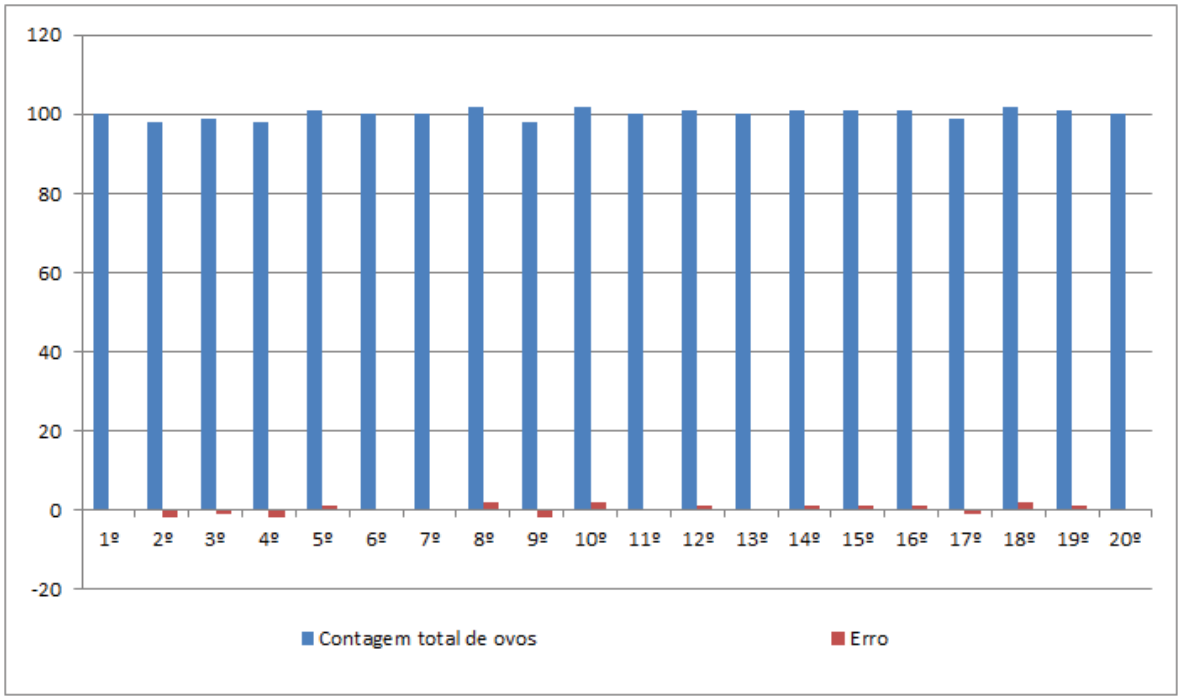
$$e_{total} = \left(\frac{2004}{2000} - 1\right) * 100 \tag{16}$$

$$e_{total} = 0{,}2\ \% \tag{17}$$

Observe que o erro total do sistema é menor que o erro obtido em cada amostragem. Isso é devido ao acúmulo dos resíduos de cada foto e ao fato de o algoritmo estar baseado no cálculo da média do tamanho dos ovos. Esta que tem sua maior assertividade quando o número de contagens tende ao infinito.

Na Figura 53 é apresentado o gráfico de desempenho do sistema, onde são apresentados os totais acumulados em cada amostragem e o respectivo erro de contagem.

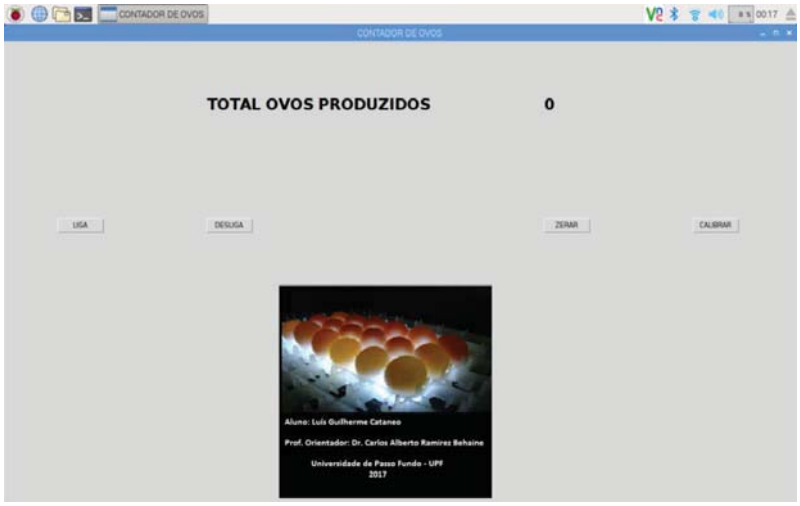
Figura 53 – Gráfico de desempenho do sistema.



Fonte: Próprio autor.

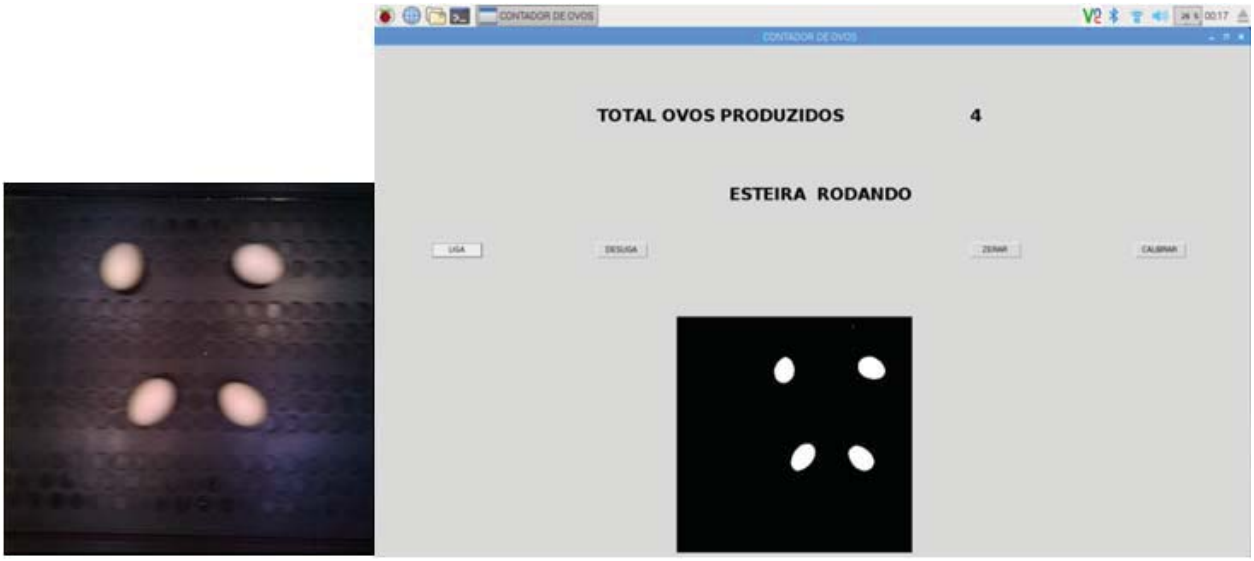
Nas Figura 54 à Figura 59 de encontra-se uma sequência de contagem de 10 ovos onde é apresentada a sequência de imagens capturadas com o respectivo total acumulado indicado na interface desenvolvida.

Figura 54 – Tela inicial sequência de contagem.



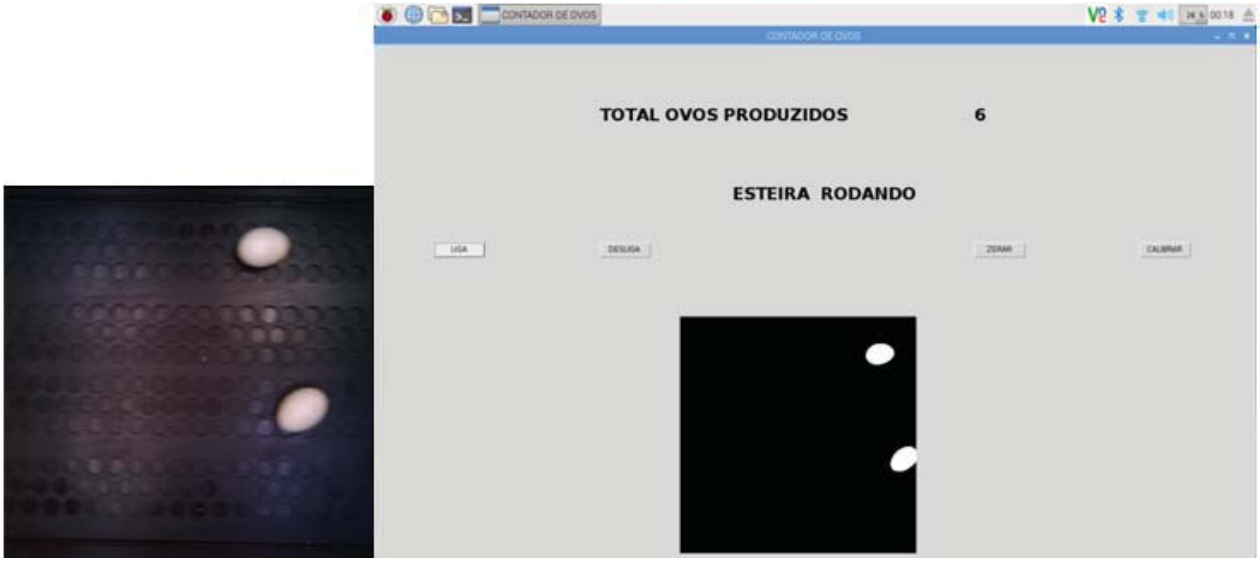
Fonte: Próprio autor.

Figura 55 – 1º contagem.



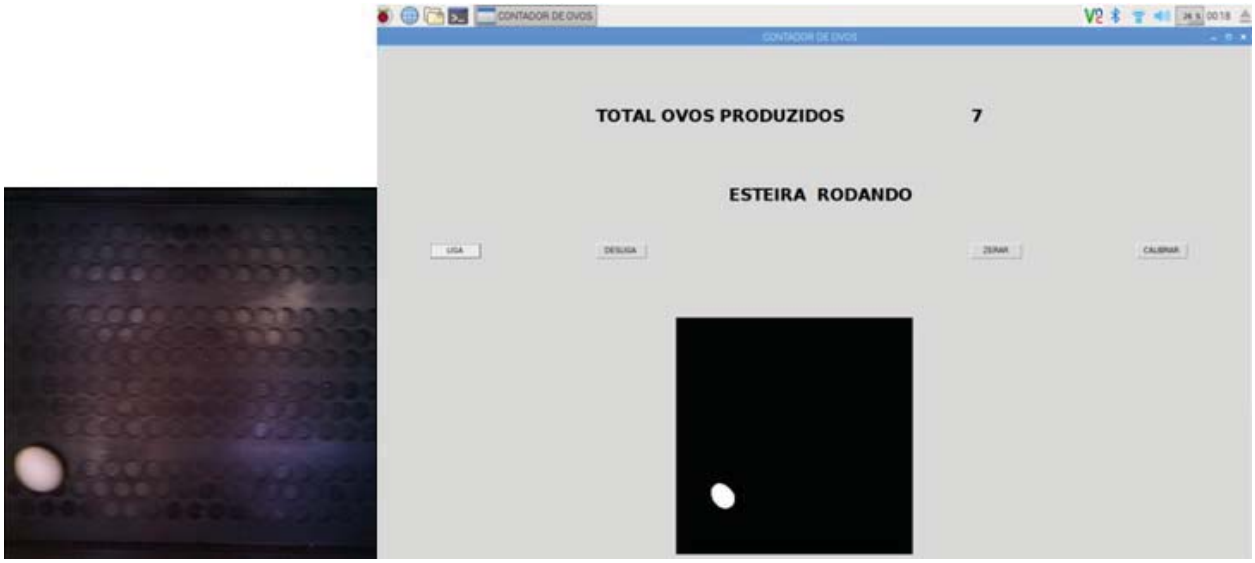
Fonte: Próprio autor.

Figura 56 – 2 º contagem.



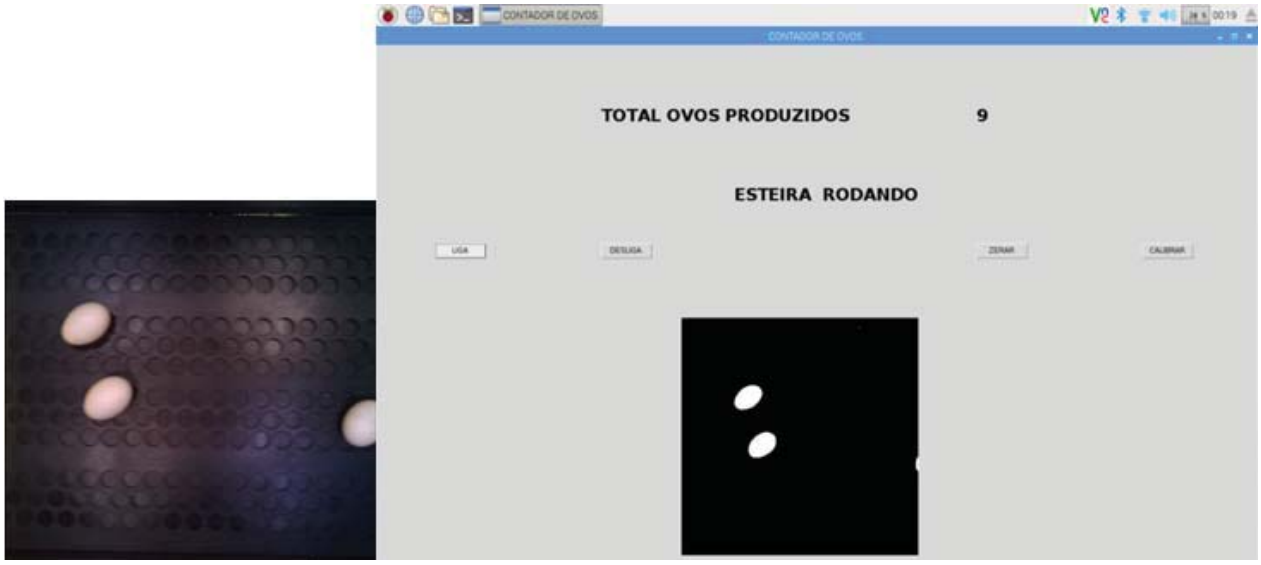
Fonte: Próprio autor.

Figura 57 – 3º contagem.



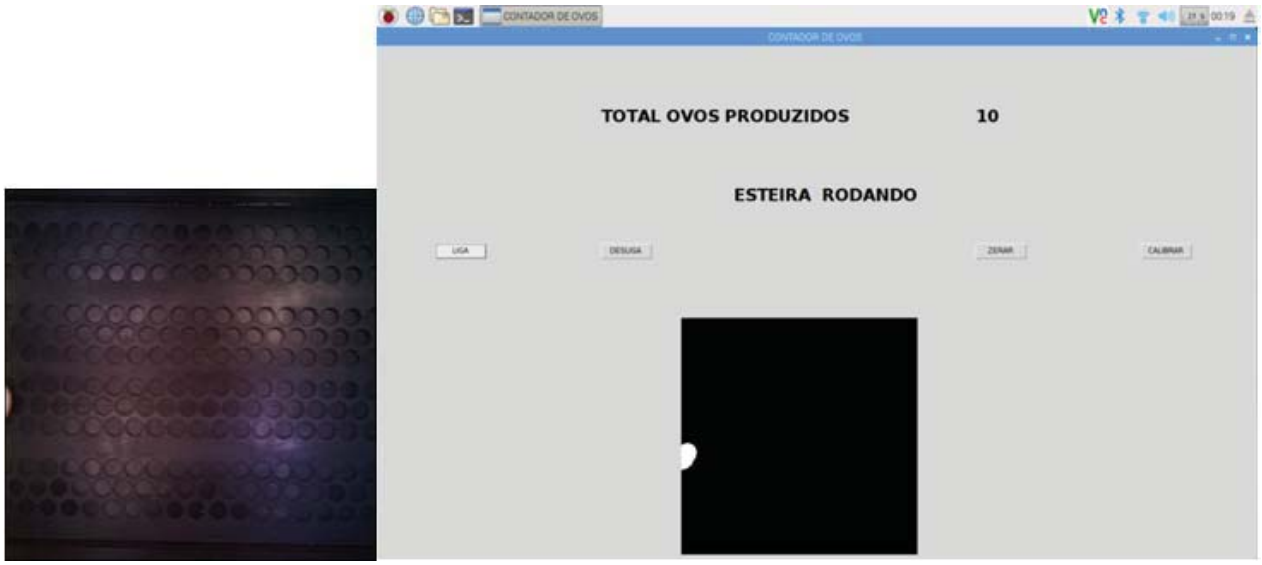
Fonte: Próprio autor.

Figura 58 – 4º contagem.



Fonte: Próprio autor.

Figura 59 – 5º contagem.



Fonte: Próprio autor.

5 CONSIDERAÇÕES FINAIS

Esse projeto consiste em uma grande oportunidade de se familiarizar com sistemas embarcados em *Linux*, como o *Raspberry Pi*, apesar de ter sido utilizadas lógicas pouco complexas ele submete um grande desafio para quem não é usuário desta plataforma. Além de uma oportunidade de familiarizar-se com sistemas baseados em processamento de imagens e de conhecer uma linguagem de programação de "alto nível" como o *Python*.

Em relação ao sensor utilizado no sistema de sincronismo, no geral, obteve um bom resultado, pois mesmo com esta forma de sensoriamento foi possível adquirir a sequência de fotos nos intervalos pré-determinados. Todavia, fica como recomendação para projetos futuros modificar este sensor, substituindo o sensor foto-sensível por um encoder com isso facilitaríamos os ajustes finos, e ganharíamos maior precisão, não correndo o risco de sobrepor ou obtermos lacunas entre as imagens capturadas.

Por fim, o sistema desenvolvido é uma ferramenta muito poderosa e obteve um resultado excelente na aplicação para a qual foi projetada, sendo este resultado um erro acumulado de 0,2 % nas contagens realizadas.

REFERÊNCIAS

BAUMSTARK, L. **Poultry cage having an automatic egg counting mechanism**, 1958. Disponível em: <https://www.google.com/patents/US2827875>

BELUSSO, D.; HESPANHOL, A. N. A evolução da avicultura industrial brasileira e seus efeitos territoriais. **Revista Percurso**, v. 2, n. 1, p. 25–51, 2010.

BOARETTO, N. et al. Inovação em processo e produto : Um estudo de caso no controle da produção em aviários de postura. **XXV Encontro Nac. de Eng. de Produção**, v. -, n. -, p. 4196–4202, 2005.

CAVINATO, M. V. Projecto 01A-Realce de imagens utilizando transformação de intensidades. **Instituto de \informática - Universidade Federal do Rio Grande do Sul**, p. 1–9, 2009.

EAMES, A. et al. **Raspberry PI Projects Book**. v. 1

GONZALEZ, R. C.; WOODS, R. E. **Processamento digital de imagens**. 3. ed.

KHANDELWAL, S. **The Hacker News**. Disponível em: <http://thehackernews.com/2016/02/raspberry-pi-3-microcomputer.html>.

LOKHORST, C.; VOST, H. W. An automatic egg weighing and counting system for detailed analysis and control of egg production. **J. agric. Engng Res.**, v. 57, p. 137–144, 1994.

MANASA, J. et al. Real time object counting using Raspberry pi. **International Journal of Advanced Research in Computer and Communication Engineering**, v. 4, n. 7, p. 540–544, 2015.

MICROCHIP. Disponível em: <http://www.microchip.com/wwwproducts/en/PIC12F675>. Acesso em: 16 jan. 2017.

PIZZOLANTE, C. C. et al. A trajetória tecnológica na avicultura de postura. **Pesquisa & Tecnologia**, v. 8, n. 2, p. 1–6, 2011.

POSTURA, M. DE S. E Q. PARA A. DE. **Manual de Segurança e Qualidade para a Avicultura de Postura**. Qualidade ed. Brasilia: Embrapa Informação Tecnológica, 2004.

RASPBERRY PI FOUNDATION. **Camera Module**. Disponível em: <https://iprototype.nl/docs/raspberry-pi-camera-module-datasheet.pdf>.

ROCHA, J. S. R. et al. Qualidade do ovo de consumo. **Palestra apresentada na 7ª Edição do PUCVET**, v. -, n. -, p. 1–12, 2010.

APÊNDICE A – IMAGEM SISTEMA MECÂNICO DESENVOLVIDO

Figura 60 – Protótipo do sistema.

Fonte: Próprio autor.

APÊNDICE B – CÓDIGO SISTEMA DE SINCRONISMO

```
#include <12f675.h>
#use delay (clock=4000000, int=4000000)
#fuses HS,NOPUT,INTRC_IO,NOMCLR,MCLR
#byte OSCCAL = 0x90
    #BIT CAL5 = OSCCAL.7
    #BIT CAL4 = OSCCAL.6
    #BIT CAL3 = OSCCAL.5
    #BIT CAL2 = OSCCAL.4
    #BIT CAL1 = OSCCAL.3
    #BIT CAL0 = OSCCAL.2
int pulso=0;
int estouro=0;

#int_TIMER1
void  TIMER1_isr(void)
 //a interrupção é utilizada para sinalizar quando a esteira está parada
{
  if(estouro>20){
//estouro é a variavel que indica quantas vezes o TIMER1 chegou no máximo da contagem
    disable_interrupts(INT_TIMER1);//alterei o tempo
    output_Low(PIN_A2);
    enable_interrupts(INT_TIMER1);
    }
  estouro++;
}
void main()
{
  //configura o oscilador interno
  CAL5=0;
  CAL4=0;
  CAL3=1;
  CAL2=1;
```

```
  CAL1=1;
  CAL0=1;
  //1/(4Mhz/4/8), prescaler de 8 - vai incrementar a cada 8us
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
  output_High(PIN_A2);
  // Habilita interrupção do TIMER1
  enable_interrupts(INT_TIMER1);
  //Habilitação global de interrupções
  enable_interrupts(GLOBAL);
  while(TRUE)
  {
    while ( input(PIN_A0) );
      pulso++;
      set_timer1(0);//liga o TIMER1
      output_High(PIN_A2);
     //reset do tempo entre um pulso e outro
      estouro=0;
    while ( !input(PIN_A0) );
    //pulso de sincronismo para captura da imagem
    if(pulso==17 ){
      output_High(PIN_A1);
      delay_ms(1);
      output_Low(PIN_A1);
      pulso=0;
      }
  }
}
```

APÊNDICE C – CÓDIGO SISTEMA ELETRÔNICO DE CONTAGEM DE OVOS

```
from tkinter import *
from tkinter import ttk
from time import sleep
import picamera
import picamera.array
import sys
sys.path.append('/usr/local/lib/python3.4/site-packages')
import cv2
import RPi.GPIO as GPIO                    #Importa a biblioteca das GPIO
import time                                #Importa a biblioteca de tempo

#Configura o modo de definição de pinos como BOARD (contagem de pinos da placa)
GPIO.setmode(GPIO.BOARD)
#Desativa os avisos
GPIO.setwarnings(False)
#Configura o pino 16 da placa (GPIO23) como entrada e com resistor de PULL DOWN
GPIO.setup(16, GPIO.IN,GPIO.PUD_DOWN)
#Configura o pino 18 da placa (GPIO24) como entrada e com resistor de PULL DOWN
GPIO.setup(18, GPIO.IN,GPIO.PUD_DOWN)
#Configura o pino 12 da placa (GPIO18) como saída
GPIO.setup(12, GPIO.OUT)
#Configura o pino 10 da placa (GPIO15) como saída
GPIO.setup(10, GPIO.OUT)

a = [1]
ovos = a
ovos[0] = 0
contadorovos = 0

#PEGA PARÂMETROS DO ARQUIVO TXT
file = open("parametros.txt", "r")
e=file.readline()
```

```
b=file.readline()
c=file.readline()
d=file.readline()
file.close()

#Transforma parâmetros em int e passa para as variáveis do programa
altura=int(b)
largura=int(c)
k=int(d)
areamed=int(e)

i = [1]
atualizaimagem = i
atualizaimagem[0] = 1

GPIO.output(10, 1)

def resetsincronismo():
    #Reset placa sinal de sincronismo
    GPIO.output(12, 1)
    time.sleep(0.001)
    GPIO.output(12, 0)

def contarpixel(img):
        contadorpixel = 0
        visitado = img < 255
        for x in range (0, altura):
                for y in range (0, largura):
                        if not visitado[x, y]:
                                contadorpixel += 1
        contadorovos=(contadorpixel/areamed)
        return contadorovos
```

```
def areascalib(img):
        contadorpixelcal = 0
        visitado = img < 255
        for x in range (0, altura):
                for y in range (0, largura):
                        if not visitado[x, y]:
                                contadorpixelcal += 1
        areaovo=contadorpixelcal/6
        file=open("parametros.txt","w")
        file.write(str (int(areaovo)))
        file.write('\n')
        file.write(str (int(b)))
        file.write('\n')
        file.write(str (int(c)))
        file.write('\n')
        file.write(str (int(d)))
        file.write('\n')
        file.write('AREA MÉDIA DOS OVOS')
        file.write('\n')
        file.write('PIXELS EM X')
        file.write('\n')
        file.write('PIXELS EM Y')
        file.write('\n')
        file.write('PONTO DE CORTE BINARIZAÇÃO')
        file.close()
        return areaovo

def programa():

    if atualizaimagem[0]==0:
        logo = PhotoImage(file="binaria.png")
        ttk.Label(mainframe,image=logo).grid(column=2, row=4, sticky= E)
```

```
            ttk.Label(mainframe, text="ESTEIRA RODANDO",font=('Arial', 20,
'bold')).grid(column=2, row=3, sticky=(N, E))
            atualizaimagem[0]=1
            root.after(50, programa)
            root.mainloop()


        while GPIO.input(16)==0 :

            if GPIO.input(18)==0:#Se a esteira estiver parada habilita zerar e desligar
                ttk.Label(mainframe, text="ESTEIRA PARADA!!!",font=('Arial', 20,
'bold')).grid(column=2, row=3, sticky=(N, E))
                root.after(2000, programa)
                root.mainloop()
        with picamera.array.PiRGBArray(camera) as stream:
           camera.capture(stream, format='bgr')
           captura = stream.array

        img = cv2.cvtColor(captura, cv2.COLOR_BGR2GRAY)
        #transforma pra binário
        img[img > k] = 255
        img[img <= k] = 0

        cv2.imwrite('binaria.png',img)
        ovos[0]=contarpixel(img)+ovos[0]
        totalovos.set(int(ovos[0]+0.5))#0.5 critério de arredondamento
        atualizaimagem[0]=0
        root.after(1, programa)

def zerar(*args):
    try:
        if GPIO.input(18)==0:#Se a esteira estiver parada habilita zerar o contador
            ovos[0] = 0
            totalovos.set(ovos[0])
```

```
            resetsincronismo()#reset sinal de sincronismo
    except ValueError:
        pass


def calibrar():
    try:
        if GPIO.input(18)==0:#Se a esteira estiver parada habilita zerar o contador
            camera.start_preview()
            sleep(20)
            camera.stop_preview()
            with picamera.array.PiRGBArray(camera) as stream:
                camera.capture(stream, format='bgr')
                # At this point the image is available as stream.array
                captura = stream.array
            img = cv2.cvtColor(captura, cv2.COLOR_BGR2GRAY)
            #transforma pra binário
            img[img > k] = 255
            img[img <= k] = 0
            contarpixel(img)
            contador = areascalib(img)
            GPIO.output(10, 0)
            root.destroy()
            exit(0)

    except ValueError:
        pass
```

```
def desliga(*args):
    if GPIO.input(18)==0:#Se a esteira estiver parada habilita desligar o contador
        GPIO.output(10, 0)
        root.destroy()
        exit(0)
resetsincronismo() #reset sinal de sincronismo
with picamera.PiCamera() as camera:
    camera.resolution = (int(altura), int(largura))  #colunas x linhas
    root = Tk()
    root.title("CONTADOR DE OVOS ")
    mainframe = ttk.Frame(root, padding="3 3 12 12")
    mainframe.grid(column=0, row=0, sticky=(N, W, E, S))
    mainframe.columnconfigure(0, weight=4)
    mainframe.rowconfigure(0, weight=4)
    totalovos = IntVar() #cria variável totalovos
    ttk.Label(mainframe, textvariable=totalovos,font=('Arial', 20, 'bold')).grid(column=3,
row=2, sticky=(W, E)) #escreve 0 no totalizador
    ttk.Label(mainframe, text="TOTAL OVOS PRODUZIDOS  ",font=('Arial', 20,
'bold')).grid(column=2, row=2, sticky=N)
    ttk.Button(mainframe, text="LIGA", command=programa).grid(column=1, row=3,
sticky=W)
    ttk.Button(mainframe, text="DESLIGA", command=desliga).grid(column=2, row=3,
sticky=W)
    ttk.Button(mainframe, text="ZERAR", command=zerar).grid(column=3, row=3,
sticky=N)
    ttk.Button(mainframe, text="CALIBRAR", command=calibrar).grid(column=4, row=3,
sticky=W)
    for child in mainframe.winfo_children(): child.grid_configure(padx=93, pady=95)
    logo = PhotoImage(file="inicial.png")
```

```
ttk.Label(mainframe,image=logo).grid(column=2, row=4, sticky= E)
root.bind('<Return>', programa)
root.bind('<Return>', zerar)
root.bind('<Return>', desliga)
root.bind('<Return>', calibrar)
root.mainloop()
```