UNIVERSIDADE DE SÃO PAULO – USP
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

TRABALHO DE CONCLUSÃO DE CURSO

ANDRÉ DE SOUSA GALLO

São Carlos
2015

ANDRÉ DE SOUSA GALLO

SISTEMA AUTOMÁTICO PARA MARCAÇÃO DO CENTRO GEOMÉTRICO DE LENTES PARA ÓCULOS DE SOL

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em Eletrônica

ORIENTADOR: Dra. Liliane Ventura Schiabel

São Carlos
2015

Gallo, André de Sousa
G172s Sistema automático para marcação do centro geométrico de lentes para óculos de Sol / André de Sousa Gallo; orientadora Liliane Ventura Schiabel. São Carlos, 2015.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2015.

1. Óculos de Sol. 2. Centro geométrico. 3. Automático. 4. Marcação. 5. NBR ISO 12312-1. I. Título.

FOLHA DE APROVAÇÃO

Nome: André de Sousa Gallo

Título: "Sistema automático para marcação do centro geométrico de lentes para óculos de sol"

Trabalho de Conclusão de Curso defendido e aprovado em 27/11/2015,

com NOTA 9,0 (nove, zero), **pela Comissão Julgadora:**

Profa. Associada Liliane Ventura Schiabel - (Orientadora - SEL/EESC/USP)

Dr. Mauro Masili - (Pós-Doutorando - SEL/EESC/USP)

Profa. Assistente Doutora Ligia de Oliveira Ruggiero - (UNESP/Campus Bauru)

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Dr. José Carlos de Melo Vieira Júnior

Resumo

Óculos de Sol são acessórios que têm por finalidade proteger os olhos dos raios solares e são principalmente necessários em países tropicais como o Brasil, em que os índices ultravioletas atingem graus extremos. Com a existência de uma variedade de modelos, cores, formatos, é imprescindível a análise dessas lentes em relação à qualidade.

Para a realização dos testes nestas lentes, se faz necessário o conhecimento do seu centro geométrico, que é a região em que a pupila do usuário fica situada ao usar este acessório.

Neste trabalho foi desenvolvido um protótipo capaz de calcular e marcar o centro geométrico da lente de maneira automatizada, com o apoio de um sistema mecânico e métodos de visão computacional.

Lentes solares degradées e não degradées, num total de 32, foram testadas no protótipo, onde as marcações foram feitas automaticamente 4x para teste de repetibilidade. Comparações destas medidas com medidas manuais foram realizadas e o sistema ficou em concordância em aproximadamente 71,5% das medidas para lentes comuns e 82% para lentes dégradées.

O protótipo mostrou-se viável para fazer parte da rotina de marcação das lentes que serão testadas nos projetos do LIO, que necessitam de ter seus centros geométricos marcados.

Palavras-chave:

1.Óculos de Sol. 2. Centro Geométrico. 3. Automático. 4. Marcação. 5. NBR ISO 12312-1

Abstract

Sunglasses are accessories designed to protect the eyes from sunlight and are mainly used in tropical countries such as Brazil, where the UV index reaches extreme degrees. Because of the existence of a variety of designs, colors, shapes, it is essential to analyze these lenses in relation to quality.

For the tests performed in these lenses, it is necessary to know the geometric center, which is the region where the user's pupil is located when using this accessory.

In this work, a prototype able to calculate and mark the geometric center of the lens in an automated way was developed with support of a mechanical system and computer vision methods.

32 gradient and non gradient solar lenses were tested in the prototype, where the central marks have been made automatically 5 times to repeatability test. Comparisons of these measures with manual measurements, double blind, were carried out and the system was in agreement to approximately 71,5% of the measures for gradient solar lenses e 82% for non gradient solar lenses.

The prototype proved to be feasible to be part of a routine to mark central points of lenses that will be tested in the LIO projects.

Key words:

1.Sunglasses. 2.Geometric Center. 3. Automatic. 4. Trace. 5. NBR ISO 12312-1

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

O Sol, apesar de ser indispensável para a existência de vida na Terra, é também um agente prejudicial à saúde se não tomados os devidos cuidados. O uso de protetores solares é essencial para pessoas que se expõem demasiadamente ao sol. Porém o único modo de proteger os olhos contra as radiações ultravioletas é utilizando óculos de sol.

Com uma variedade de modelos, fabricantes, materiais e cores, muitas vezes o consumidor adquire seus óculos sem uma maior preocupação com a qualidade e resistência das lentes. Além da proteção contra raios UVA/UVB, os óculos também devem ser resistentes. Se algo atingir os olhos do usuário, os óculos não devem se quebrar causando traumas ou até perda da visão.

Por esses motivos o Laboratório de Instrumentação Oftálmica – EESC/USP (LIO) vem criando mecanismos que permitam a análise automática de lentes e armações de óculos. Essas análises permitem que óculos sejam certificados de acordo com a norma da ABNT NBR ISO 12312-1 [1].

Um dos trabalhos do LIO [2], onde os autores estão desenvolvendo um protótipo automático para estudar os efeitos do sol nas lentes dos óculos. No trabalho, 100 lentes serão expostas ao sol durante um ano. Com o protótipo as lentes serão expostas durante o dia, acompanhando a direção do sol e serão guardadas durante a noite ou em casos que possam interferir no estudo como chuvas e queimadas.

Outro projeto aplicado do LIO, realizado por Magri [3] foi o desenvolvimento de um sistema para estudar a inflamabilidade de óculos de sol. O projeto consistiu em construir um forno elétrico com temperatura ajustada por um termopar e um controlador PID. O objetivo deste trabalho é garantir a segurança dos usuários de óculos solares verificando a qualidade e certificando que os mesmo não inflamem.

No trabalho referente a teste de impacto [4], um dos parâmetros a ser encontrado é o centro geométrico da lente, o qual é tomado como ponto de referência para realização de ensaios. O centro geométrico, de acordo com a NBR ISO 12312-1 [1], pode ser tomado como ponto de referência caso o mesmo seja desconhecido. Atualmente, a medição do ponto médio geométrico é uma tarefa manual que pode embutir tempo e erros ao processo de certificação. Este trabalho vem complementando o conjunto de testes já feitos no Laboratório LIO para testes em óculos solares. A seção a seguir explica um pouco sobre a norma regulamentadora.

1.1 Norma ABNT NBR ISO12312-1

Substituindo a norma NBR 15111, uma tradução da norma europeia EM 1836:2001, a norma NBR ISO 12312-1 tem como objetivo garantir a qualidade de óculos solares e relacionados para consumidores e comerciantes. Para serem seguros para os usuários, os óculos devem passar por todos os testes propostos por essa norma, garantindo ao consumidor um produto de qualidade. A norma tem como objetivo testar várias propriedades da lente como: refração, transmitância, difusão da luz, inflamabilidade, resistência da armação entre outros.

No âmbito deste trabalho, se faz necessário o conhecimento de pontos de referência de cada lente. A NBR ISO 12312-1 [1] não possui uma definição de ponto de referência, o mesmo é fornecido através do fabricante de cada lente. Quando não se conhece o ponto de referência, utiliza-se o centro geométrico ou ponto médio geométrico da lente para a realização de ensaios. Assim, como as lentes possuem dimensões e formatos variados, se faz necessário o cálculo deste centro para cada tipo de lente fabricada. Alguns dos requisitos que fazem utilização deste ponto estão descritos a seguir:

- **Material do filtro e qualidade da superfície**: o centro deve ser calculado para aferir a qualidade do material. A norma NBR ISO 12312-1 da ABNT [1] determina que num raio de 30 mm em volta do ponto de referência, as lentes não devem apresentar nenhum defeito de material ou de fabricação que possa alterar ou prejudicar a visão como, por exemplo, bolhas, arranhões, inclusões, manchas, etc.

- **Uniformidade de transmitância luminosa**: Homogeneidade da transmitância luminosa é um dos requisitos gerais para lentes de óculos de sol que determina que a diferença relativa de transmitância luminosa entre dois pontos quaisquer, dentro de um círculo de diâmetro 40 mm, centrado no ponto de referência, não deve ser maior que 10%. No caso de lentes dégradé, o requisito de uniformidade da transmitância é válido para um círculo de 10 mm de raio centrado no ponto de referência da lente.

- **Ângulo de espalhamento**: Quanto aos requisitos para utilização em condução de veículos, no ponto de referência, o espalhamento angular da lente não pode exceder 3%.

1.2 Objetivos

Visto a importância da medição do centro geométrico da lente para o controle de qualidade das lentes de óculos solares, este trabalho visa o desenvolvimento e confecção de um protótipo para o cálculo e marcação automática do centro geométrico de lentes de óculos solares. Por sua vez, o protótipo tem como função facilitar e agilizar o processo de marcação do centro geométrico das lentes, que hoje é feito à mão com a ajuda de instrumentos de medição como o paquímetro. Além disso, o equipamento deve ser pequeno e por esse motivo a construção e escolha dos materiais deve ser cuidadosa a ponto de contemplar este objetivo.

No Laboratório de Instrumentação Oftálmica da Escola de Engenharia de São Carlos são realizados inúmeros testes e ensaios com lentes de óculos solares. O protótipo que será contemplado neste relatório fará parte de um conjunto de equipamentos que terá como objetivo avaliar e certificar lentes de óculos de sol. O LIO tem como objetivo a certificação via INMETRO para ser capaz de realizar os testes propostos pela norma de análises da qualidade dos óculos solares

A certificação dos óculos garante ao consumidor qualidade, que não ponha em risco sua saúde ocular, causando traumas e doenças oftalmológicas. No Brasil, um país onde clima tropical é predominante e a maior parte da população tem pelo menos um par de óculos de sol, é importante garantir que óculos de baixa qualidade não cheguem até o consumidor, fiscalizando e analisando, não só a lente, mas também a armação dos óculos.

O desenvolvimento deste equipamento consiste na junção de métodos de processamento de imagem, com ajuda da biblioteca *Opencv* para *Python* [5], e a utilização de um microcomputador modelo *Raspberry Pi* , uma *webcam*, dois motores e dois botões. Foi construído um aparato para apoiar e marcar o centro da lente. Todos os desenhos e funções serão descritos durante o decorrer do texto.

Para as lentes dégradées, serão também encontrados os pontos 10 mm acima e abaixo do centro da lente, para o posterior teste de transmitância luminosa, de acordo com o requerido na norma NBR ISO 12312-1 [1].

2. MATERIAIS E MÉTODOS

Este trabalho consiste na junção de duas partes: software e hardware, aqui denominados métodos e materiais, respectivamente, os dois segmentos que se comunicam durante o processo para a determinação e marcação do centro das lentes. Este capítulo tem como objetivo descrever todos os métodos e materiais necessários à elaboração deste protótipo, assim como, um embasamento para o uso de cada método e material utilizado.

2.1 Métodos

Os métodos utilizados para a construção do equipamento devem analisar imagens provenientes de uma *webcam* para encontrar o centro da lente. Duas áreas são envolvidas no processo: o processamento da imagem e a retirada de uma informação, ou seja, uso de visão computacional.

Pode-se dizer que o projeto utiliza processamento e visão computacional, pois, quando há uso de processamento de imagem, a entrada do sistema é uma figura e a saída é a mesma, apenas com modificações. Como exemplos de métodos de processamento pode-se citar remoção de ruídos e "borramento", detecção de borda, aplicação de filtros de média, mediana, operações morfológicas, entre outros.

Já na visão computacional, a entrada é uma imagem e a saída uma informação. Um exemplo é o reconhecimento de padrões, onde há reconhecimento de objetos, dando um rótulo a cada um deles. Uma grande aliada na utilização de métodos de processamento e visão computacional é a biblioteca *OpenCV* explicada na Seção 3.1.1.

Os métodos necessários para a identificação da lente e marcação do centro e posições 10 mm acima e abaixo do centro são: binarização (Seção 3.1.2), análise de componentes conectados (Seção 3.1.3), análise de componentes principais (Seção 3.1.4) e cálculo dos momentos invariantes de Hu (Seção 3.1.5). Cada método e respectivas explicações de sua necessidade serão descritos nas seções a seguir.

2.1.1.*OpenCV*

Para o desenvolvimento do programa de identificação da lente foi utilizada uma biblioteca de visão computacional com código aberto chamada *"OpenCV".* Essa biblioteca é escrita em linguagem C e C++ e está disponível para os principais sistemas operacionais: Windows, Linux e Mac [5]. No trabalho, a biblioteca será utilizada juntamente com a linguagem de programação Python.

O *OpenCV* é uma ferramenta de aplicação simples e eficiente para se trabalhar com imagem. Essa biblioteca possui funções que abrangem tanto na área de visão computacional quanto na área de processamento de imagem, sendo assim foi essencial no desenvolvimento do software do protótipo.

Marengoni e Stringhini [5] descrevem em seu artigo uma introdução e um tutorial sobre visão computacional e a biblioteca *OpenCV*. São apresentados métodos e exemplos de segmentação da imagem e reconhecimento de padrões muito utilizados na área.

2.1.2. Binarização

A imagem obtida da *webcam* é uma imagem em cores, e não há separação do que é fundo e do que é lente. Para esse problema foi utilizada a binarização que será explicada nesta seção.

O nome "binarização" remete ao conceito de duas unidades (bi) ou composição por dois elementos. O processo de binarização da imagem consiste em transformar todos os pixels em apenas dois níveis de cinza, o branco e o preto. O conceito de limiarização remete à forma na qual são selecionados os pixels que serão pretos e os pixels que serão brancos. Através de métodos é escolhido um limiar, ou seja, os pixels com intensidade maior que o limiar se tornarão brancos e os pixels com intensidade menor que o limiar se tornarão pretos. A Figura 1 mostra um exemplo da utilização do método de binarização.

Existem vários métodos para a escolha deste limiar como o método do vale [6], Gonzalez e Woods [6] e o bimodal de Otsu [7]. Todos esses métodos foram estudados para a análise de qual seria o melhor a ser aplicado no trabalho. Estes serão expostos nesta seção, mostrando seu funcionamento.

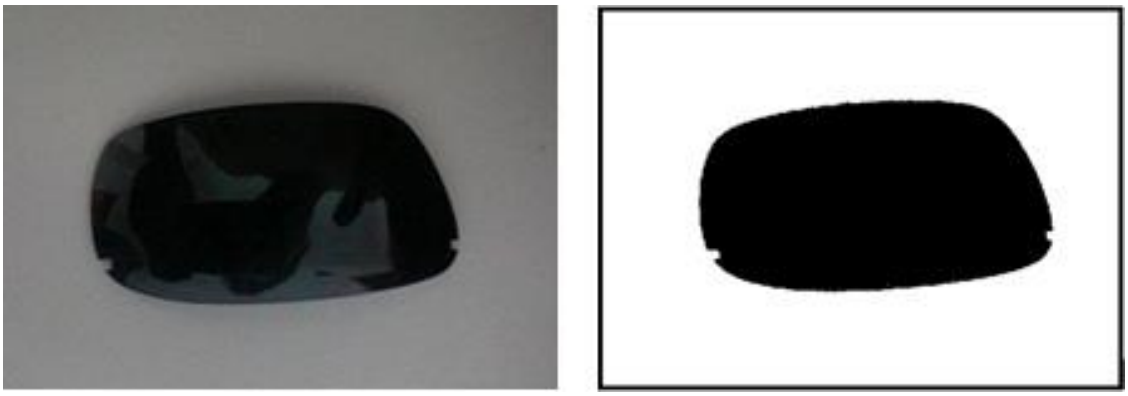


Figura 1 - Binarização da imagem da lente

2.1.2.1 Método do Vale

O primeiro encontrado na literatura é o método do vale [6]. Para explicar melhor esta técnica de binarização é necessário o conhecimento sobre histogramas.

O histograma de uma imagem é um gráfico que no eixo vertical se encontra a quantidade de cada pixel e no eixo horizontal se encontra a intensidade do pixel na escala de cinza, ou seja, o histograma de uma imagem mostra a quantidade de pixels para cada nível de cinza em um determinado intervalo. A Figura 2 mostra um exemplo de uma imagem em escala de cinzas e o seu histograma, respectivamente.

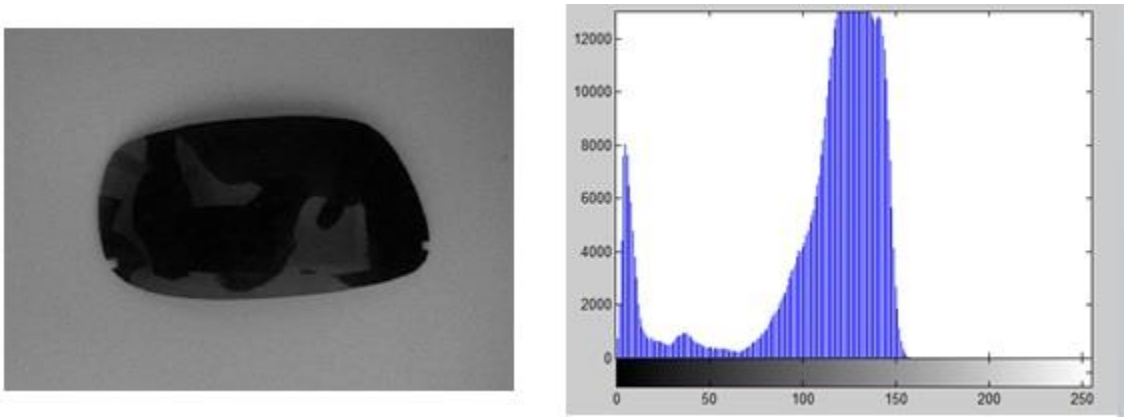


Figura 2 - Imagem de uma lente (esquerda) e respectivo histograma (direita)

O Método do Vale consiste em analisar visualmente o histograma da imagem e encontrar um limiar que se adeque ao resultado desejado. Este método se baseia na ideia de que na maioria das vezes

uma boa escolha de limiar se encontra no vale mais próximo do centro do histograma da imagem [6]. Neste exemplo da Figura 2 o limiar escolhido seria entre 60 e 70 na escala de cinza de 0 a 255.

O método tem a desvantagem de ser uma análise visual para a escolha do melhor limiar. Não é viável, com relação ao custo computacional, aplicar uma rotina para encontrar o "vale", no caso onde a primeira derivada é igual a zero, pois podem existir vários "vales" dentro de um histograma e isso deixaria o código "pesado". Por esse motivo, quando se usa este método, cabe ao usuário identificar o melhor ponto para aplicar a limiarização em cada projeto. Apesar de funcionar bem em algumas aplicações devido á escolha manual do limiar, esta técnica não pode ser aplicada no trabalho, visto que a única interação do usuário com o programa é o aperto de um botão.

2.1.2.2 Gonzalez e Woods

O método de Gonzalez e Woods [6] basicamente utiliza um código que, a partir de um limiar inicial, divide a imagem em dois grupos de pixels, calcula a intensidade média de cada uma das regiões e recalcula o limiar a partir da Equação (1). Nesta equação, T é o limiar e $\overline{\mu_1}$ e $\overline{\mu_2}$ são as médias das intensidades das regiões G_1 e G_2 respectivamente [6].

$$\boldsymbol{T = \frac{1}{2}[\overline{\mu_1}(G_1) + \overline{\mu_2}(G_2)]} \tag{1}$$

Após calcular este novo limiar o programa repete os passos e atualiza o limiar até que a diferença entre os limiares encontrados, ou seja, o ΔT, seja menor que um valor pré-definido pelo programador de acordo com a finalidade do código.

De acordo com análises feitas neste trabalho, o método de Gonzalez e Woods funciona bem. No entanto, a facilidade de encontrar o próximo método na biblioteca *OpenCV* fez com que fosse mais vantajoso e automático devido a não necessidade de atribuição de valor inicial ao limiar.

2.1.2.3 Método de Otsu

O método de Otsu [7] olha para o histograma da imagem como uma função de probabilidade, onde $P_r(r_q)$ é a probabilidade de um pixels ter a intensidade r_q, n_q é o número de pixels com intensidade r_q, n é o numero total de pixels da imagem e L é o maior nível de intensidade da imagem.

$$\boldsymbol{p_r(r_q) = \frac{n_q}{n} \qquad q = 1, 2, 3, \ldots, L - 1} \tag{2}$$

Esse método consiste em tomar o limiar inicial como 1 e dividir a imagem em duas classes: uma de pixels com intensidades abaixo do limiar e outra de pixels com intensidade acima do limiar. A partir destas duas classes são calculadas as probabilidades de um pixel pertencer à classe 1, $P_1(T)$ (Equação 3), a probabilidade de um pixel pertencer a classe 2, $P_2(T)$ (Equação 4), a intensidade média dos pixels da classe 1, $m_1(T)$ (Equação 5), a intensidade média dos pixels da classe 2, $m_2(T)$ (Equação 6) e a intensidade média global, ou seja, de todos os pixels, m_g (Equação 7). Calculados esses dados, encontra-se a variância entre as classes em relação à variância global a partir da Equação 8.

$$P_1(T) = \sum_{i=0}^{T} p_i \tag{3}$$

$$P_2(T) = \sum_{i=T+1}^{L-1} p_i = 1 - P_1(T) \tag{4}$$

$$m_1(T) = \frac{1}{P_1(T)} \sum_{i=0}^{T} i p_i \tag{5}$$

$$m_2(T) = \frac{1}{P_2(T)} \sum_{i=T+1}^{L-1} i p_i \tag{6}$$

$$m_g = \sum_{i=0}^{L-1} i p_i \tag{7}$$

$$\sigma^2(T) = P_1(m_1 - m_g)^2 + P_2(m_2 - m_g)^2 \tag{8}$$

Depois de calculada a variância para os dois conjuntos limiarizados a partir do valor 1, o algoritmo incrementa o valor do limiar e calcula as variâncias novamente. O algoritmo termina quando todos os limiares possíveis são utilizados. Desse modo o código escolhe o valor de limiar que maximiza a variância extraclasse, ou seja, maximiza a variância para o grupo de pixels 1 e 0 [6].

Este método foi escolhido para o trabalho devido à sua popularidade e capaz de funcionar bem em diferentes condições de iluminação. Além disso, uma implementação do método é encontrada na biblioteca *OpenCV*, pelo uso da função *threshold* com o parâmetro CV_THRESH_OTSU. No trabalho,

foi utilizada a binarização invertida, ou seja, ao final da binarização, a lente fica com os pixels em 1 e o fundo com pixels em 0.

2.1.3. Análise de Componentes Conectados

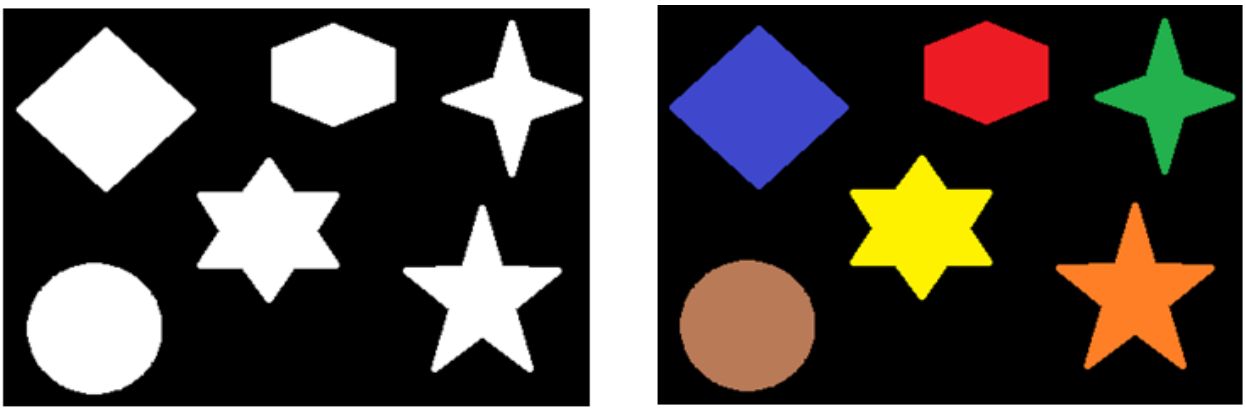
Muitas vezes, elementos externos à lente são binarizados e identificados erroneamente como objeto. Como esses elementos não são conectados fisicamente, uma maneira de "limpar" a imagem, eliminando essas impurezas na imagem binária é aplicar o método de componentes conectados.

O método funciona em imagens binárias. Supondo que pixels com valor 1 são objetos e 0 apenas o fundo da imagem, conjuntos de pixels brancos desconexos podem ser rotulados com números para os representar. Por exemplo, em uma imagem binária existem dois grupos desconexos representando uma pessoa e um carro. Para analisar cada objeto separadamente, é possível atribuir a cada pixel pertencente ao carro os valores de 1 e aos pixels da pessoa o rótulo 2.

De maneira geral, o algoritmo de componentes conectados tem como objetivo encontrar e rotular cada componente de uma imagem binária. Com base na ideia de grafos conexos, pixels que são vizinhos na imagem tem relação a pontos que possuem arestas nos grafos. Além disso, se eles não possuem o mesmo valor binário (mesmo sendo vizinhos), a aresta entre eles é excluída. Para imagens, pode se considerar uma vizinhança de 4 ou 8 vizinhos para o algoritmo, como for desejado.

O algoritmo funciona da seguinte forma: se um vizinho *q* de *p*, onde *p* é um pixel branco, é um pixel branco e rotulado, atribui-se o mesmo rótulo de *q* à *p*. Do contrário, se todos os vizinhos de *p* são pretos, atribui um novo rótulo ao pixel. Ainda, se mais de um vizinho é branco e rotulado, atribui-se ao novo pixel um dos dois rótulos e estes são marcados como equivalentes (atualizando-os posteriormente para o mesmo valor).

A biblioteca "scipy" do Python possui uma função *label* que retorna a imagem com os rótulos. No trabalho, é pressuposto que a lente é o maior objeto da imagem. Portanto, é analisado o rótulo com maior número de pixels associados. A todos os pixels com esse rótulo é atribuído o valor 1 (lente) e ao restante, o valor 0. Ao final do processo, uma nova imagem binária com apenas a lente é obtida.



(a) Imagem Binarizada (b) Componentes identificados

Figura 3 - Exemplo de análise de componentes conectados

2.1.4 Análise de Componentes Principais - *PCA*

Imitar a visão e o sistema de reconhecimento de imagens do ser humano é o objetivo principal dos estudiosos na área de visão computacional. Um dos problemas a serem resolvidos é a questão de rotação, escala e posição dos objetos. O ser humano é capaz de identificar um objeto conhecido, como uma caneta, estando ela em qualquer posição, escala ou orientação. Porém uma máquina pode ter dificuldades para reconhecê-la caso os parâmetros estejam modificados, fora do que ela foi treinada.

O PCA, do inglês, *Principal Component Analysis*, ou seja, análise dos componentes principais, é um algoritmo que pode ser utilizado para alteração de escala, orientação e posição de imagens [8].

Os algoritmos de PCA funcionam da seguinte forma: primeiramente transformam a imagem em um conjunto de dados e a partir deles se calcula o vetor médio. Este vetor médio serve para transladar os dados para a origem, subtraindo o conjunto de dados do vetor médio. Com isso calcula-se a matriz de covariância a partir da Equação 9, onde $\bar{x}$ e $\bar{y}$ são as coordenadas do vetor médio.

$$cov(x,y) = \frac{\sum_{i=1}^{n}[(x_i-\bar{x})(y_i-\bar{y})]}{n} \quad (9)$$

A partir da matriz de covariância calcula-se os seus autovalores e autovetores e constrói-se a matriz da Transformada de Hotelling cujas linhas são formadas pelos autovetores da matriz de covariância arranjados de forma que a primeira linha da matriz seja o autovetor correspondente ao maior autovalor e assim sucessivamente até que a última linha seja o autovetor correspondente ao menor

autovalor. Desta forma o primeiro elemento da matriz de Hotelling corresponde à componente principal do conjunto de dados utilizado, ou seja, corresponde ao eixo principal da imagem, ou o eixo com maior quantidade de informação (maior variância de dados) [8]

Inicialmente, houve uma preocupação com as lentes dégradé e a marcação dos pontos adicionais em relação ao eixo principal das lentes e sua rotação, por exemplo, caso a lente fosse posicionada de forma errada. O PCA foi implementado e testado com sucesso em algumas lentes. No entanto, como existe uma variedade de modelos, em algumas lentes, a componente principal, ou seja, a de maior variância dos dados, não era compatível com a rotação real da lente causando erros na marcação. Foi decidido no final do projeto a não utilização do PCA e a suposição de que a lente será colocada na base do protótipo alinhada com a posição da *webcam*, ou seja, o posicionamento da lente é de responsabilidade do usuário o qual deve estar atento neste momento para evitar erros na marcação.

Como o ponto de referência necessário é o centro geométrico da lente, o seu correto posicionamento é indispensável para uma boa medição, tendo em vista que o centro geométrico dos objetos varia com a sua rotação. A norma NBR ISO 12312-1 [1] não determina uma forma correta de encontrar o centro geométrico das lentes, portanto fica subentendido que a lente deve ser posicionada da forma na qual ela é utilizada e posicionada no rosto do usuário.

2.1.5. Momentos Invariantes de Hu

Momentos de uma imagem são características espaciais retiradas através da análise das intensidades dos seus pixels. Alguns destes momentos são invariantes em relação à translação, rotação e à mudança de escala, os quais são chamados de Momentos Invariantes de Hu. Através destes momentos invariantes é possível calcular características como área e o centro de massa da imagem.

A Equação 10 mostra como são calculados os Momentos de uma imagem. Onde m é o Momento, p e q são as ordens de x e y, respectivamente, n é o número total de pixels da imagem e I é o valor do pixel.

$$m_{p\,q} = \sum_{i=1}^{n} I(x,y)x^{p}y^{q} \tag{10}$$

No projeto foram usados Momentos de ordem 0 e de ordem 1, ou seja, m_{00}, m_{01}, m_{10}. Com esses momentos é possível calcular o centro de massa da imagem conforme as equações 11 e 12. Onde *x* e *y* são as coordenadas do centro de massa da imagem [9].

$$x = \frac{m_{10}}{m_{00}} \tag{11}$$

$$y = \frac{m_{01}}{m_{00}} \tag{12}$$

O último passo dos métodos é calcular o ponto central da lente. O método de utilizar os Momentos Invariantes de Hu foi implementado e testado dando resultados satisfatórios. Porém, como foi dito acima, com esse método calcula-se o centro de massa do objeto e segundo a norma da ABNT NBR ISO 12312-1 [1] o ponto de referência utilizado para realizar ensaios com as lentes é o centro geométrico. Portanto o método serviu apenas para testes e calibração do protótipo.

2.1.6 Centro geométrico a partir dos pontos extremos

O centro geométrico dos objetos é o ponto médio entre os extremos da imagem, ou seja, é o ponto médio entre os extremos superior e inferior e extremos à direita e à esquerda da imagem.

Para calcular as extremidades da imagem foi utilizada a função, da biblioteca *OpenCV*, *"cv2.findContours"*, que encontra o contorno da imagem e a função *"cv2.boundingRect"* que retorna a largura, altura e os extremos superior e à esquerda da imagem. A partir destes valores calcula-se o centro da imagem utilizando as equações 13 e 14, onde x e y são as coordenadas do centro geométrico da imagem, w e h são a largura e a altura da imagem, respectivamente e p e q são a coordenada x do ponto extremo a esquerda da imagem e a coordenada y do ponto extremo superior da imagem, respectivamente.

$$x = \frac{2p + w}{2} \tag{13}$$

$$y = \frac{2q + h}{2} \tag{14}$$

A partir desses pontos x e *y*, cálculos são feitos para que os motores atinjam esse ponto no centro da imagem. Para isso, um aparato de hardware é necessário. Estes serão explicados na próxima seção.

2.2. Materiais

2.2.1. *Raspberry Pi*

O primeiro e principal componente escolhido foi o microprocessador a ser utilizado. Como foi dito no começo deste documento, uma das principais características deste projeto é a mobilidade do protótipo. Para que isso fosse possível era necessária a utilização de um processador embarcado no protótipo. Utilizar um computador, de tamanho regular que já possui um processador embutido, não seria viável já que ficaria pesado e grande. Porém, a alternativa escolhida foi justamente um microcomputador denominado de *Raspberry Pi*. Este computador foi escolhido por ele possuir o tamanho aproximado de um cartão de crédito convencional e por poder ser alimentado por uma fonte simples de 5 volts por 2 amperes. Além destes benefícios ele possui um processador BCM2835, baseado em ARM, de 700 MHz, ou seja, mais que suficiente para o projeto. A Figura 4 mostra a *Raspberry Pi B*, que foi o modelo adquirido para o projeto [11].

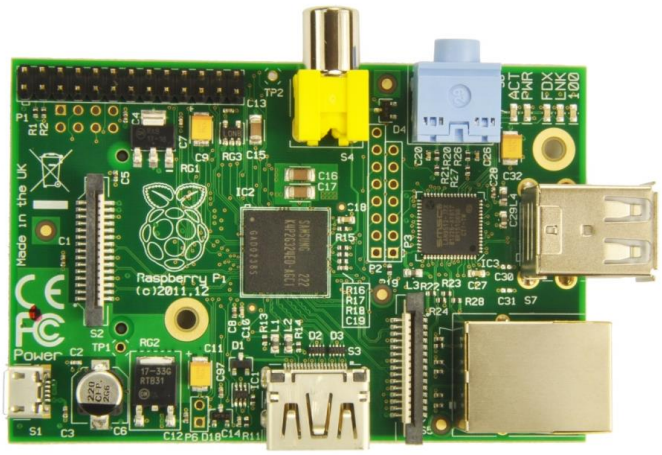


Figura 4 - Raspberry Pi modelo B [1]

A *Raspberry Pi* possui uma porta para cartão de memória SD, no qual ficam gravados todos os dados do sistema da *Raspberry Pi*, ou seja, este cartão de memória pode ser comparado ao HD de um computador convencional. Este modelo em específico possui que uma saída HDMI, duas portas USB, uma saída de áudio, uma saída de vídeo com conector RCA e um conector *Ethernet RJ45* utilizado para conectar a redes com fios. Além disso, ela possui 26 pinos GPIO (Figura 5), os quais serão utilizados para comunicação com os motores, botões e outros periféricos [11]. Estes 26 pinos podem ser utilizados tanto para entrada quanto para saída de sinal, por exemplo, no caso dos motores alguns pinos são utilizados

[1] Figura disponível em: elinux.org/RPi_Hardware. Acesso em Outubro de 2015.

como saída de sinal que são mandados para os motores, ou, no caso dos botões, os pinos são utilizados para reconhecer se há ou não tensão, ou seja, são utilizados como entrada se sinal A *webcam* por sua vez será conectada a uma das portas USB.

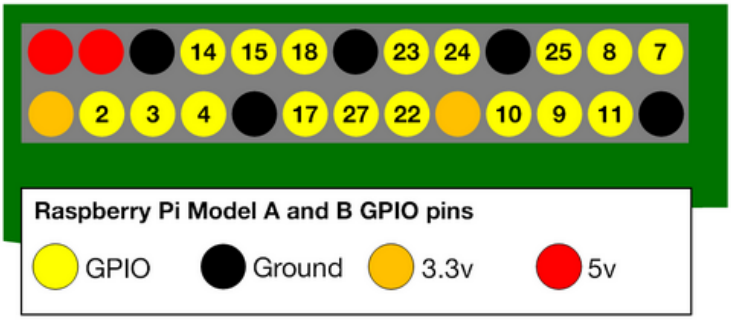


Figura 5 - Pinos GPIO Raspberry Pi modelo B[2]

2.2.2 Conjunto motor

Para a parte móvel do protótipo serão utilizados dois motores, um para o eixo x e outro para o eixo y, assim o protótipo pode varrer e encontrar o centro da lente em qualquer ponto do plano. Como é necessário precisão em seus movimentos, foram escolhidos motores de passo. O modelo dos motores de passo adquiridos foi o 28BYJ-48 em conjunto com o *driver board* ULN2003. Este conjunto é geralmente utilizado para em projetos com microcontroladores *Arduino*, porém com algumas adaptações ele pode ser utilizado com facilidade com a Raspberry Pi. Este motor possui alimentação de 5 volts e um passo de 5,625°/64, o que são adequadas para o projeto [12].

O *driver board* utiliza o circuito integrado ULN2003 o qual é o responsável por fornecer a corrente necessária para cada enrolamento do motor para ele gerar torque. O ULN2003 consiste em uma matriz de sete pares de transistores do tipo NPN que geram uma alta tensão nas saídas [13]. A Figura 6 mostra o esquemático de cada par de transistores, ou seja, um dos sete pares de entrada e saída do circuito integrado.

[2] Figura disponível em: https://www.raspberrypi.org/documentation/usage/gpio/. Acesso em Outubro de 2015

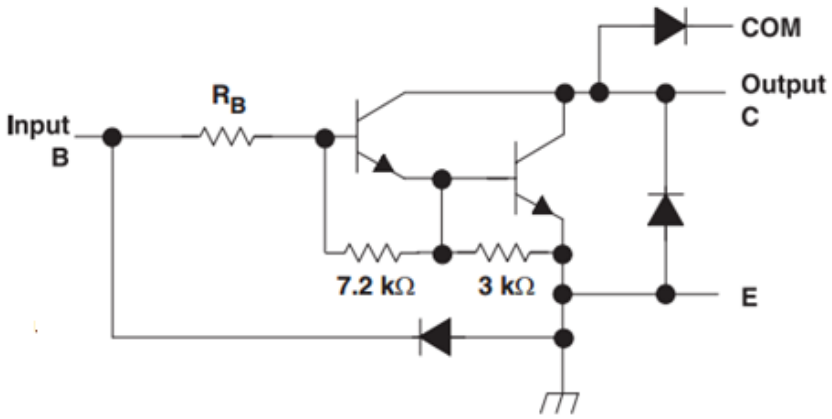


Figura 6 – Esquemático circuito integrado ULN2003 (ULN2003 Datasheet)[3]

A Figura 7 mostra o diagrama lógico e os pinos de entrada e saída do circuito integrado.

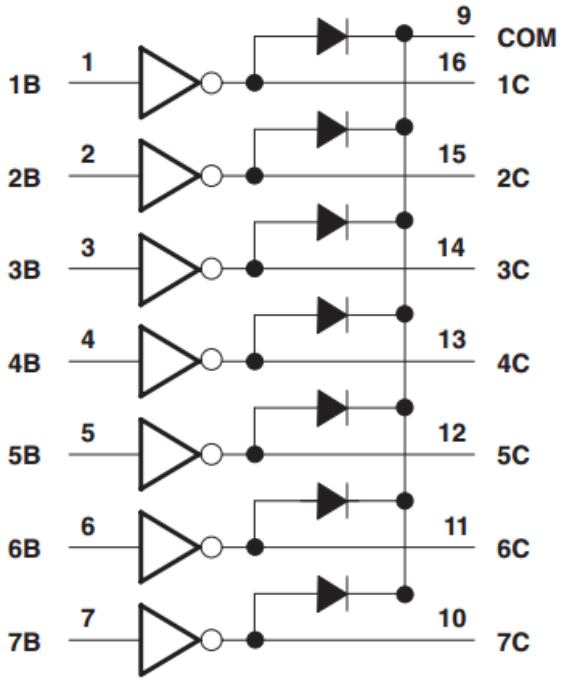
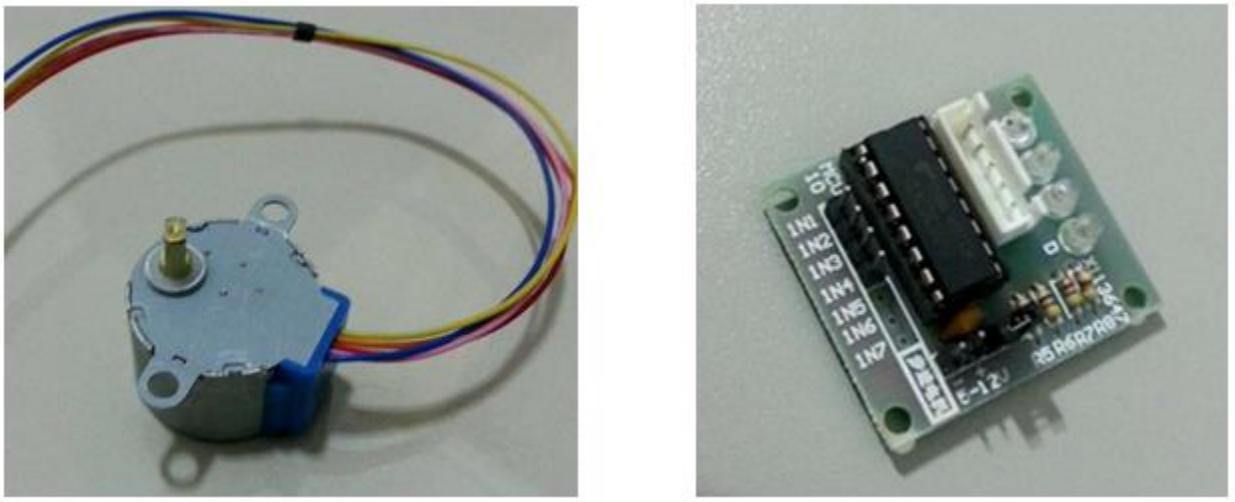


Figura 7 – Diagrama lógico circuito integrado ULN2003(Datasheet ULN2003)[4]

[3] Figura disponível em: www.geeetech.com/Documents/ULN2003%20datasheet.pdf Acesso em Outubro de 2015
[4] Figura disponível em: www.geeetech.com/Documents/ULN2003%20datasheet.pdf Acesso em Outubro de 2015

A Figura 8 mostra o modelo do motor (a) e o modelo da *driver board* (b) utilizados no protótipo.



(a) Motor de passo 28BYJ-48 (b) *driver board* ULN2003

Figura 8 - Conjunto motor

2.2.3 Botões

Assim como todo equipamento eletrônico, para interface com o usuário e ativação do protótipo foram necessários botões. Pela facilidade de uso e dimensões reduzidas foram escolhidos botões do tipo *"push-button",* que são botões que retornam ao seu estado de repouso, ou seja, "circuito aberto", após ser solto pelo usuário. Estes botões são muito utilizados em projetos com microprocessadores para dar comandos para o código. Ao pressioná-lo é possível enviar comando de "nível baixo" ou de "nível alto", dependendo da aplicação e assim, através de um comando condicional, realizar funções.

A Figura 9 mostra em detalhe o botão utilizado para interface com o equipamento.

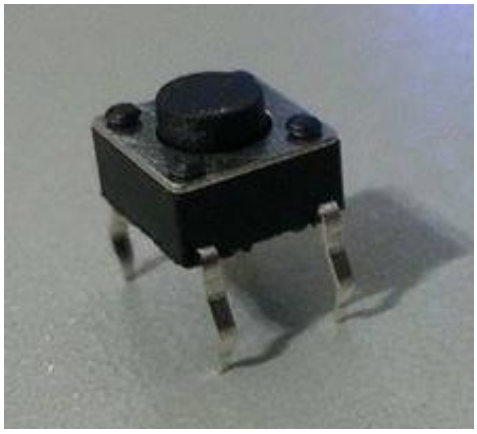


Figura 9 - Botão tipo “push-button”

3. O PROJETO

3.1 Desenhos

O primeiro passo no projeto consistiu em criar uma estrutura a qual poderia desempenhar as funções designadas para o protótipo. Uma das principais características indispensáveis do protótipo é a mobilidade. Por essa razão a estrutura a ser desenvolvida não poderia ser grande e nem pesada. A partir destes conceitos iniciais foram decididos os materiais a serem utilizados e o desenho da estrutura principal.

Os desenhos foram criados em ambiente *SolidWorks*, um software de desenho em duas e três dimensões e simulação de movimentos. A Figura 10 mostra uma simulação em três dimensões de como o protótipo deverá ficar depois de finalizada a sua confecção. O nome de referência de cada motor e respectivos eixos são identificados na figura. As setas indicam o deslocamento de cada eixo. A seguir, cada peça será detalhada com suas dimensões e funcionalidades.

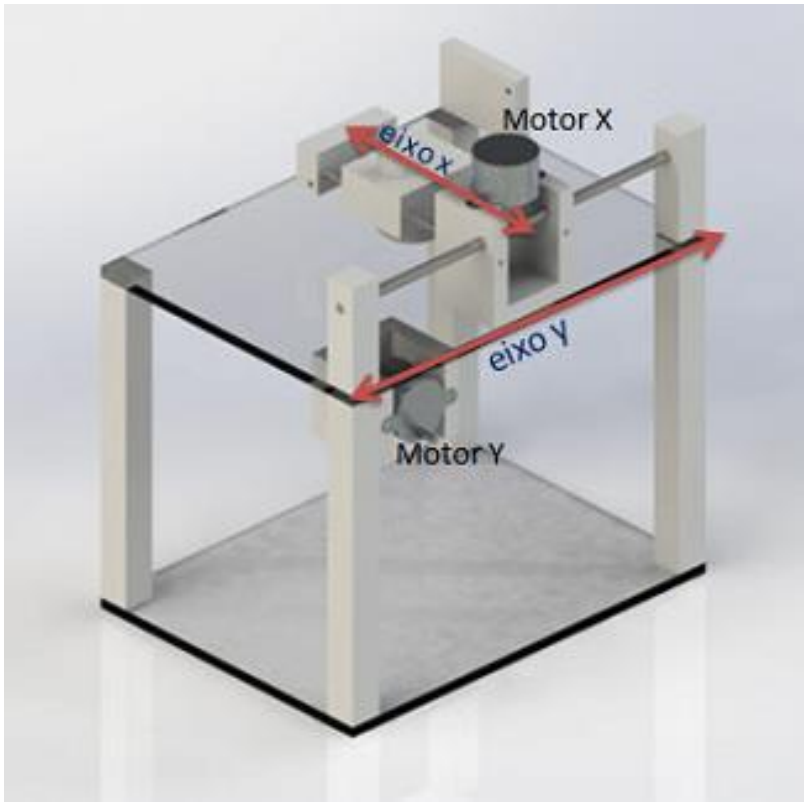


Figura 10 - Simulação em 3D

A Figura 11 mostra a base superior do protótipo, na qual ficarão presas as lentes a serem marcadas.

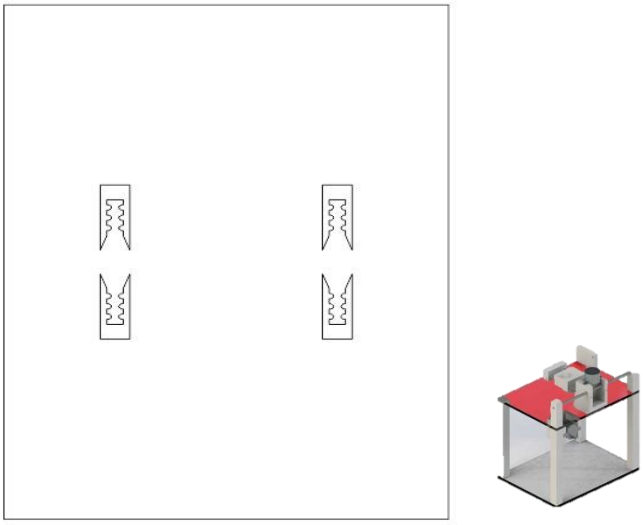


Figura 11 - Base superior

A base inferior, onde ficará apoiado boa parte da eletrônica do protótipo, será apenas uma chapa de 150 mm x 200 mm x 5 mm, igual a base superior, porém sem os furos para segurar a lente.

Estas duas peças, base inferior e superior, além de serem bases de apoio, têm função estrutural no projeto. Elas são os apoios das colunas de sustentação do protótipo.

A base superior ficará entre a lente e a câmera. É necessário que o material da base seja de tal maneira que a câmera consiga registrar a lente que se encontra sob a base. Por essa razão ela será confeccionada em acrílico, que é um material relativamente barato e com ótima transparência. Já a base inferior por uma questão de simetria e estética também será moldada em acrílico.

A Figura 12 mostra como serão as colunas de sustentação entre as bases.

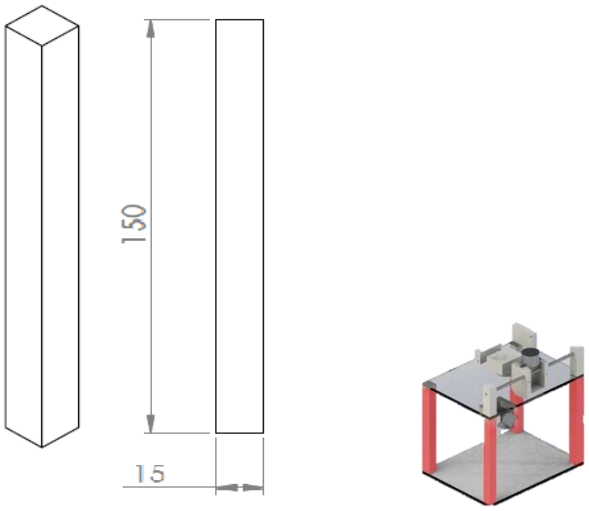


Figura 12 - Coluna de sustentação

Estas peças tem função apenas estrutural, portanto precisam ser de um material rígido. Para facilitar a utilização de um software de lentes solares, o material escolhido deveria ser branco, para não interferir na identificação da lente (escura). Por essa razão foi escolhido, para a maioria das peças, o nylon cru, que é um material barato, de fácil modelagem e branco.

A Figura 13 mostra o suporte para o motor X. Ele recebeu este nome porque sustenta o motor responsável pelo deslocamento do marcador na direção x.

Este suporte será uma peça móvel no protótipo, deslocando-se pelo eixo y. Na figura podemos ver três furos na estrutura, dois deles de 3 mm e dois de 5mm de diâmetro. Os furos de 3 mm são os

encaixes de hastes de aço prata pelas quais o marcador terá a liberdade de deslizar na direção x. Já os furos de 5 mm será o apoio da haste pela qual o suporte do motor X deslizará na direção y.

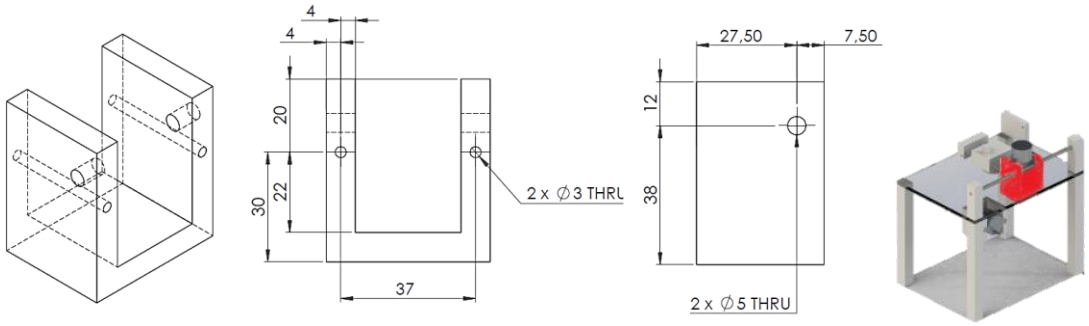


Figura 13 - Suporte motor X

A Figura 14 mostra o suporte para o motor Y. Ele recebeu este nome porque sustenta o motor responsável pelo deslocamento do marcador na direção y. Esta peça ficará presa na parte inferior da base superior.

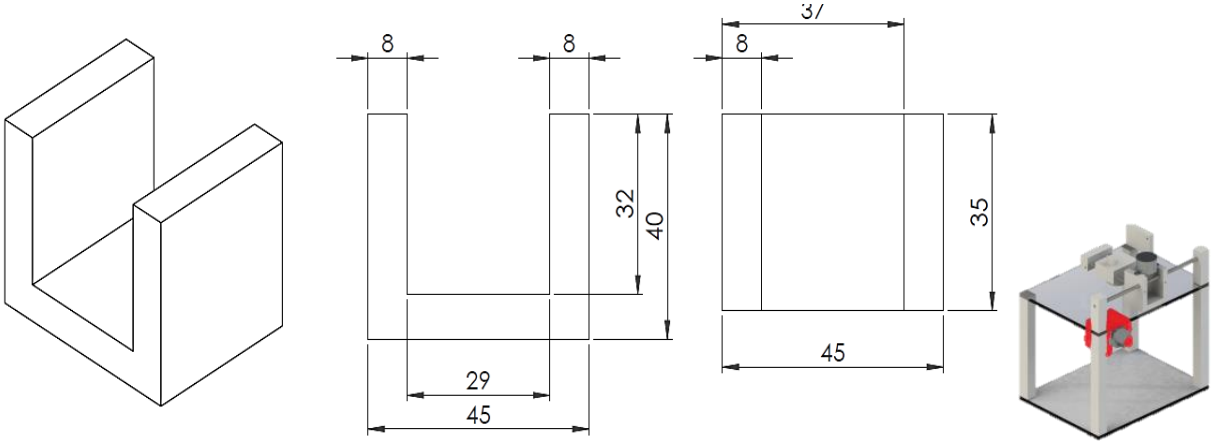


Figura 14 Suporte Motor Y

A Figura 15 mostra o delimitador de curso do eixo y. Esta estrutura serve para determinar até onde o motor pode percorrer o eixo y e também serve para fixar as colunas de sustentação na base superior.

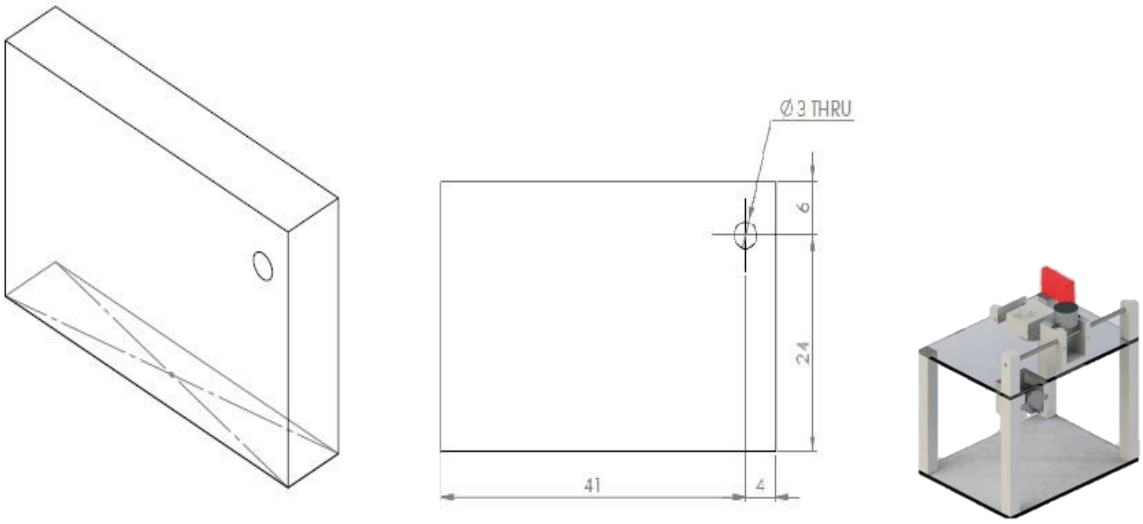


Figura 15 - Suporte contrário ao Motor Y

A Figura 16 mostra o delimitador de curso do eixo x, ou suporte contrario ao motor X. Neste suporte irão presas as duas hastes de aço prata de 3 mm de diâmetro nas quais o marcador deslizará na direção x.

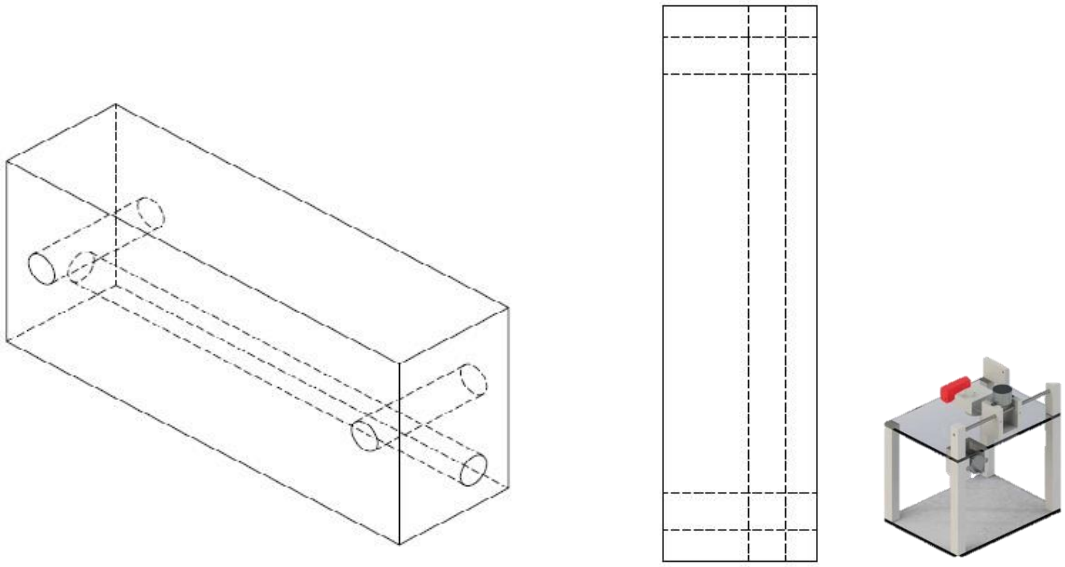


Figura 16 - Suporte contrário ao motor X

A Figura 17 mostra as colunas superiores. Essas colunas são responsáveis por sustentar a haste de 3 mm de diâmetro responsável pela direção y.

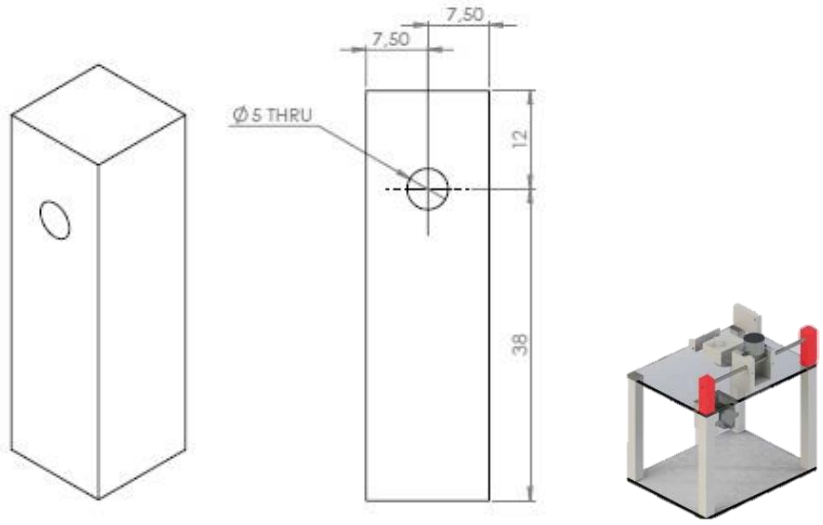


Figura 17 - Coluna Superior

A Figura 18 mostra a peça denominada de marcador. Nela podemos ver um furo de 2 mm de diâmetro. Ela recebeu esse nome, pois será por esse furo que será inserida a haste que marcará o centro da lente. Esta aba que se pode ver na parte superior da peça é onde irá presa à correia que fará o marcador se mover na direção x.

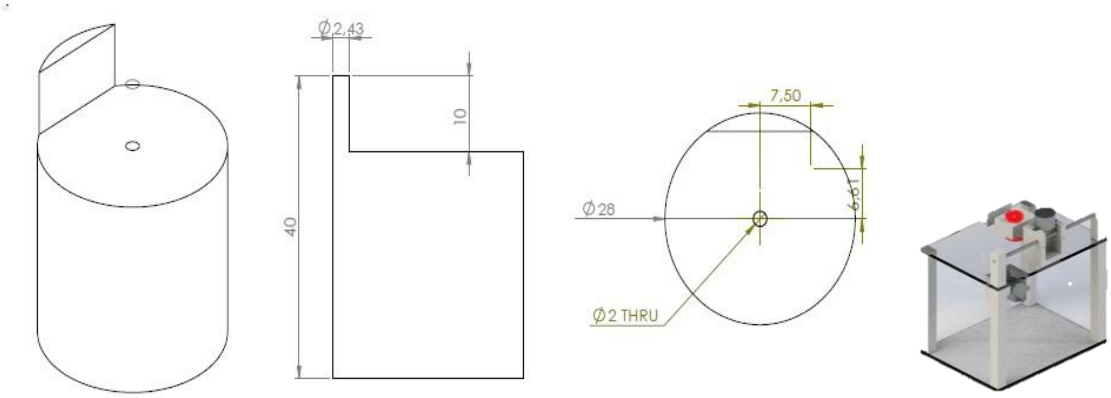


Figura 18- Marcador

A Figura 19 mostra o suporte do marcador. Esta peça possui um furo central de 28 mm de diâmetro no qual vai fixo o marcador. Os outros dois furos de 3 mm são por onde deslizarão duas hastes de aço prata as quais dão a liberdade desta peça se mover na direção x.

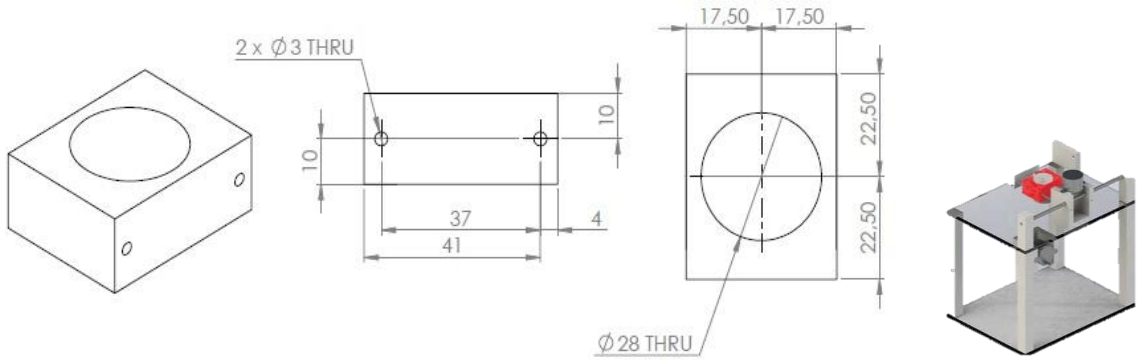


Figura 19 - Suporte Marcador

A Figura 20 mostra como ficou o protótipo após sua confecção. Nela pode-se ver a estrutura já com as polias e correias já fixadas em seus lugares.

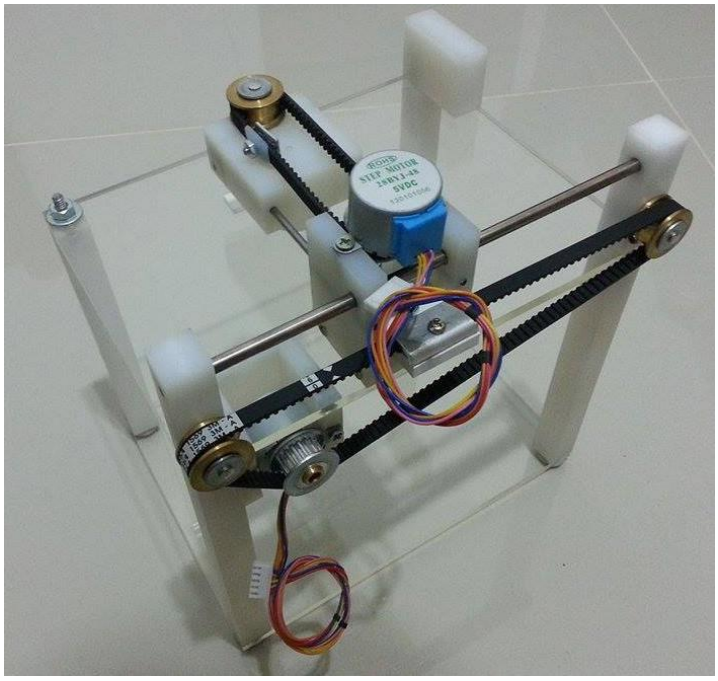


Figura 20 - Protótipo Confeccionado

3.2 Fiação e conexões

Além da parte estrutural, o projeto possui cabos, câmera, motores, microprocessador, botões e fonte de alimentação, ou seja, tudo que um protótipo automático precisa.

Assim como os desenhos, a escolha dos componentes a serem utilizados no projeto foi parte das suas primeiras etapas. Isso garantiu que todas as peças fossem projetadas e dimensionadas de acordo com os componentes escolhidos, não havendo assim problemas estruturais.

Como o projeto envolve componentes elétricos e eletrônicos a conexão entre eles e o centro de processamento é feita através de cabos, *jumpers* e fios.

Os periféricos, como a *webcam*, mouse e teclado, são conectados através das portas USB da *Raspberry Pi*. O teclado e o mouse são utilizados apenas nas fases de programação e no caso de algum tipo de manutenção. A *webcam* é responsável pela captura da imagem que será processada e utilizada na identificação da lente, portanto ela permanecerá conectada o tempo todo com o microcomputador.

Outra porta que é utilizada apenas na fase de programação ou manutenção é a saída HDMI. Nela é conectado um monitor o qual é usado, assim como em um microcomputador comum, como meio de interface com o usuário, ou seja, com ele é possível programar, configurar e testar os códigos fontes desenvolvidos.

A alimentação é feita através de uma porta micro USB na qual é conectada uma fonte de 5 volts por 2 amperes. Esta fonte é responsável por fornecer tensão e corrente para todo equipamento elétrico do protótipo.

Os motores de passo por sua vez são conectados nos pinos GPIO da *Raspberry Pi* através de fios popularmente chamados de "*jumpers".* Estes fios são comercializados com conexões fêmea ou macho. Para o protótipo foram utilizados "*jumpers"* com conexões fêmea-fêmea ou macho-fêmea.

A Figura 21 mostra como são feitas as conexões entre a *driver board* a *Raspberry Pi* e os motores. Como se pode ver elas são feitas com 6 "*jumpers"* fêmea-fêmea, 4 deles mandam os sinais relacionados aos enrolamentos do motor e outros 2 são para alimentação. Já a conexão entre a *driver board* e o motor é feita através de cinco fios com um conector específico para este motor. A Figura 22 mostra como é a conexão interna dos enrolamentos do motor.

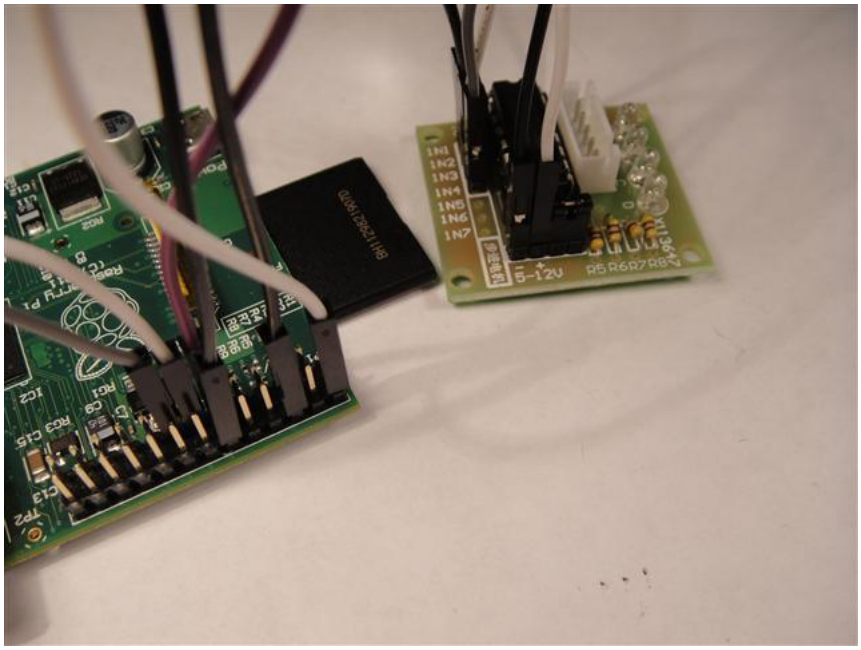


Figura 21 - Conexões Raspberry Pi – *Driver Board* [5]

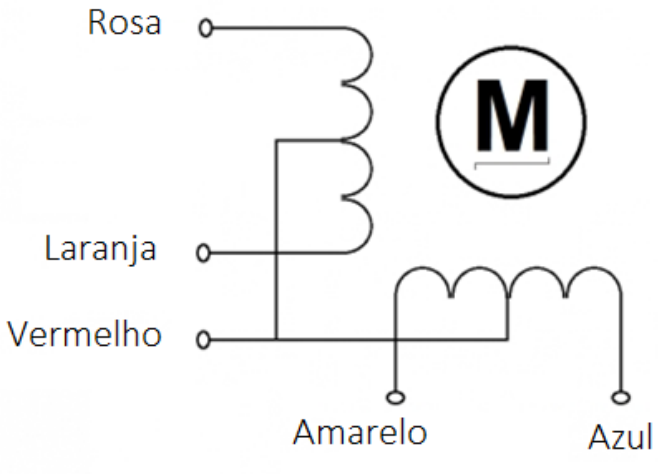


Figura 22 - Diagrama de conexões 28BYJ-48

O protótipo possui dois botões popularmente conhecidos por *"push button"*. Estes botões são responsáveis por dar os comandos de "*start"*, "retornar" e no caso de lentes dégradé comandos de "*next point"* para marcação dos pontos adicionais. As ligações dos *"push button"* são simples, ao apertar o botão o circuito é fechado e ao soltar o circuito é reaberto e a circulação de corrente é interrompida. Dessa forma um dos pinos do botão é ligado no "terra" e o outro pino em uma das portas GPIO da *Raspberry Pi*

[5] Figura disponível em: http://www.scraptopower.co.uk/Raspberry-Pi/how-to-connect-stepper-motors-a-raspberry-pi. Acesso em Outubro de 2015.

em paralelo com a alimentação de 3,3 volts. Assim quando o botão é pressionado ele fecha o circuito com o "terra" colocando a porta GPIO em nível baixo. A Figura 23 mostra como é feita a conexão entre o botão e a *Raspberry Pi.* Isso significa, que quando o botão não está pressionado, o valor de retorno do pino associado a ele é sempre alto. Quando o botão é pressionado, tem-se o valor baixo de tensão.

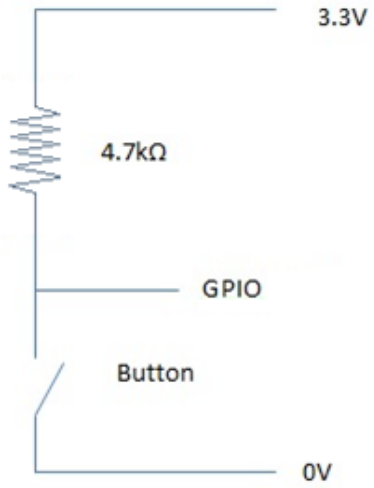


Figura 23 - Conexão botão – Raspberry Pi

A Figura 24 mostra um fluxograma o qual representa o funcionamento do protótipo, isto é, como é feita a ligação entre os materiais utilizados.

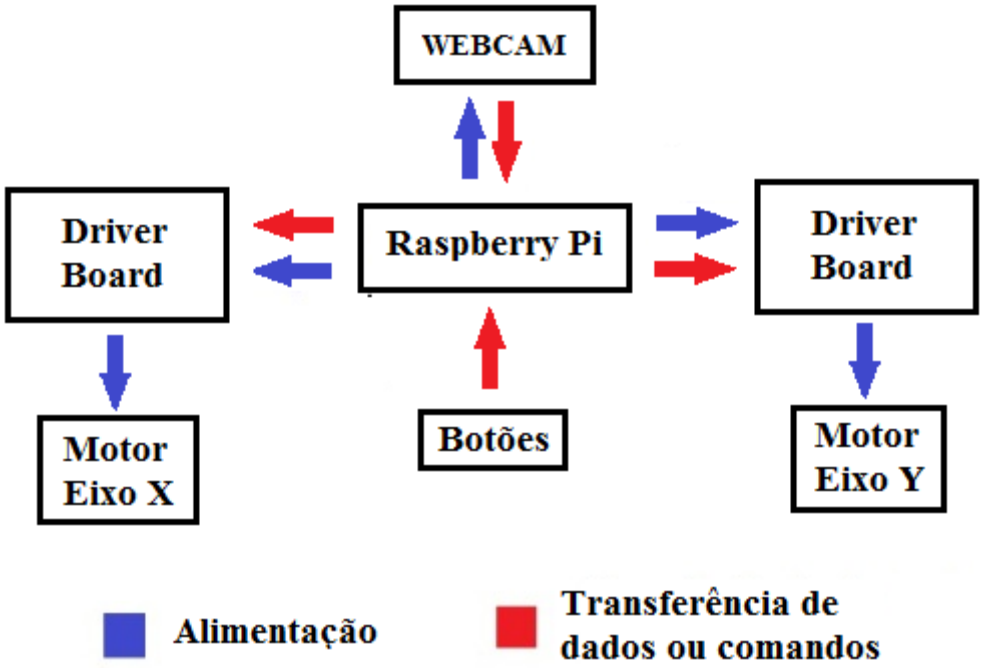


Figura 24 - Diagrama de Fluxo

3.3 O algoritmo

Essa seção explica como está organizada a comunicação entre os métodos. Cada passo do algoritmo é essencial para a o completo funcionamento do programa. Por isso, na implementação, todo o cuidado foi realizado para que o programa ficasse legível dividindo-o em arquivos e funções. O programa apresenta dois fluxos principais, o primeiro com a captura, marcação e processamento das imagens. O segundo fluxo é uma parada de emergência. Caso o motor trave, é possível corrigir manualmente a posição dos motores e reiniciar o programa.

3.3.1. Fluxo Normal de Funcionamento

Durante o estudo do desenvolvimento do software, decidiu-se criar um código em linguagem *Python*. Está escolha se baseou na facilidade de trabalhar essa linguagem na *Raspberry Pi*. Além disso, a *Python* é uma linguagem que trabalha bem com visão computacional através da biblioteca *OpenCV* [14].

O código desenvolvido foi programado para dois tipos de lentes, as lentes normais e as lentes dégradé. Para utilizar o protótipo é necessário **apertar um dos botões** presentes. O botão da esquerda tem função de selecionar o modo "lente normal" e dar o comando de "voltar à posição inicial". Já o botão da direita é responsável por selecionar o modo "lente dégradé" e dar os comandos "próximo ponto" e "voltar à posição inicial".

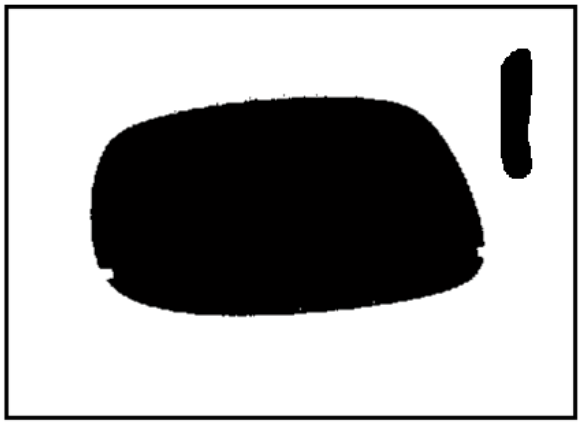
O código para utilizar os botões na *Raspberry Pi* é simples e utiliza apenas uma condicional. O botão é conectado conforme ilustrado na figura 23. A partir daí, o pino de entrada ligado ao botão estará sempre em alta (retornará *True* ao programa em *Python*). Ao pressionar o botão, fecha-se o circuito para o neutro e o pino fica em "baixa" (retorna *False*). Desse modo define-se uma condicional a qual determina que o programa realize tal função caso o pino associado ao botão retorne o valor *False.*

A Figura 25 mostra passo a passo o processamento da imagem até por fim encontrar centro e os pontos desejados na lente, no caso de lentes comuns, e marcação dos pontos adicionais são descartadas. O primeiro passo depois de capturada a imagem é a **binarização** da mesma. Esta etapa utiliza um limiar definido automaticamente pela função "*cv2.THRESH_OTSU"* para transformar a imagem original em uma imagem binária, ou seja, com apenas dois tons de cinza, branco e preto. Após a binarização a imagem é **invertida**, utiliza-se o negativo da imagem para que a lente se torne branca e se possa trabalhar com ela.

O próximo passo é encontrar os **componentes conectados** da imagem. Esta etapa é responsável por eliminar possíveis objetos presentes na imagem que não fazem parte da lente. O programa rotula cada componente da imagem binária e elimina todos aqueles que não forem pertencentes ao de maior área. Nesta etapa, é assumido que a lente ocupa a maior parte da imagem e que outros objetos são apenas ruídos a serem eliminados para não interferir no cálculo do centro.

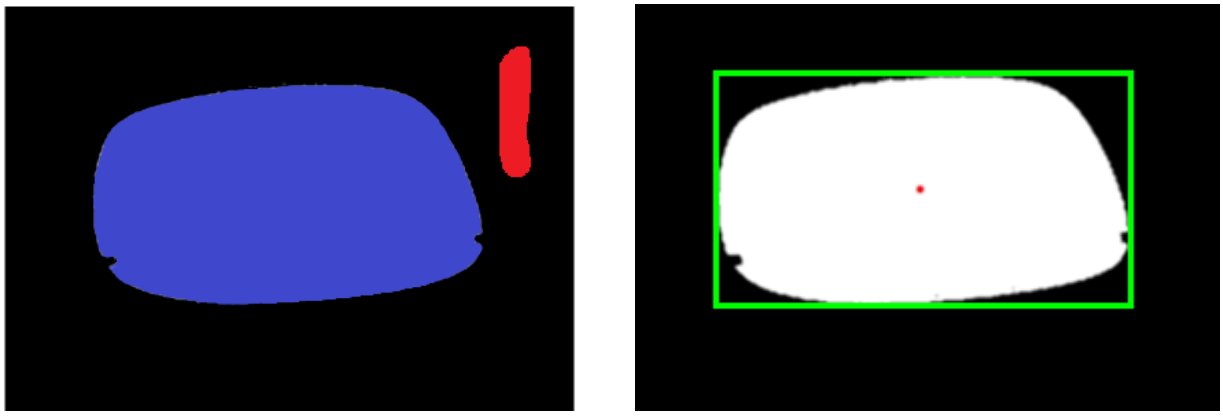
Por último é calculado o centro. A função "*cv2.boundingRect"* é utilizada para encontrar os pontos extremos da imagem. Para o cálculo do centro geométrico da lente, são utilizados os pontos extremos da imagem, sua largura e seu comprimento, a partir destas medidas é calculado o seu centro, como foi mostrado na seção 2.1.6. Nesse ponto, a imagem é constituída apenas da lente em branco, por isso garantimos que o centro geométrico encontrado é o centro da lente.

Se a lente for dégradé, é necessária a **marcação dos pontos adicionais**. Assim como para relacionar o andar dos motores com a quantidade de pixels, também foi calculado a relação 10 mm e quantidade de pixels para a marcação dos pontos adicionais. É importante que essa marcação seja feita com relação ao eixo central da lente.



(a) Binarização por método de Otsu

(b) Inversão dos pixels da imagem



(c) Rotulação por componentes conectados (d) identificação e marcação do centro geométrico

Figura 25 - Processamento da imagem

A segunda parte do programa desenvolvido tem a função de **movimentar o motor**. Durante esses meses foi possível estudar o funcionamento do motor de passo e da *driver board* adquiridos e desenvolver um programa que consegue movimentar o motor de acordo com os parâmetros atribuídos e quantidade de passos, sentido e velocidade solicitada.

O código desenvolvido para o movimento dos motores é simples. Cada motor possui dois enrolamentos com um "tap" central cada, que são "curtocircuitados" entre si, como mostra a Figura 22. O programa começa definindo os pinos de saída da *Raspberry Pi*, de acordo com o motor a ser utilizado, X ou Y) e em seguida uma sequência que é enviada para as 2 entradas desses dois enrolamentos. A partir dai, inicia-se um laço o qual irá seguir a sequência de comandos a ser enviada para os pinos de saída e incrementará a variável que controla a quantidade de passos dada. Após o motor ter girado o número de voltas estabelecido o programa aguarda um comando, que é dado por um botão, para que **o motor volte para a posição inicial**. Este mesmo padrão de código é utilizado tanto para movimentar o eixo X quando para movimentar o eixo Y.

O programa primeiramente se movimenta para marcar o centro da lente, movendo os motores X e Y um de cada vez. Quando alcançada a posição central, o programa espera que o botão correspondente ao tipo da lente detectada seja pressionado. Isso é utilizado para que o usuário tenha tempo suficiente de colocar o marcador no furo designado. Se houver necessidade, no caso de lentes dégradé, os passos são repetidos para a **marcação dos pontos adicionais**.

O código engloba desde a captura da imagem, gerada pela *webcam*, até o comando dos passos dados para os motores. A seguir encontra-se o fluxograma resumido, do programa desenvolvido, para facilitar a compreensão de como funciona e os passos que o protótipo executa no decorrer do seu funcionamento.

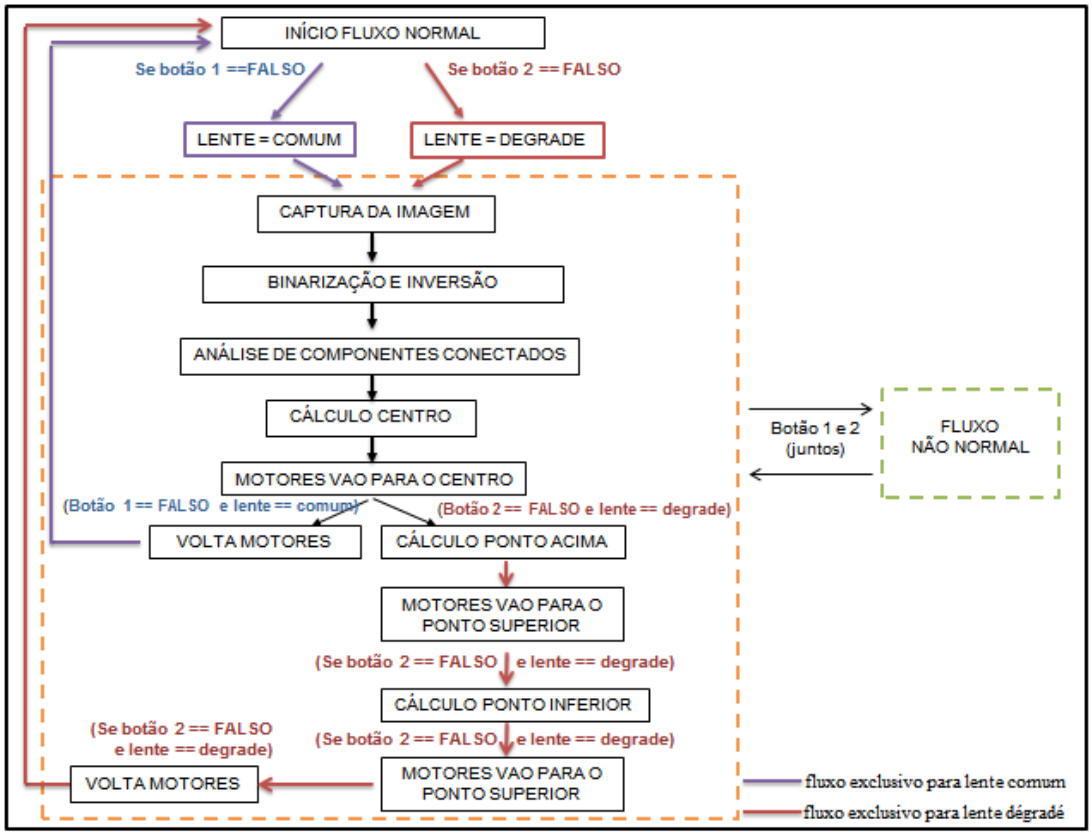


Figura 26 – Fluxograma Normal do algoritmo

3.3.2. Parada de Emergência do Programa (Desvio do Fluxo Normal)

A qualquer momento do funcionamento do programa o usuário pode pará-lo pressionando os dois botões ao mesmo tempo.

No entanto, como o inicio do programa supõe-se que os cursores estejam em sua posição inicial ao iniciar uma nova detecção da lente, é necessário que esses sejam retornados corretamente. Para isso, é possível controlar os dois motores X e Y para que voltem a posição inicial pressionando os botões 1 e 2 respectivamente.

Para relembrar o usuário, é anexado uma etiqueta em cada um dos botões informando à qual lente está relacionado (para inicio do fluxo normal) e à qual motor está relacionado. No fluxo 'não normal', enquanto o botão 1 estiver apertado, o motor X é retornado em 1 passo. Do contrário, se o botão 2 estiver pressionado é o motor Y que retorna um passo. Para voltar ao inicio do fluxo normal, ou seja, para a escolha de uma nova lente (comum ou dégradé) a ser detectada e marcada, basta apertar os dois botões ao mesmo tempo. A Figura 27 mostra o fluxograma do programa não padrão.

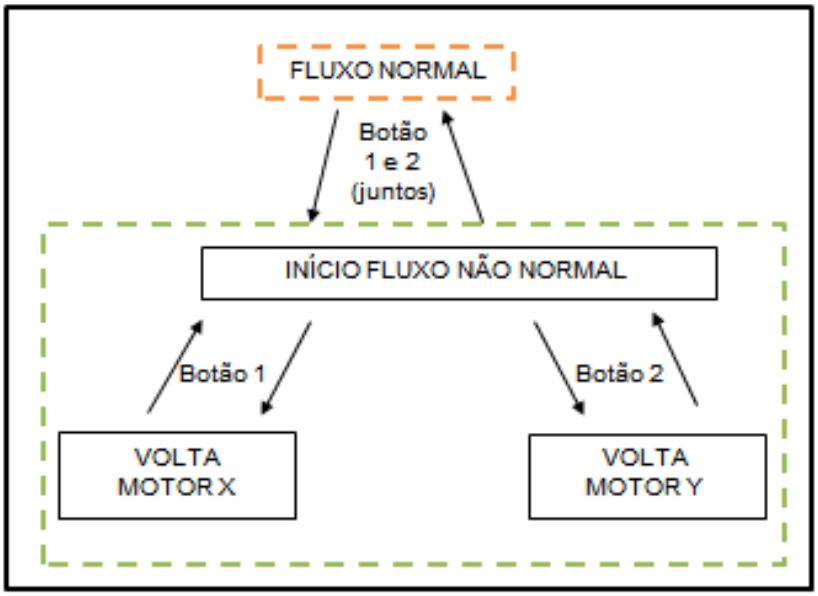


Figura 27- Fluxo Não Normal do Programa (Parada de Emergência)

3.3.3. Funções de implementação

Dado o inicio do programa, algumas funções foram criadas para facilitar a criação do fluxo do programa. Inicialmente, após a escolha de uns dos botões, a função ***pegaDaWebcam*** acessa a câmera e retorna ao programa principal uma foto retirada pela *webcam*.

Para binarizar a imagem capturada da *webcam*, a função ***binarizacao*** recebe uma imagem em níveis de cinza e através do método de Otsu, converte-a para uma imagem de apenas dois níveis. Além disso, a imagem recebe um parâmetro para inverter ou não os pixels (negativo da imagem), necessário para o inicio do programa, já que queremos a lente em branco. A função ***componentesConectados*** recebe uma imagem binária e retorna uma imagem onde só o maior componente é exibido.

Para o cálculo do centro, a função ***calculaCentro*** recebe a imagem de saída dos *componenteConectados* e retorna as posições em pixels de x e y do centro. O método utiliza os pontos extremos da imagem para calcular o centro geométrico da imagem, ou seja, da lente.

A função ***andaMotor*** por exemplo, recebe como parâmetro o número de passos para o motor, a identificação de qual motor deve mover e a direção do motor. Isso facilita as chamadas das funções das quais os motores devem ir para o centro e pontos adicionais. Para calcular a relação pixel-passo-mm foi medido quantos passos eram necessários para o eixo andar 5 mm (e a quantidade de pixels relacionada) e para qualquer quantidade de pixels é utilizada uma regra de três.

O cálculo da quantidade de passos para pontos adicionais é feito manualmente, pois só é necessário andar um número fixo de passos no motor Y. O capítulo Apêndice detalha todos os códigos desenvolvidos na linguagem Python.

4. RESULTADOS

Após o desenvolvimento completo do projeto incluindo a montagem e os códigos, foram realizados ensaios para certificar e obter resultados a fim de comprovar a eficiência do protótipo.

Neste capítulo serão apresentados os resultados obtidos a partir dos ensaios realizados. Nos testes foram utilizadas 32 de diferentes tamanhos e diferentes tipos. Os ensaios incluem a medição de 11 lentes dégradées e 21 lentes comuns.

As medições feitas pelo protótipo foram comparadas com as medições feitas à mão, método de estimativa do centro geométrico utilizado pelo laboratório LIO atualmente e única medida disponível, o qual faz uso de um paquímetro, aparelho de medição de precisão. A partir destas medidas foram geradas quatro tabelas para fim de comparação e análise. A variação foi calculada de acordo com a Equação 13.

$$Distância = \sqrt{(X_{Manual} - X_{Automático})^2 + (Y_{Manual} - Y_{Automático})^2} \quad (13)$$

A NBR ISO 12312-1 [1] não define uma margem de erro para o centro geométrico das lentes. Em casos como este, como padrão, podemos usar como margem aceitável um erro de ±0,5 da menor unidade utilizada para a medição, neste caso ±0,5 mm.

As coordenadas dos pontos encontrados manualmente e pelo protótipo obedecem ao padrão matricial, ou seja, o ponto inicial (0,0) está no extremo superior esquerdo. Além disso, estas coordenadas estão em milímetros, ou seja, o ponto de coordenada (5,5) está a 5 mm à direita da borda esquerda e 5 mm a baixo da borda superior da lente. A Figura 28 exemplifica estas coordenadas.

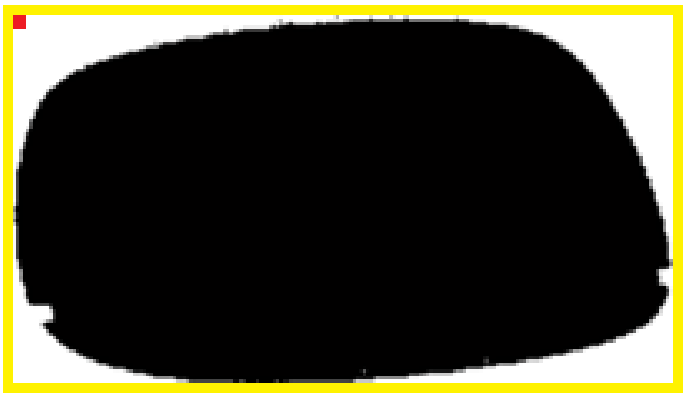


Figura 28 - Coordenadas dos pontos

4.1 Lentes comuns

A Tabela 1 mostra os resultados obtidos com os 21 ensaios realizados com lentes comuns. Na tabela podem-se ver as coordenadas x e y dos centros geométricos encontrados pelo protótipo nas 4 medições realizadas em cada lente.

A média dos centros geométricos, também mostrada na Tabela 1, é considerada como ponto de comparação.

Lentes	Centros encontrados automaticamente				
	(X_{Aut}, Y_{Aut}) [mm]	(X_{Aut}, Y_{Aut})[mm]	(X_{Aut}, Y_{Aut})[mm]	(X_{Aut}, Y_{Aut})[mm]	Centro médio [mm]
01	(23.54,21.78)	(23.44,21.72)	(23.58,21.7)	(23.32,21.52)	(23.47,21.68)
02	(23.18,21.42)	(23.4,21.28)	(23.22,21.22)	(23.44,21.22)	(23.31,21.28)
03	(26.1,15.72)	(26.3,16.0)	(26.2,15.54)	(26.36,16.12)	(26.24,15.84)
04	(31.06,18.5)	(30.1,18.08)	(30.26,18.56)	(30.18,18.18)	(30.4,18.33)
05	(28.5,17.0)	(28.6,17.12)	(28.56,17.44)	(28.46,17.2)	(28.53,17.19)
06	(28.7,17.36)	(28.76,17.3)	(28.78,17.22)	(28.56,17.32)	(28.7,17.3)
08	(30.28,18.54)	(30.42,18.34)	(30.22,18.44)	(30.28,18.38)	(30.3,18.42)
09	(33.88,17.52)	(33.8,17.4)	(33.82,17.56)	(33.68,17.74)	(33.79,17.55)
E14	(27.3,17.62)	(27.38,17.64)	(27.48,17.68)	(27.3,17.34)	(27.36,17.57)
E126	(27.06,21.6)	(26.8,21.32)	(26.58,21.18)	(26.74,21.42)	(26.79,21.6)
E244	(33.16,19.12)	(33.06,19.0)	(32.92,19.3)	(33.22,19.72)	(33.09,19.29)
E39	(24.34,20.46)	(24.3,20.4)	(24.54,20.34)	(24.38,20.48)	(24.39,20.42)
E128	(23.3,22.06)	(23.12,22.04)	(23.54,21.86)	(23.34,21.66)	(23.32,21.90)
E22	(29.48,18.0)	(29.4,17.66)	(29.0,17.84)	(29.2,17.88)	(29.27,17.84)
E301	(32.64,21.04)	(32.12,21.38)	(32.36,21.24)	(32.7,21.08)	(32.45,21.18)
E230	(33.8,19.42)	(33.96,19.32)	(33.68,19.42)	(33.66,19.12)	(33.77,19.32)
E198	(32.6,26.04)	(32.68,26.24)	(32.62,26.04)	(32.68,25.98)	(32.64,26.07)
E240	(30.02,18.4)	(30.10,18.32)	(30.2,18.4)	(29.88,18.1)	(30.05,18.30)
E232	(32.5,20.0)	(32.54,20.08)	(32.36,20.2)	(32.72,20.2)	(32.53,20.12)
E54	(32.82,22.0)	(31.98,21.68)	(32.34,21.56)	(33.48,22.16)	(32.65,21.85)
E241	(31.22,20.6)	(31.02,20.68)	(31.18,20.56)	(31.96,20.86)	(31.09,20.67)

Tabela 1 - Ensaios realizados pelo protótipo com lentes comuns

Na Tabela 2 são exibidos os resultados obtidos com as lentes comuns. Nela são mostrados os centros geométricos encontrados com as medições manuais, as quais são comparadas com as médias das medições feitas pelo protótipo. Assim são calculadas as distâncias entre esses dois pontos através da equação (13).

Podemos observar através da Tabela 2 que, considerando uma margem de 0,5 mm entre os dois pontos, os resultados gerados estão em concordância em 71,5% das lentes.

Lentes	Centro encontrado manualmente (X_{Man}, Y_{Man}) [mm]	Centro médio encontrado automaticamente (X_{Aut}, Y_{Aut}) [mm]	Distância euclidiana entre os centros [mm]
01	(23.58,21.7)	(23.47,21.68)	0,11
02	(23.46,21.36)	(23.31,21.28)	0,17
03	(26.5,15.7)	(26.24,15.84)	0,30
04	(30.68,18.30)	(30.4,18.33)	0,28
05	(29.3,17.0)	(28.53,17.19)	0,79
06	(29.26,17.6)	(28.7,17.3)	0,64
08	(30.3,18.38)	(30.3,18.42)	0,04
09	(34.12,17.84)	(33.79,17.55)	0,43
E14	(27.2,16.86)	(27.36,17.57)	0,73
E126	(26.7,21.36)	(26.79,21.6)	0,26
E244	(32.8,19.62)	(33.09,19.29)	0,45
E39	(24.42,20.0)	(24.39,20.42)	0,42
E128	(23.34,21.56)	(23.32,21.90)	0,35
E22	(29.68,17.6)	(29.27,17.84)	0,48
E301	(32.3,20.98)	(32.45,21.18)	0,26
E230	(33.0,19.52)	(33.77,19.32)	0,80
E198	(32.46,25.28)	(32.64,26.07)	0,82
E240	(30.06,18.7)	(30.05,18.30)	0,40
E232	(32.72,19.6)	(32.53,19.87)	0,33
E54	(32.86,21.78)	(32.65,21.85)	0,22
E241	(32.2,20.6)	(31.09,20.67)	0,86

Tabela 2 - Resultados lentes comuns

4.2 Lentes dégradé

A Tabela 3 mostra os resultados obtidos com os 11 ensaios realizados com lentes dégradées. Na tabela podem-se ver as coordenadas x e y dos centros geométricos encontrados pelo protótipo nas 4 medições realizadas em cada lente.

A média dos centros geométricos, também mostrada na Tabela 3, é considerada como ponto de comparação.

Lentes	Distância média				Centro médio [mm]
	$(X_{Aut}, Y_{Aut})[mm]$	(X_{Aut}, Y_{Aut})[mm]	(X_{Aut}, Y_{Aut})[mm]	(X_{Aut}, Y_{Aut})[mm]	
07	(28.14,22.02)	(27.94,22.12)	(28.26,21.78)	(28.04,22.3)	(28.09,22.05)
E269	(33.02,24.5)	(33.52,24.22)	(33.22,24.34)	(33.88,24.92)	(33.41,24.49)
E181	(33.94,21.7)	(33.32,21.54)	(33.64,21.86)	(34.02,22.42)	(33.73,21.88)
E196	(33.3,24.18)	(33.12,24.28)	(33.64,24.86)	(33.34,24.58)	(33.35,24.47)
E220	(33.12,24.34)	(33.82,24.24)	(33.64,24.12)	(34.02,24.64)	(33.65,24.33)
E75	(31.72,26.12)	(31.34,26.12)	(31.52,26.04)	(31.12,25.72)	(31.425.26)
E263	(33.24,23.88)	(33.64,23.7)	(33.8,24.18)	(33.14,24.38)	(33.45,24.03)
E233	(32.14,24.32)	(31.92,24.84)	(32.54,24.92)	(32.2,25.2)	(32.2,24.82)
E348	(30.9,22.16)	(30.72,22.06)	(31.12,21.76)	(31.42,21.9)	(31.04,21.97)
E30	(30.52,23.74)	(30.84,23.44)	(30.8,23.94)	(31.14,24.3)	(30.82,23.85)
E270	(33.64,21.42)	(33.3,21.2)	(34.18,21.88)	(35.1,22.18)	(34.05,21.67)

Tabela 3 - Ensaios realizados pelo protótipo com lentes dégradées

Na Tabela 4 são exibidos os resultados obtidos com as lentes dégradé. Nela são mostrados os centros geométricos encontrados com as medições manuais, as quais são comparadas com as médias das medições feitas pelo protótipo. Assim são calculadas as distâncias entre esses dois pontos por meio da equação (13).

Podemos observar através da Tabela 4 que, considerando uma margem de 0,5 mm entre os dois pontos, os resultados gerados estão em concordância em 82% das lentes.

Lentes	Centro encontrado manualmente (X_{Man}, Y_{Man}) [mm]	Centro médio encontrado automaticamente (X_{Aut}, Y_{Aut}) [mm]	Distância euclidiana entre centros [mm]
07	(28.28,21.86)	(28.09,22.05)	0,27
E269	(33.52,24.5)	(33.41,24.49)	0,11
E181	(33.84,21.9)	(33.73,21.88)	0,11
E196	(33.58,24.36)	(33.35,24.47)	0,26
E220	(33.76,24.56)	(33.65,24.33)	0,25
E75	(31.32,25.8)	(31.42,26.0)	0,23
E263	(33.6,24.58)	(33.45,24.03)	0,56
E233	(32.56,25.1)	(32.2,24.82)	0,46
E348	(31.56,22.14)	(31.04,21.97)	0,55
E30	(31.06,24.08)	(30.82,23.85)	0,33
E270	(33.78,22.1)	(34.05,21.67)	0,45

Tabela 4 - Resultados lentes dégradées

5. CONCLUSÕES E TRABALHOS FUTUROS

O trabalho de conclusão de curso, além de ser obrigatório para a sua formação, está desempenhando um importante papel no crescimento do aluno. Mostrando as dificuldades na realização prática de um projeto e ensinando novos conceitos a respeito de eletrônica, programação e bioengenharia.

O objetivo deste projeto era projetar e construir um protótipo automático que encontrasse e marcasse o centro geométrico de lentes de óculos de sol, para com isso facilitar e agilizar o trabalho de análise e ensaios com as lentes. Durante o período de dez meses foi possível finalizar todas as etapas planejadas, como se pode ver no cronograma acima e os resultados obtidos foram muito bons levando em consideração uma margem de erro de até 2 mm, a qual é determinada pela NBR ISO 12312-1 [1] no ensaio de deformação da ponte de óculos solares. Como a NBR ISO 12312-1 [1] não define uma margem de erro geral para o centro geométrico foi usada uma margem aceitável de +/-0,5 mm de variação entre a média dos centros geométricos encontrados pelo protótipo e o centro geométrico encontrado manualmente. Desta forma o protótipo se mostrou confiável dando resultados de 82% de concordância para lentes dégradées e 71,5% de concordância para lentes comuns.

Além disso, foi possível perceber que as áreas de visão computacional e processamento de imagens são ótimas ferramentas para este tipo problema, porém ainda assim existem problemas, como por exemplo o método de análise do principal componente. Esta ferramenta funciona de forma excelente em estruturas mais uniformes do que as lentes, porém para este tipo de aplicação o *PCA* não pode ser aplicado.

Futuramente o projeto pode ser aperfeiçoado. Algumas ideias como confecção de uma placa de circuito impresso maior e mais completa é uma delas. Além disso, o projeto poderia contemplar uma tela de LCD, do inglês, *Liquid Crystal Display*, ou seja, tela de cristal líquido, a qual poderia fornecer informações para o usuário como, por exemplo, em que estágio está o programa, se o programa está pronto para ser iniciado e até as coordenadas dos pontos de referência encontrados pelo protótipo. Outro tipo de melhoria válida para este projeto seria atualizar o modelo da *Raspberry Pi* utilizado. Hoje em dia já existem modelos deste microcomputador mais completos, com mais pinos de GPIO e outras facilidade que poderiam ajudar no aperfeiçoamento do protótipo. Com mais pinos de entrada e saída, por exemplo, poderiam ser desenvolvidas outras funções com mais botões para facilitar a interface com o usuário.

6. CRONOGRAMA DESENVOLVIDO

Atividades	Fev/Março	Abril/Maio	Junho/julho	Agosto/Set	Out/Nov
1	**X**				
2	**X**	**X**			
3		**X**	**X**	**X**	
4		**X**	**X**	**X**	
5				**X**	**X**

1. Desenho do protótipo
2. Desenvolvimento do hardware
3. Desenvolvimento do software
4. Confecção e montagem
5. Ajustes e testes finais

BIBLIOGRAFIA

[1] A. B. D. N. TÉCNICAS, NBR 12312-1: Proteção dos olhos e do rosto: óculos para proteção solar e óculos relacionados: parte 1: óculos para proteção solar para uso geral., Rio de Janeiro, 2015.

[2] L. Gomes e L. Ventura, "Development of automated prototype for studying the effect of solar aging on sunglasses," em *SPIE Biophotonics South America*, 2015.

[3] R. Magri e L. Ventura, "Óculos de Sol: Sistema para Verificar a Inflamabilidade," em *XXIV Congresso Brasileiro de Engenharia Biomédica*, Uberlândia, 2014.

[4] G. M. Iwamoto, *Automatização do Teste de Medida de Impacto em Lentes Solares,* São Carlos, 2012.

[5] S. S. Maurício Marengoni, "Tutorial: Introdução à Visão Computacional usando OpenCV," *RITA: Revista de Informática Teórica e Aplicada,* vol. 16, n. 1, pp. 125-160, 2009.

[6] R. C. Gonzalez e R. E. Woods, Digital Image Processing, Pearson, 2002.

[7] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica,* pp. 23--27, 1975.

[8] H. H. Abass e F. M. M. Al-Salbi, "Rotation and Scaling Image Using PCA," *Computer and Information Science,* p. 97, 2011.

[9] S. X. Liao, *Image Analysis by Moments,* Manitoba, 1993.

[10] A. B. D. N. TÉCNICAS, ABNT NBR 15111: Proteção pessoal dos olhos – Óculos de sol e filtros de proteção contra raios solares para uso geral: citações em documentos: apresentação. Rio de Janeiro, 2004., Rio de Janeiro, 2004.

[11] E. U. G. Halfacree, *Raspberry Pi Manual do Usuário,* São Paulo, 2013.

[12] Changzhou Fulling Motor Co. Ltd, "Datasheet Motor de Passo 28BYJ-48.," [Online]. Available: http://www.geeetech.com/Documents/Stepper%20motor%20datasheet.pdf. [Acesso em Outubro 2015].

[13] Texas Instruments, "Datasheet Circuito Integrado ULN2003," [Online]. Available: www.geeetech.com/Documents/ULN2003%20datasheet.pdf. [Acesso em Outubro 2015].

[14] N. N. C. Menezes, Introdução à Programação com Python: Algoritmos e Lógica de Programação para Iniciantes, 2ª ed., São Paulo: Novatec, 2014.

APÊNDICE

Segue abaixo a relação de código e devidas explicações de cada função.

Função: **main**

- Arquivo: **principal.py**
- Programa principal acionado pela raspberry quando ligada

```
import RPi.GPIO as GPIO
from lenteFunc import lente

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(11, GPIO.OUT)
GPIO.output(11, False)

botao_Comum = 25
botao_Degrade = 8
botao_reset = 7

#seta os pinos como entradas
GPIO.setup(botao_reset, GPIO.IN)
GPIO.setup(botao_Comum, GPIO.IN)
GPIO.setup(botao_Degrade, GPIO.IN)

ligado = 1
printed = 0

while ligado:

    if(printed == 0):
        print "aperte p comecar"
        GPIO.output(11, True) #acende o LED
        printed = 1

    #lente comum
    if((GPIO.input(botao_Comum) == False) &( GPIO.input(botao_Degrade)
== True)):
        GPIO.output(11, False) #apaga LED
        lente(botao_Comum)
        printed = 0

    #lente degrade
    elif((GPIO.input(botao_Degrade) == False) &
(GPIO.input(botao_Comum) == True)):
        GPIO.output(11, False) #apaga LED
```

```
        lente(botao_Degrade)
        printed = 0
```

Função: **lente**

- Arquivo: **lenteFunc.py**
- Invocada por: **main**
- Análisa se é lente comum ou degradê, acha os pontos centrais e anda os motores para os pontos necessários. Além disso verifica se os botões foram pressionados para a parada de emergência (desvio do fluxo normal).

```
from imageProcess import pegarPontosImagem_Lente
from motorFunc import motorAndar
import RPi.GPIO as GPIO
from paradaEmergencialFunc import paradaEmergencial

#pin botao - lenteDegrade = 8
#pin botao - lenteComum = 25
#led = 7
def lente(botao):
    parou = 0

    botao_Degrade = 8
    botao_Comum = 25

    #encontra centro
    centrox,centroy = pegarPontosImagem_Lente()
    print centrox, centroy

    #marcar o centro
    print "andando Y"
    parou = motorAndar((2650+(14.5*centrox)), 1, 1) #anda motor Y
pro rumo do centro
    print "andando X"

    if(parou == 1):
        paradaEmergencial()
        return
    parou = motorAndar(((centroy*16.8)-600), 2, -1) #anda motor X
pro rumo do centro
    if(parou == 1):
        paradaEmergencial()
        return

    #espera proximo clique
    GPIO.output(11, True)
    print "aperte botao para voltar ou proximo(caso lente degrade)"

    # LENTES COMUNS
```

```
    #volta ao comeco dos eixos os motores
    if(botao == botao_Comum):
        voltar = 1
        while voltar:
            if (GPIO.input(botao_Comum) == False):
                GPIO.output(11, False)
                print "voltando X"
                parou = motorAndar(((centroy*15.5)-610), 2, 1)
#anda motor X de volta

                if(parou == 1):
                    paradaEmergencial()
                    return

                print "voltando Y"
                parou = motorAndar((1.18*(2600+(14.4*centrox))), 1,
-1) #anda motor Y de volta

                if(parou == 1):
                    paradaEmergencial()
                    return

                voltar = 0
    #LENTES DEGRADE
    elif(botao == botao_Degrade):
        voltar = 1

        #ponto acima
        while voltar:
            if (GPIO.input(botao_Degrade) == False):
                GPIO.output(11, False)
                parou = motorAndar(760, 2, -1) #anda motor X
                if(parou == 1):
                    paradaEmergencial()
                    return
                voltar = 0

        #espera proximo clique
        voltar = 1
        GPIO.output(11, True)
        print "aperte botao para voltar ou proximo(caso lente
degrade)"

        #ponto abaixo
        while voltar:
            if (GPIO.input(botao_Degrade) == False):
                GPIO.output(11, False)
                parou = motorAndar(2*760*0.87, 2, 1) #anda motor X
                if(parou == 1):
                    paradaEmergencial()
                    return
```

```
                voltar = 0

        voltar = 1

        #espera proximo clique
        voltar = 1
        GPIO.output(11, True)
        print "aperte botao para voltar ou proximo(caso lente
degrade)"

        while voltar:
            if(GPIO.input(botao_Degrade) == False):
                GPIO.output(11, False)
                parou = motorAndar((((centroy*15.4)-610)-(760*0.7)),
2, 1)  #anda motor X de volta
                if(parou == 1):
                    paradaEmergencial()
                    return
                parou = motorAndar(1.17*(2580+(14.4*centrox)), 1, -
1) #anda motor Y de volta
                if(parou == 1):
                    paradaEmergencial()
                    return
                voltar = 0
```

Função: **pegarPontosImagem_Lente**

- Arquivo: imageProcess.py
- Invocada por: **lente**
- Retorna o centro da lente. Converte a imagem para binário, utiliza *componentesConectados* para eliminar regiões além da lente. Depois chama a função *calculaCentro* para retornar os pontos centrais

```
from componentesConectadosFunc import componentesConectados
from retornaPontosFunc import calculaCentro
from converterImagemFunc import binarizacao
from converterImagemFunc import convertParaEscaladeCinza
from pegaDaWebcamFunc import pegaDaWebcam
from converterImagemFunc import gammaCorrect
from converterImagemFunc import converterParaHSV
import matplotlib.pyplot as MPL
import cv2


#recebe a imagem da webcam e retorna
#o (centrox, centroy)
def pegarPontosImagem_Lente():
```

```
    ret1, frame1=  pegaDaWebcam()
    print "leu da webcam"
    centrox = -1
    centroy = -1
    if ret1:
        imagemCinza = convertParaEscaladeCinza(frame1)
        cv2.imwrite("cinza.jpg",imagemCinza)
        print "escala de cinza"
        imagem_gamma = gammaCorrect(imagemCinza,0.35)
        print "gamma"
        cv2.imwrite("gamma.jpg",imagem_gamma)
        ret1, binary = binarizacao(imagem_gamma,1)
        cv2.imwrite("binaria.jpg", binary)
        print "binario"
        labeled = componentesConectados(binary)
        print "cc"
        MPL.imsave('componente.jpg',labeled)
        imagem = cv2.imread('componente.jpg',0)
        ret1, binaria = binarizacao(imagem,0)
        cv2.imwrite("binaria2.jpg", binaria)
        print "binario"
        centrox, centroy = calculaCentro(binaria)
        print centrox, centroy
    return centrox, centroy
```

Arquivo: **converterImagemFunc.py**

Arquivo possui funções auxiliares invocadas por vaias funções:

- **gammaCorrect**: aplica função gamma na imagem
- **binarizacao**: binarinariza a imagem passada por parâmetro. Se o parâmetro inv é 1, o valor maior que o threshold fica em branco e menor em preto.
- **convertParaEscaladeCinza**: converte a imagem para escala de cinza

```
from __future__ import print_function
import cv2
import numpy as np
import argparse
import scipy
import Image
import ImageFilter
import os,sys


def gammaCorrect(image, gamma):

    invGamma = 1.0/gamma
    table = np.array([((i/255.0)**invGamma)*255 for i in
np.arange(0,256)]).astype("uint8")
```

```
    im_gamma = cv2.LUT(image, table)
    return im_gamma

def binarizacao(imagemCinza,inv):

    if inv == 1:
        #ret1, binary =
cv2.threshold(imagemCinza,100,255,cv2.THRESH_BINARY_INV)
        ret1, binary = cv2.threshold(imagemCinza,0,255,
cv2.THRESH_OTSU + cv2.THRESH_BINARY_INV)
    else:
        ret1, binary = cv2.threshold(imagemCinza,0,255,
cv2.THRESH_OTSU)
    return ret1,binary

def convertParaEscaladeCinza(frame1):
    imgray = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
    return imgray
```

Função: **componentesConectados**

- Arquivo: **componentesConectadosFunc.py**
- Invocada por: **pegarPontosImagem_Lente**
- Implementação do método de componentes conectados, retorna o maior componente da imagem binária passada por parâmetro

```
from scipy import ndimage
import numpy as NP
import matplotlib.pyplot as MPL

def componentesConectados(binary):
    blur_radius = 1.0
    t = 50
    imgf = ndimage.gaussian_filter(binary,blur_radius)

    labeled, nr_objects = ndimage.label(imgf > t)

    array = NP.array(labeled).flatten().tolist()

    from collections import Counter

    c = Counter(array)

    num = c.values().index(max(c.values()[1:]))

    cols, rows = binary.shape
    for i in range(0,cols):
        for j in range(0,rows):
```

```
            if(labeled[i][j] == num):
                labeled[i][j] = 1
            else:
                labeled[i][j] = 0

    #MPL.imsave('centro.jpg',labeled)

    return labeled
```

Função: **calculaCentro**

- Arquivo: **retornaPontosFunc.py**
- Invocada por: **pegarPontosImagem_Lente**
- Encontra o centro geométrico da imagem utilizando como base o contorno da lente e um menor retângulo exinscrito ao contorno. Sendo possível calcular o centro da lente

```
import cv2
import math
import numpy as np

def calculaCentro(image):

    cv2.imwrite('imagem.jpg', image)
    im = cv2.imread('imagem.jpg')
    imgray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    ret, thresh = cv2.threshold(imgray,127,255,0)
    contours, hierarchy =
cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    x,y,w,h = cv2.boundingRect(contours[0].astype('int'))
    cv2.rectangle(im,(x,y),(x+w,y+h),(0,255,0),2)
    centrox = int(((2*x)+w)/2)
    centroy = int(((2*y)+h)/2)
    img = cv2.circle(im,(centrox,centroy), 4,(0,0,255),-1)
    cv2.imwrite('centro.jpg',im)

    print int(centrox), int(centroy)
    return centrox, centroy
```

Função: **pegaDaWebcam**

- Arquivo: **pegaDaWebcamFunc.py**
- Invocada por: **pegarPontosImagem_Lente**
- Lê um frame da webcam e retorna a imagem

```
import cv2
from scipy import ndimage
```

```
def pegaDaWebcam():

    cap = cv2.VideoCapture(0)
    ret1 = cap.set(3,320)
    ret1 = cap.set(4,240)
    ret1, frame1 = cap.read()
    frame1 = ndimage.rotate(frame1, 180)
    cv2.imwrite("captura.jpg", frame1)
    return ret1, frame1
```

Função: **motorAnda**

- Arquivo: **motorFunc.py**
- Invocada por: **lente**
- Recebe quantos passos o motor deve andar. A variável *motor* indica qual dos motores deve andar (1 para X e 2 para Y) e a *direction* indica a direção ao qual o motor indicado deve andar. O programa analisa se os dois botões foram pressionados informa à função lente para entrar no fluxo de parada de emergência.

```
import sys
import time
import RPi.GPIO as GPIO

#passoParameter = qtde de passos a andar
#motor = 1 para motor X
#        2 para motor Y
#direction = -1 para volta (no motor y... vai p direcao do motor)
#            1 para ida (no motor y... vai pra direcao da rodinha)

def motorAndar(passoParameter, motor, direction):

    print passoParameter, motor, direction

    GPIO.setmode(GPIO.BCM)

    #pinos para cada motor
    StepPins_X = [17,22,23,24]
    StepPins_Y = [14,15,18,10]

    botao_MotorY = 8
    botao_MotorX = 25
    if (motor == 1): #mexer o motor x
        StepPins = StepPins_X
    else:
        StepPins = StepPins_Y
    #print StepPins
```

```
    #Seta os pinos como pinos de saida

    for pin in StepPins:
        GPIO.setup(pin,GPIO.OUT)
        GPIO.output(pin, False)

    Seq = [[1,0,0,0],
           [1,1,0,0],
           [0,1,0,0],
           [0,1,1,0],
           [0,0,1,0],
           [0,0,1,1],
           [0,0,0,1],
           [1,0,0,1]]

    StepCount = len(Seq)-1
    StepDir = direction # Set to 1 or 2 for clockwise
                # Set to -1 or -2 for anti-clockwise


    if len(sys.argv)>1:
        WaitTime = int(sys.argv[1])/float(1000)
    else:
        WaitTime = 10/float(1000)

    StepCounter = 0
    passo = 0
    #variavel passo ira determinar quanto o motor vai girar.
    #necessario determinar o valor de "passo" para andar 1mm na webcam


    while passo <= passoParameter :

        if((GPIO.input(botao_MotorX) == False) &
(GPIO.input(botao_MotorY) == False)):
            passo = passoParameter
            print "emergencia"
            return 1



        #print passo
        for pin in range(0, 4):
            xpin = StepPins[pin]
            if Seq[StepCounter][pin]!=0:
                GPIO.output(xpin, True)
            else:
                GPIO.output(xpin, False)

        StepCounter += StepDir
```

```
        if (StepCounter>=StepCount):
            StepCounter = 0
        if (StepCounter<0):
            StepCounter = StepCount


        time.sleep(WaitTime)
        passo = passo + 1

    return 0
```

Função: **paradaEmergencial**

- Arquivo: **paradaEmergencialFunc.py**
- Invocada por: **lente**
- Quando os dois botões são pressionados ao mesmo no programa este entra no fluxo de desvio. Neste modo, dependendo de qual botão é pressionado um dos motores retorna um passo para a posição inicial dos eixos. Se os dois botões são pressionados novamente dentro deste modo, o programa é retornado à main, onde espera por uma nova lente.

```
from motorFunc import motorAndar
import RPi.GPIO as GPIO

def paradaEmergencial():

    botao_MotorY = 8
    botao_MotorX = 25
    jaSoltou = 0

    while True:
        if((GPIO.input(botao_MotorX) == False) &
(GPIO.input(botao_MotorY) == False) & (jaSoltou == 1)):
            while((GPIO.input(botao_MotorX) == False) |
(GPIO.input(botao_MotorY) == False)):
                doNothing = 1
            return
        elif((GPIO.input(botao_MotorX) == False) &
(GPIO.input(botao_MotorY) == True)):
            motorAndar(50,2,1)
        elif((GPIO.input(botao_MotorX) == True) &
(GPIO.input(botao_MotorY) == False)):
            motorAndar(50,1,-1)
        if((GPIO.input(botao_MotorX) == True) &
(GPIO.input(botao_MotorY) == True) & (jaSoltou == 0)):
            jaSoltou =1
```