UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UTFPR)
CURSO DE ENGENHARIA DE COMPUTAÇÃO

AUGUSTO DE OLIVEIRA ROSA
GUSTAVO INACIO PEREIRA GUEDES
JOÃO MATHIAS RAVAZZI MARTINS

ORGANIZADOR DE USO DE FICHAS DE LAVANDERIA

OFICINA DE INTEGRAÇÃO 1 – RELATÓRIO FINAL

CURITIBA

2019

AUGUSTO DE OLIVEIRA ROSA
GUSTAVO INACIO PEREIRA GUEDES
JOÃO MATHIAS RAVAZZI MARTINS

ORGANIZADOR DE USO DE FICHAS DE LAVANDERIA

Relatório Final da disciplina Oficina de Integração 1, do curso de Engenharia de Computação, apresentado aos professores que ministram a mesma na Universidade Tecnologica Federal do Paraná como requisito parcial para obtenção da aprovação na disciplina.

Orientador: Prof. Dr. Gustavo Benvenutti Borba
Prof. M.Sc. Ronnier Frates Rohrich

CURITIBA

2019

RESUMO

. ORGANIZADOR DE USO DE FICHAS DE LAVANDERIA. 19 f. Oficina de Integração 1 – Relatório Final – Curso de Engenharia de Computação, UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UTFPR). Curitiba, 2019.

Este documento descreve o Organizador de Uso de Fichas de Lavanderia, projeto realizado para a disciplina de Oficina de Integração 1. O projeto foi baseado na possibilidade de otimizar o processo de registro das fichas de lavanderia da Casa do Estudante Universitário do Paraná. Para tal, foi criado um dispositivo controlado pelo microcontrolador Arduino e por uma aplicação de python que remove as fichas de uma pilha e executa reconhecimento ótico de caracteres nelas individualmente, registrando os casos com sucesso em uma planilha. O processo tem sucesso na maioria das vezes, falhando aleatoriamente devido a condições ambiente ou leitura incorreta pelo motor de reconhecimento ótico de caracteres, ambos difíceis de controlar. Os conhecimentos adquiridos durante o desenvolvimento abrangeram diversas áreas, como visão computacional, reconhecimento ótico de caracteres, impressão 3D e uso de microcontroladores. Esta experiência, junto com os algoritmos desenvolvidos, pode servir de uso futuro em projetos com atributos semelhantes aos presentes aqui.

Palavras-chave: Registro de fichas de lavanderia. Dispositivo controlado por Arduino. Aplicação em python. Visão Computacional em python.

ABSTRACT

. LAUNDRY TOKEN USER MANAGER. 19 f. Oficina de Integração 1 – Relatório Final – Curso de Engenharia de Computação, UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ (UTFPR). Curitiba, 2019.

This document describes the Laundry Token User Manager, project put foward for the Integration Workshop 1 subject. The project was rooted in the possibility of optimizing the laundry token registration process in the University Student House of Paraná. In order to achieve this, a device controlled by an Arduino microcontroller and a python application was created, which removes tokens from a stack and runs optical character recognition on each one, registering successful readings in a worksheet. The process is mostly successful, with random failures caused by the environiment or incorrect reading from the optical character recognition engine, which are hard to keep control of. The knowledge acquired over the development covers many areas, like computer vision, optical character recognition, 3D printing and use of microcontrollers. The experience gained, alongside the developed algorithms, can be of future use in projects of similar attributes to the one presented here.

Keywords: Laundry token registration. Arduino-controlled device. Python application. Computer vision in python.

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Na Casa do Estudante Universitário do Paraná (CEUPR), para os moradores poderem utilizar-se das máquinas de lavar roupa e das secadoras disponibilizadas pela instituição, há um controle de uso com fichas, cujo formato e material se assemelha a moedas de 10 e 25 centavos. Cada ficha é etiquetada com o número do quarto e uma letra que representa qual dos dois moradores dorme naquele quarto. Semanalmente são recolhidas as fichas das máquinas, analisadas e registradas numa planilha, tudo feito manualmente.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

Criação de um sistema para checagem e registro em planilha das fichas de lavanderia de forma automática, onde o usuário insere a ficha e o valor é lido e registrado automaticamente.

1.2.2 OBJETIVOS ESPECÍFICOS

Construção um dispositivo microcontrolado por Arduino que:

- Empurre uma ficha para fora do compartimento de armazenamento;
- Guie-a para o compartimento de análise e efetue a leitura da informação presente;
- Separe as que puderam e as que não puderam ser lidas;
- Registre as informações em uma planilha.
- Seja gerenciado por uma aplicação de python.

2 HARDWARE

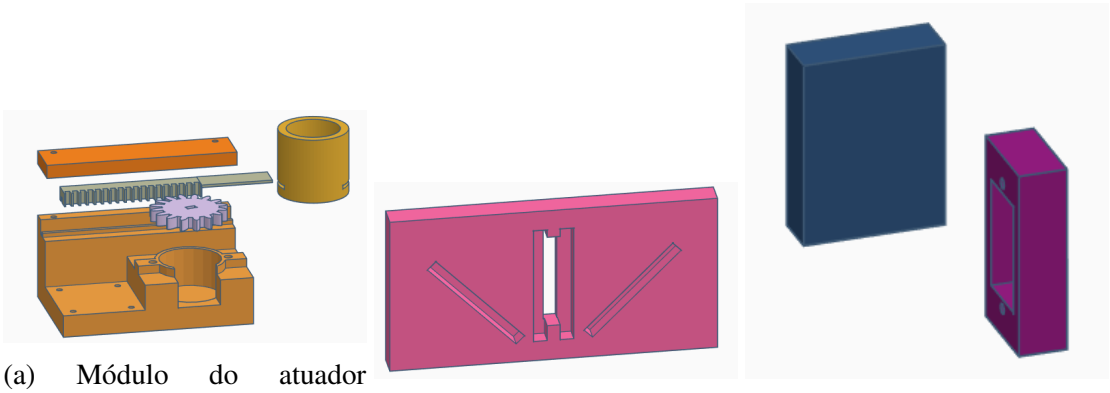
Todo o hardware do projeto está montado em uma estrutura de MDF: uma caixa retangular, com dimensões internas 20cm x 25cm x 30cm, um par de pés retangulares de 15cm de altura e uma tampa. A tampa tem um furo para a introdução das fichas e outros dois na face inferior da caixa, por onde elas saem. Dentro, alguns suportes de madeira permitem a fixação dos componentes na posição correta. A disposição dos componentes pode ser vista na Figura 2. Para que os mecanismos projetados funcionassem, alguns componentes foram modelados no Tinkercad[1] e impressos em 3D. Os modelos estão presentes na Figura 1. Além destes, foram usados os componentes listados na Tabela 1, arranjados como no esquemático da Figura 3.

Componente	Modelo/Especificações	Quantidade
Arduino	UNO	1
Cabo serial USB		1
Motor de passo	28BYJ-48	1
Driver para motor de passo	ULN2003	1
Módulo relé	SRD-05VDC-SL-C 5V 2 canais	1
Fonte externa	5V 2A	1
Servomotor	9g SG90	1
LED	Branco, alto brilho	2
Resistor	220Ω	2
Lâmina de vidro	Dimensões 3*cm*x3*cm*x3*mm*	2
Espelho	Dimensões 3*cm*x3*cm*x3*mm*	2
Webcam	Logitech C270	1

Tabela 1: Tabela de componentes eletrônicos e materiais e seus respectivos modelos ou especificações.

2.1 ENTRADA DE FICHAS

Alinhado com o furo na tampa, há um compartimento cilíndrico onde as fichas são introduzidas pelo usuário e armazenadas em uma pilha até o momento de sua leitura. O tubo apresenta duas aberturas em sua base: uma voltada para o atuador linear e outro para o escorregador à frente. A primeira é estreita, apenas grande o suficiente para que a ponta do



(a) Módulo do atuador linear ao lado do tubo de armazenamento

(b) Suporte para o compartimento de análise

(c) Separador e suporte do servomotor

Figura 1: Modelos 3D utilizados.

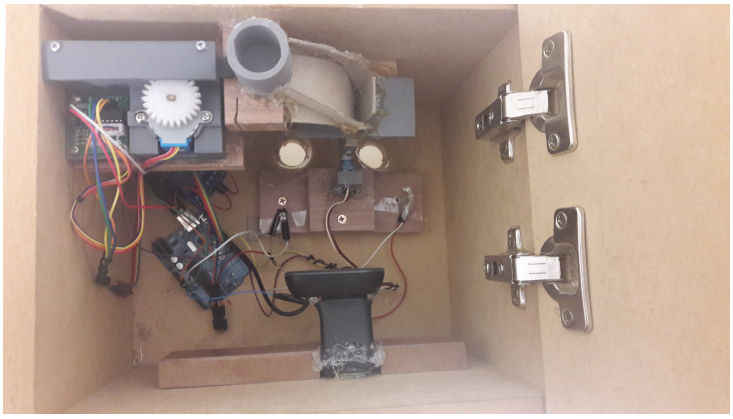


Figura 2: Visão aérea da caixa com todos os componentes.

atuador entre, empurre a ficha da pilha e saia. Já a segunda é mais larga, para que a ficha saia completamente do compartimento e caia na próxima seção. Esta abertura, porém, tem a altura quase exata de uma ficha, para que apenas uma por vez seja removida.

2.2 ATUADOR LINEAR

O módulo do atuador linear foi criado a partir do motor de passo citado na Tabela 1, do driver que realiza sua comunicação com o Arduino e das partes impressas que suportam o equipamento e transformam sua rotação em movimento linear. No motor está encaixada uma engrenagem que se comunica com uma cremalheira, formando o atuador linear em si. Quando o motor gira, a engrenagem empurra o trilho dentado para frente, que entra no tubo, empurra uma ficha e volta para a posição inicial.

O motor, ao contrário dos outros componentes, não é alimentado pelo Arduino, mas sim por uma fonte externa, devido à grande quantidade de corrente necessária para seu

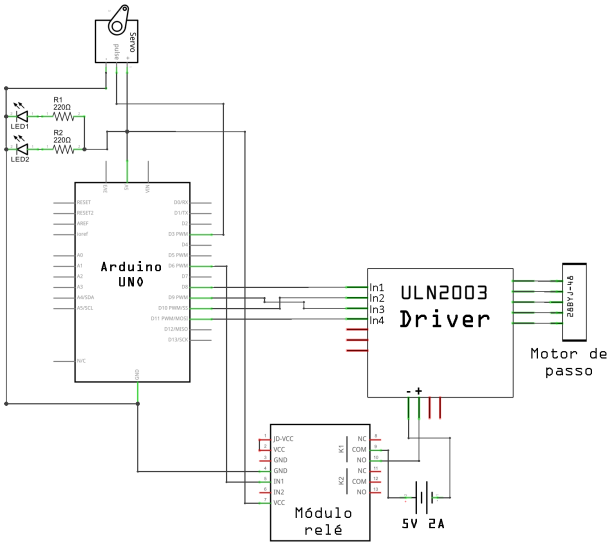


Figura 3: Esquemático dos componentes eletrônicos.

funcionamento. Para controlar esta fonte, foi usado um módulo relé controlado pelo Arduino, conforme a Figura 3. O pino correspondente está sempre emitindo sinal, de forma que o driver recebe corrente apenas enquanto o Arduino está ligado, este por sua vez controlando e alimentando o motor usando sinais recebidos das portas digitais do Arduino.

2.3 COMPARTIMENTO DE ANÁLISE

Quando removida do tubo, a ficha é guiada para o local onde a leitura ocorre. Isto é feito por um escorregador de papelão, o qual tem paredes, para guiá-la, e uma curva acentuada no final, para que ela passe o mais perpendicular possível pela entrada do compartimento e não trave. O compartimento de análise é composto pelas lâminas de vidro e espelhos mencionados na Tabela 1, apoiados no suporte para o compartimento de análise da Figura 1b. O suporte tem uma fenda no centro, para a passagem da ficha, duas posições para encaixar os vidros e duas para os espelhos. As lâminas de vidro delimitam o local em que a ficha fica, com as laterais fechadas com cola quente e aberturas na parte inferior e superior, e os espelhos fazem com que ambos os lados sejam vistos pela câmera, fixada no lado oposto da caixa. Para a iluminação, são usados dois LEDs de alto brilho associados em série com um resistor de 220Ω cada, posicionados para iluminar o melhor possível sem causar reflexos nos espelhos.

2.4 SEPARAÇÃO E SAÍDA DE FICHAS

Quando a ficha cai no compartimento, o que a impede de passar direto é o separador, o item retangular da Figura 1c. O separador é fixado no servomotor, que por sua vez é introduzido

no objeto vazado da Figura 1c. O servomotor, controlado por uma porta PWM do Arduino, gira o separador por um ângulo de 30°para a direita ou esquerda. A ficha escorrega e cai em um de dois funis, feitos com garrafas PET, que a direciona para um dos orifícios da parte de baixo da caixa. Dependendo do resultado da leitura, que pode falhar ou ter sucesso, o servomotor gira para um dos lados, levando-a a sair por um buraco específico, consequentemente separando as fichas que tiveram uma leitura válida das que não tiveram.

3 SOFTWARE

Para fazer todo o processo de analise de fichas, o software é dividido em dois, um para o microcontrolador do equipamento eletrônico e uma para um computador com sistema operacional Windows. Ambos se comunicam por conexão USB. A comunicação entre ambos começará a partir da solicitação do usuário através da sua interação com o programa de computador. O funcionamento segue o fluxograma da Figura 4

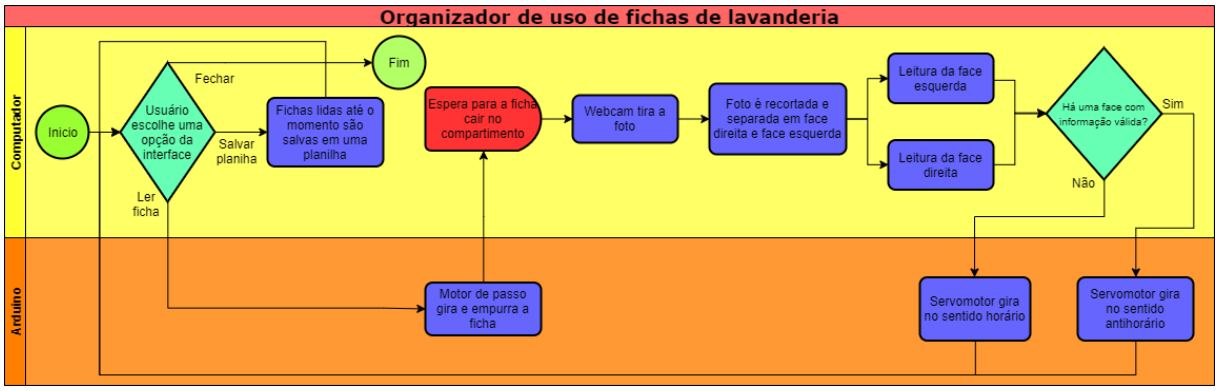


Figura 4: Fluxograma seguindo a trajetória da informação pelo sistema. O funcionamento detalhado está presente nesta seção.

3.1 ARDUINO

O Arduino foi utilizado para controlar o motor de passo e o servomotor, ambos para o controle do fluxo de fichas no dispositivo. Para tal, foram utilizadas, respectivamente, as bibliotecas padrão *servo* e *stepper*. No código do Arduino existem dois escopos principais: o *setup*, para inicialização do que será usado, e o *loop*, o laço principal que é executado até o desligamento do Arduino. No *setup*, além de inicializar as variáveis, o servomotor é ligado por padrão na posição 90°, na qual ele fica perpendicular ao chão.

No *loop*, apenas duas grandes tarefas são realizadas: girar o motor de passo e vericar informações do serial. Para a primeira tarefa, há a função *continuarMotor*. O motor de passo funciona com um ímã que é deslocado por bobinas posicionadas ao redor[2]. Ligando

as bobinas certas, o motor gira uma quantidade exata de unidades de ângulo, chamadas de passos. Para o controle preciso do processo foi utilizado um driver próprio, especificado na Tabela 1, que recebe sinal digital de quatro entradas predeterminadas para girar um número de passos pré-calculado. A função que gira o motor, primeiramente, checa se ele chegou à posição desejada, determinada previamente no *setup* ou por outra função. Se sim, a variável global que armazena seu ângulo é zerada. Caso alguma outra função tenha mudado este ângulo desejado, para um valor positivo ou negativo, o motor tentará chegar até ele girando exatamente um grau e incrementando ou decrementando a variável que registra sua posição atual. Assim, quando a função for chamada novamente, ele estará um pouco mais próximo do objetivo e dará mais alguns passos até ele, fazendo com que o motor gire aos poucos até o objetivo enquanto o Arduino processa outras tarefas. A função *empurrarMoeda* é uma das que determina a posição desejada para qual o motor deve girar, fazendo com que ele empurre a ficha da pilha.

Voltando ao *loop* principal, este também verifica a comunicação serial feita pela biblioteca do python Pyserial[3]. O código coleta o caractere fornecido pelo computador por meio do serial e compara-a com alguns casos previstos, realizando uma tarefa dependendo do caractere. Se o caractere for E, é chamado o método *empurrarMoeda* e o motor gira; se for R, L ou M, será chamado *servomotorgirar* com parâmetro 1, -1 ou 0, colocando-no na posição de 120°, 60°ou 90°respectivamente. Como o motor está posicionado para que 90°seja sua posição de equilíbrio, os comandos R e L efetivamente giram o motor para a direita ou esquerda.

3.2 COMPUTADOR

O software de computador tem uma Interface Gráfica do Utilizador (GUI), que coordena a interação do usuário com a comunicação com o Arduino, com o processamento de imagem e com o salvamento de planilha. Foi utilizado o paradigma orientado a objetos, que segmentou o código em quatro partes: Gerenciador Gráfico, Gerenciador do Arduino, Gerenciador de Imagem e Gerenciador da Planilha:

- O Gerenciador Gráfico é responsável pela interface e eventos presentes no sistema.
- O Gerenciador do Arduino cuida da comunicação com o Arduino, quando e como elas são feitas.
- o Gerenciador da Planilha transforma a lista de fichas lidas em uma planilha, também fazendo backup das imagens lidas já recortadas para futuras leituras, no caso de erros.

- O Gerenciador de Imagem faz todo o processamento de imagem para o reconhecimento dos caracteres presentes na ficha.

3.2.1 GERENCIADOR GRÁFICO

A GUI é escrita em Python utilizando a biblioteca TKinter[4] e foram utilizados somente objetos simples como botões, labels, edits e listbox. O usuário escolhe a porta em que o Arduino está conectado e clica no botão de conectar. Há um botão para iniciar a leitura de uma ficha, um para apagar a lista de informações registradas e outro para salvá-la em uma planilha. No centro, são vistas as fotos tiradas pela câmera e a lista de informações que o programa leu.

3.2.2 GERENCIADOR DO ARDUINO

A classe de gerenciamento do Arduino utiliza a biblioteca pyserial para se comunicar com o Arduino. A classe procura portas USB conectadas e possibilita conectar a interface e o microcontrolador, também recebendo e enviando informações de status deste. Ela também é responsável por fazer o atuador linear e o separador funcionarem, enviando informações para o motor de passo e o servomotor se moverem. Para identificar o status do Arduino e controlar as tarefas, é enviado um sinal para o mesmo ser avisado que o programa de computador quer saber seu status, fazendo o Arduino enviar um caractere que representa o que está sendo executado. Os caracteres são explicados na Tabela 2.

Caractere	Significado
S	Solicitação de Status
P	Atuador pronto para empurrar
T	Ficha foi empurrada

Tabela 2: Tabela de caracteres que podem ser enviados pelo Arduino e seu respectivo significado.

3.2.3 GERENCIADOR DE IMAGEM

A classe GerenciadorImagem é a responsável pelo uso da câmera e pela visão computacional. A principal biblioteca usada é o OpenCV 4[5], uma biblioteca open source de visão computacional, implementada em C++ mas utilizada em Python neste software devido

à otimização e à maior facilidade de uso. Também é usado o NumPy[6], o qual tem funções matemáticas úteis e é usado extensivamente pela implementação em Python do OpenCV.

A classe possui dois métodos relacionados à câmera, chamados *getListaCameras* e *setIndiceCamera*, criados por causa do modo com que o OpenCV lida com câmeras. Quando um objeto do tipo câmera é criado, é necessário escolher um índice para que o equipamento correto seja localizado, o qual só pode ser localizado automaticamente ou por tentativa e erro, ou por implementações mais complexas. Por simplicidade, o método *getListaCameras* testa alguns índices que podem pertencer a câmeras e retorna os existentes. Já o *setIndiceCamera* inicializa um atributo com o índice escolhido por quem está utilizando a classe, dizendo ao objeto qual câmera usar. O restante dos métodos são relacionados ao processo de leitura, explicado passo-a-passo a seguir.

3.2.3.1 FOTOGRAFIA

Antes de iniciar a leitura, a classe principal chama *tirarFoto* para fotografar as fichas. A foto é recortada em duas regiões de interesse, determinadas durante o desenvolvimento do projeto com o auxílio do código criado por pysource[7]. O recorte é feito por *slicing*, já que toda imagem do OpenCV é armazenada como um vetor do NumPy, o qual pode ser tratado como uma lista multidimensional. Os recortes são espelhados, salvos em uma pasta e retornados para serem usados no método *getQuarto*, onde está encapsulado o processo de leitura. As duas imagens recortadas, cada uma contendo uma das faces da ficha, são lidas em paralelo. Enquanto o lado esquerdo é lido na thread principal pelo método *lerImagem*, o direito é repassado para *threadedLeitura*, que chama o mesmo método e o coloca em uma fila para ser coletado novamente na thread principal. Para tal, são usados os módulos threading e queue.

3.2.3.2 LOCALIZAÇÃO E RECORTE DO TEXTO

O primeiro desafio para identificar o texto é saber onde ele está. Para isso, é usado o fato que a informação das fichas está contida em um retângulo vermelho, então a localização almeja identificar este retângulo e alinhá-lo para que seja lido. Isto é feito convertendo a imagem BGR (Blue-Green-Red) para HSV (Hue-Saturation-Value), explicado na Figura 5. A cor vermelha de interesse está localizada nas faixas de Hue 0 a 20 e de 144 a 179, então são aplicadas duas máscaras nestas faixas, as quais são somadas, criando uma imagem binária: branco onde o valor de Hue está dentro da faixa especificada e preto onde não está, como na Figura 6b.

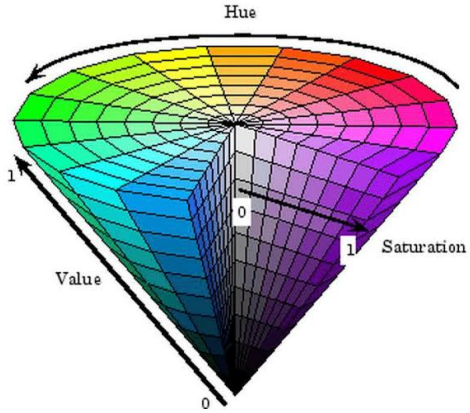


Figura 5: Visualização do espaço de cor HSV. O valor Hue é o que interessa, pois determina exatamente a cor do pixel, independente de saturação ou iluminação. Fonte: [9].

Nesta imagem, são encontrados os contornos[8]. Contornos são linhas que delimitam uma região contínua, e aqui são usados para encontrar o retângulo branco da máscara. O maior é selecionado e é usado sobre ele o método *minAreaRect* do OpenCV, que determina o menor retângulo possível que contenha o contorno, visto na Figura 6c sobre a imagem original. Este retângulo é exatamente a área que contém o texto. O processo pode ser visto na Figura 6.

(a) Foto original (b) Máscara sobre os píxeis vermelhos (c) Retângulo sobreposto na imagem original

Figura 6: Exemplo de fotografia tirada pelo dispositivo(6a) e a máscara encontrada a partir dos píxeis com valor Hue dentro do especificado(6b). A partir dela é encontrado o retângulo com texto(6c).

A variável que armazena o retângulo contém três informações: posição, área e ângulo em relação ao eixo X. Como o ângulo pode ser tanto em relação ao maior quanto ao menor lado, é verificado se a altura é maior que a largura, invertendo-as em caso positivo e adicionando 90°ao ângulo, assim garantido que o retângulo fique horizontal. Após a checagem, a imagem original sofre um desfoque gaussiano, que remove ruídos pequenos, é rotacionada com o método do OpenCV *warpAffine* e é recortada na região delimitada pelo retângulo encontrado. Assim, o programa tem uma imagem contendo exatamente a região com a informação, como na Figura 7.

(a) Imagem desfocada (b) Imagem rotacionada (c) Retângulo recortado

Figura 7: Imagem desfocada(7a) rotacionada a partir do ângulo do retângulo(7b) e a área recortada da original(7c)

3.2.3.3 LEITURA

Se a imagem fosse enviada para leitura neste momento, ainda haveria ruído demais para o texto ser reconhecido. Por isso, é preciso tratar a imagem um pouco mais e isolar exatamente o texto. Para fazer isto, os canais de cor da imagem são separados em B, G e R. Como a região recortada apresenta apenas o fundo vermelho, com R alto, e texto preto, com R baixo, apenas este canal já é suficiente para destacar o texto, o que pode ser visto na Figura 8a.

A imagem então sofre uma limiarização adaptativa[10] gaussiana, feita pelo OpenCV, que transforma um pixel em branco ou preto dependendo dos seus vizinhos, como na Figura 8b. As regiões brancas vistas são dilatadas[11] bem pouco, o que corrige alguns erros de descontinuidade de tais regiões. A imagem resultante, vista ao final da Figura 8, é então passada para o método *leitura*, que é responsável por isolar os caracteres. Como o texto pode estar de ponta cabeça, é criada uma cópia invertida verticalmente para ser analisada também.

(a) Canal vermelho (b) Limiarização adaptativa (c) Pequena dilatação

Figura 8: O canal vermelho da Figura 7c é separado(8a), limiarizado(8b) e dilatado(8c).

Mais uma vez, são procurados contornos na imagem binarizada. Estes irão encontrar o texto, mas também manchas criadas pela limiarização. Por isso, apenas os quatro de maior área são selecionados, os quais são quase sempre os quatro caracteres da ficha. Um retângulo contendo estes quatro contornos é desenhado e recortado da imagem. Assim, o programa tem a imagem da Figura 9, contendo exatamente a informação da ficha, binarizada e legível pela máquina.

O método *OCR* é o responsável pelo Reconhecimento Ótico de Caracteres, ou OCR.

Figura 9: Imagem final enviada para OCR. Note que esta é a cópia invertida, pois a imagem original, vista na Figura 8c, está de ponta cabeça.

Para tal, é usado o Tesseract[12], rede neural desenvolvida pela Google para realizar OCR. O Tesseract apresenta treze configurações para diferentes tipos de texto, como em círculo, várias linhas ou várias palavras. Cada configuração identifica caracteres diferentes, e testes mostraram que as condições das imagens usadas aqui não permitem determinar de antemão qual delas é melhor. Por isso, são testadas as oito mais prováveis de serem a adequada. Cada uma é armazenada em uma lista e analisada pelo padrão número-número-número-letra, retornando-a se passar no teste. Foi visto experimentalmente que, às vezes, há a troca de alguns números por letras, como 2 por Z e 8 por B, o que pode ser uma fonte de erros.

A informação final é levada de volta ao método *getQuarto*, que compara o que foi lido em ambos os lados da ficha. Se houve exatamente uma informação válida, esta é retornada. Caso contrário, é retornado um erro. No caso usado de exemplo nesta subseção, o resultado obtido foi `104B`.

3.2.4 GERENCIADOR DE PLANILHA

A classe GerenciadorPlanilha, além de um método auxiliar de determinação da data e hora atuais, tem apenas o método *criarPlanilha*. Este método recebe como parâmetro uma lista de elementos lidos, que será criada durante a leitura das fotografias das fichas. Dentro, ele cria uma outra lista de quartos, a qual contém todos os quartos que espera-se verificar. Após isto, cria-se um dicionário, associando-se espaços em branco a todos os quartos da lista, e compara-se quais quartos estão na lista dos lidos. Se eles estiverem, escreve-se um ”X”no lugar do espaço em branco que estava anteriormente.

Quando o usuário pedir, por meio da interface, é gerada a planilha utilizando a biblioteca xlsxwriter[13]. Primeiramente é criado um *workbook*, o arquivo da planilha, utilizando o método chamado *workbook*. Então, adiciona-se uma *worksheet* ao *workbook*, que é tabela da planilha, utilizando o método *addworksheet* do xlsxwriter. Finalmente, cada informação do dicionário é escrita, com o número do quarto na primeira coluna e seu x ou espaço em branco na segunda. Terminando este processo, a planilha é fechada e salva com o nome no formato ano-mês-dia-horário.

4 RESULTADOS

Os resultados obtidos de testes com a versão final do projeto foram satisfatórios. No compartimento de armazenamento, o atuador linear consegue remover a ficha completamente e apenas uma por vez. A ficha também tem sucesso em escorregar até o compartimento e cair nele em uma posição onde a câmera consiga capturar ambos os seus lados pelos espelhos.

Já a leitura, devido à sua natureza complexa, não funciona sempre. O principal fator é a iluminação, que pode deixar a imagem escura demais ou ilegível com os reflexos. Os LEDs de iluminação foram posicionados cuidadosamente para reduzir ao máximo estes dois problemas, porém ainda assim podem ocorrer.

O último obstáculo é a própria rede neural para OCR, que pode reconhecer caracteres erroneamente. Ainda assim, testes de leitura tiveram sucesso aproximadamente 73% das vezes, baseado em uma amostra de 30 tentativas, significativamente melhor que os resultados obtidos no início do desenvolvimento do projeto, os quais eram quase sempre menores que 50%. Já a separação das fichas não apresenta problemas, com elas sempre sendo direcionadas para o reservatório correto.

5 CONCLUSÃO

Como já foi citado, houve dificuldade em coordenar a claridade dentro do dispositivo, o que prejudicou a eficiência da leitura. Em caso de tentativa de melhora do projeto, seria melhor utilizar equipamentos profissionais, como câmeras de melhor definição, lentes anti reflexo e luz difusa, os quais ajudariam na leitura da ficha. Porém, mesmo com alguns problemas imprevisíveis e difíceis de resolver que surgiram nas últimas etapas de execução, foi possível concluir o projeto com resultados satisfatórios. Além disso, os autores finalizaram o projeto com uma bagagem de conhecimentos novos que foram necessários para a execução, como a linguagem de programação Python, visão computacional com OpenCV, OCR, interface gráfica com TKinter e criação de arquivos .xlsx com xslxwriter, o que pode ser aproveitado no futuro. Também houve uma atenção especial para a documentação desenvolvido o trabalho em equipe, organização e conhecimentos sobre documentação dos autores.

Foi produzido um vídeo sobre o projeto.[14]

REFERÊNCIAS

[1] Autodesk tinkercad. https://www.tinkercad.com/.

[2] DEJAN. How a stepper motor works. https://howtomechatronics.com/how-it-works/electrical-engineering/stepper-motor/.

[3] pyserial. https://pyserial.readthedocs.io/en/latest/.

[4] Graphical user interface with tk. docs.python.org/3/library/tk.html.

[5] Opencv - open source computer vision library. https://opencv.org/.

[6] Numpy. https://numpy.org/.

[7] PYSOURCE. Mouse events - opencv 3.4 with python 3 tutorial 27. https://pysource.com/2018/03/27/mouse-events-opencv-3-4-with-python-3-tutorial-27/.

[8] Contours in opencv. https://docs.opencv.org/master/d3/d05/tutorial_py_table_of_contents_contours.html.

[9] Image processing toolbox - hsv color space. https://edoras.sdsu.edu/doc/matlab/toolbox/images/color11.html#18256.

[10] Opencv - image thresholding. https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html.

[11] Opencv - eroding and dilating. https://docs.opencv.org/master/db/df6/tutorial_erosion_dilatation.html.

[12] Tesseract open source ocr engine. https://github.com/tesseract-ocr/tesseract.

[13] xlsxwriter. https://xlsxwriter.readthedocs.io/.

[14] Organizador de uso de fichas de lavanderia. https://www.youtube.com/watch?v=wjYlIn-IDVE.