Capítulo 2

Acionamento de Dispositivos Eletroeletrônicos Utilizando Visão Computacional

Marcos Melo Ferreira, Caio Eduardo Falcão Matos

Abstract

Computer Vision has been widely used in developing industry, medicine, automation, and public safety tools. Due to the current relevance of the topic, a mini-course on the theme "Acionamento de Dispositivos Eletroeletrônicos Utilizando Visão Computacional"was accepted to be presented at the tenth Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI), held between September 28 and 30, 2022 in São Luis Maranhão. This mini-course aims to present a method that allows computer vision techniques to control electronic devices. The mini-course is expected to arouse participants' interest in developing projects in which knowledge in Artificial Intelligence and Electronics is used.

Resumo

A Visão Computacional tem sido amplamente utilizada no desenvolvimento de ferramentas utilizadas na indústria, medicina, automação e segurança pública. Devido a relevância atual do tema foi proposto o minicurso "Acionamento de Dispositivos Eletroeletrônicos Utilizando Visão Computacional", aceito para ministração na décima Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI), realizada entre os dias 28 e 30 de Setembro de 2022 em São Luís, Maranhão. Este minicurso tem como objetivo principal apresentar um método que possibilite a utilização de técnicas de visão computacional para controle de dispositivos eletroeletrônicos. Espera-se que o minicurso desperte o interesse dos participantes em desenvolver projetos em que sejam empregados conhecimentos das áreas de Inteligência Artificial e Eletroeletrônica.

2.1. Introdução

Recentemente, a Inteligência Artificial tem sido amplamente empregada por meio de diversas aplicações em diferentes áreas, que vão desde a criação de ferramentas para auxílio de diagnóstico de doenças por meio exames de imagem [Ferreira et al., 2020],

[Shang et al., 2019], [Raman et al., 2019] a mecanismos de análise de sentimentos de usuários de redes sociais [Zhang et al., 2018] e detecção de *fake news* presentes em textos postados diariamente na Internet [Kaliyar et al., 2021]. Uma das sub-áreas da Inteligência Artificial é a Visão Computacional, que estuda métodos para processamento de imagens do mundo real por um computador. Por meio da visão computacional é possível detectar e diferenciar objetos presentes em uma imagem, classificar exames de imagem, determinando quais possuem maior probabilidade de pertencerem a um paciente que está acometido por algum tipo de doença. Uma das diversas possibilidades de aplicações, é o controle de dispositivos eletroeletrônicos por meio da visão computacional. O controle é realizado pela detecção de mãos, presentes em uma imagem capturada por meio de uma webcam. Caso alguma mão seja detectada, o computador irá calcular parâmetros que serão enviados a uma placa eletrônica responsável por acionar os dispositivos. Utilizando esta tecnologia pode ser possível desenvolver aplicações de automação residencial e de tecnologias assistivas. Deste modo é interessante que alunos de cursos da área de Tecnologia da Informação sejam apresentados e possuam noções sobre Inteligência Artificial e uma de suas sub-áreas, a Visão Computacional.

2.2. Visão Computacional

A visão computacional é uma das áreas da Inteligência Artificial em que mais pesquisas têm sido feitas, com o propósito de cada vez mais elevar a acurácia de métodos que possibilitem às máquinas a capacidade de interpretar visualmente informações, ou seja, enxergar. Esta capacidade de enxergar significa que as máquinas podem transformar dados de uma imagem ou de um vídeo em uma nova representação, utilizada para que algum objetivo seja alcançado, como detecção e identificação de objetos presentes em uma imagem [Kaehler and Bradski, 2016]. Essa capacidade possibilitou o desenvolvimento de aplicações em diferentes áreas, como na indústria onde são realizadas verificações não invasivas em embalagens e inspeção de maquinário e estruturas, e na medicina, onde informações são extraídas de exames de forma automática para auxiliar os médicos na obtenção de diagnósticos de forma mais rápida e precisa. Como é um campo que tem atraído a atenção de pesquisadores devido ao grande número de possibilidades de aplicação, diversas tecnologias vem sendo desenvolvidas rapidamente.

A realização do processamento de imagens consiste inicialmente na extração de características das imagens. Estas características são utilizadas por algoritmos de aprendizado de máquina, que podem ser configurados para obtenção de coordenadas de *bounding boxes*, classificação das imagens, segmentação semântica de um alvo, por exemplo veias presentes em olhos, dentre outros [Prince, 2012]. Anteriormente, métodos manuais (*handcrafted*) eram utilizados para obtenção das características. Recentemente, métodos baseados em redes neurais alcançaram resultados superiores nas tarefas de processamento de imagens. Estas redes, especificamente as convolucionais, conseguem realizar a extração de características de forma automática, e melhoram este processo durante o seu treinamento, onde ocorre o aprendizado de quais características são mais relevantes para que determinada tarefa seja realizada.

2.3. Redes Neurais Convolucionais

As redes neurais convolucionais (CNN, do inglês *Convolutional Neural Network*) são um modelo de rede neural profunda. Enquanto redes neurais simples são formadas por poucas camadas, redes profundas são formadas por várias camadas, uma camada de entrada, uma de saída e várias camadas escondidas. Outra característica deste modelo de rede é ser do tipo *feedforward*, ou seja, não existe realimentação. Os dados percorrem a rede em uma única direção, da entrada para a saída [Goodfellow et al., 2016].

A principal características deste modelo é a realização da operação matemática denominada convolução. (Equação 1).

$$s(t) = (x * w)(t) \tag{1}$$

onde x representa o valor de uma variável em um instante t, e w representa uma função de ponderação. Em redes convolucionais, o primeiro argumento da equação refere-se a um dado de entrada, por exemplo os pixels que formam uma região de uma imagem, e o segundo refere-se a um filtro (*kernel*). O resultado é geralmente denominado mapa de características (*feature maps*). Redes Convolucionais são redes neurais que utilizam a operação matemática convolução em pelo menos uma de sua camadas [Goodfellow et al., 2016]. Estas camadas são denominadas convolucionais, e fazem parte do extrator de características visuais. Como os computadores utilizam dados discretos, pode-se definir a convolução discreta como sendo a Equação 2.

$$s(t) = (x * w)(t) = \sum_{a=0}^{N} x(a)w(t-a) \tag{2}$$

onde x e w serão definidos apenas para valores inteiros de t.

A Figura 2.1 ilustra a operação de convolução entre uma região de uma imagem e um filtro. Esta região da imagem é chamada de campo receptivo local [Academy, 2019]. Este campo receptivo é deslizado por toda a imagem, a um passo medido em *pixels* denominado *stride length*. Apesar da figura ilustrar a convolução sendo realizada com os pixels variando entre 0 e 255, antes do cálculo acontece uma normalização. Cada pixel é normalizado para um valor entre zero e um.

Em redes neurais que utilizam multiplicação de matrizes, a saída de um neurônio é utilizado como entrada para todos os neurônios da próxima camada [Haykin, 1999]. Como essa conectividade "total" entre neurônios não acontece em uma camada convolucional, essas redes possuem interações esparsas. Isso acontece devido ao filtro possuir dimensões menores que a entrada [Goodfellow et al., 2016], como representado na Figura 2.1. Com isso é necessário a realização de menos operações matemáticas diminuindo o custo computacional. Através da convolução é possível detectar pequenas, porém relevantes características que serão utilizadas para classificação das imagens. Além disso, em uma rede convolucional existe o compartilhamento de pesos. Um mesmo filtro é utilizado para realizar a convolução em toda a imagem, o que não acontece em uma rede totalmente conectada, onde existe um peso para cada conexão entre o neurônio de uma camada e os neurônios da camada seguinte. Isso implica em dizer que existem menos parâmetros para serem analisados e ajustados, o que diminui o custo computacional. Outra vantagem

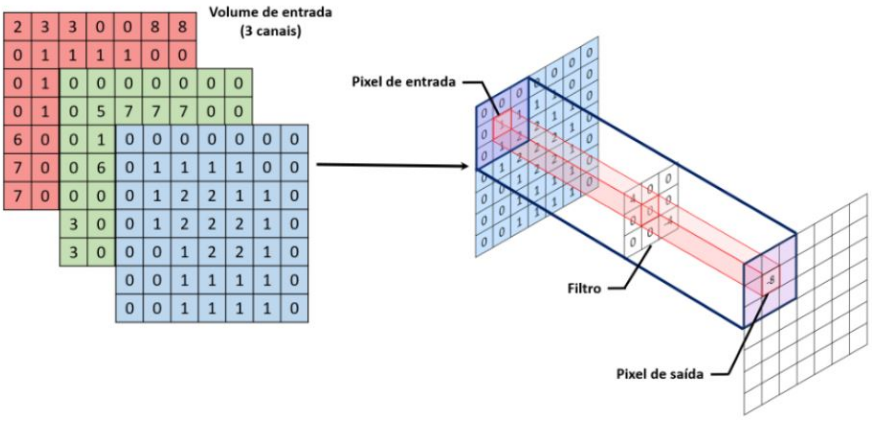

Figura 2.1. Convolução Discreta. [Araújo et al., 2017]

dessas redes é o fato de necessitarem de pouco pré-processamento. Enquanto em outros métodos o design dos filtros é feito por algum especialista, com o treinamento necessário, as Redes Convolucionais são capazes de aprender a ajustar os filtros [Saha, 2018].

As redes convolucionais possuem, além de camadas que realizam a operação de convolução, camadas que realizam uma operação denominada *pooling*. Em geral, o extrator de características deste tipo de rede é formado por blocos que contém duas ou mais camadas de convolução seguidas de uma camada de *pooling*. Esta função substitui a saída de uma camada por um índice estatístico. Por exemplo, a operação *Max Pooling*, predominantemente utilizada, reporta a saída máxima dentre as saídas existentes em uma vizinha retangular [Goodfellow et al., 2016]. Assim como a camada de convolução, a camada de *Max Pooling* também reduz o tamanho de uma imagem, uma vez que somente retorna um valor de uma determinada região da imagem de entrada. Além disso, a utilização do *Max Pooling* produz invariância à pequenas translações da imagem de entrada. Mesmo que haja alguma variação deste tipo, sendo que são imagens de um mesmo objeto, essa invariância evita que a rede classifique as variações de uma mesma imagem como imagens diferentes.

Em CNNs utilizadas para classificação de imagens, após os blocos formados pelas camadas de Convolução/Max Pooling, que tem por finalidade extrair características de imagens de entrada, são utilizadas camadas que irão utilizar estas características para gerar o resultado final da classificação. Os classificadores dos modelos são formados por camadas totalmente conectadas, denominadas camadas densas, exatamente iguais a uma rede MLP (do inglês, *Multi Layer Perceptron*) [Araújo et al., 2017]. Isto significa que todos os neurônios de uma camada estão conectados aos neurônios da camada seguinte [Goodfellow et al., 2016]. Em termos matemáticos, um neurônio pode ser descrito pelas Equações 3 e 4

$$h_j = \sum_{i=1}^{m} w_{ij} x_i \tag{3}$$

$$y_j = \varphi(h_j + b_j) \tag{4}$$

onde x_i são os m sinais de entrada, w_{ij} são os pesos sinápticos do neurônio j, e b_j corresponde ao *bias*, responsável por realizar o deslocamento da função de ativação definida por φ [Araújo et al., 2017].

Para que seja possível utilizar os mapas de características como entrada do classificador é necessário alterar o formato matricial dos dados para vetorial. Esse processo é denominado *flattening*. Em geral, as primeiras camadas densas utilizam a função ReLu (*Rectified Linear Unit*) Equação 5, como função de ativação, e a última camada do classificador utiliza a função de ativação *softmax* [Bishop, 2006], definida pela Equação 6

$$f(x) = max(0,x) \tag{5}$$

$$\sigma(Z)_i = \frac{e^{zi}}{\sum_{j=1}^{K} e^{zj}} \tag{6}$$

onde z é um vetor de números reais que é normalizado em uma distribuição de K probabilidades. Esta função tem como saída uma distribuição de probabilidades, que é utilizada para classificação das imagens, e o número de neurônios da camada é igual ao número de classes em que as imagens podem ser classificadas. Outro parâmetro importante dos classificadores é o percentual de *dropout*, que consiste na remoção aleatória de um percentual de neurônios a cada iteração de treinamento, com o propósito de evitar *overfitting* e reduzir o tempo de treinamento [Goodfellow et al., 2016]. A arquitetura básica de uma CNN utilizada para classificação de imagens é ilustrada na Figura 2.2.

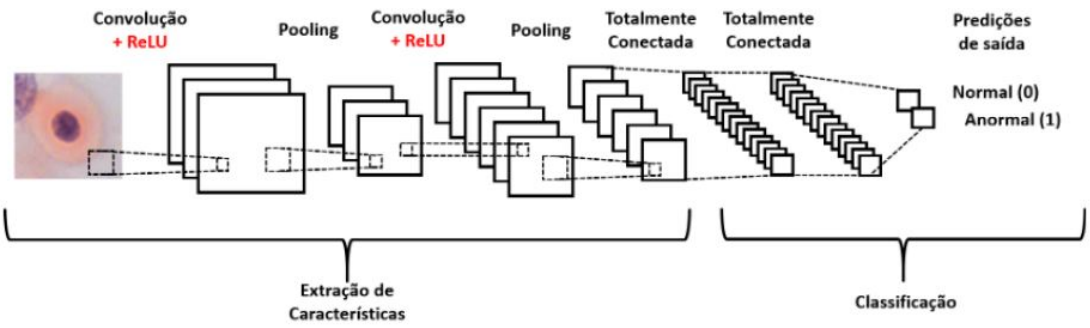

Figura 2.2. Arquitetura de uma rede convolucional. [Araújo et al., 2017].

2.4. Detecção de Objetos

A classificação de imagens somente faz sentido para imagens que possuem apenas um objeto, podendo serem citados os exames que contém imagens de um determinado órgão do corpo. Para o caso de imagens que possuem vários objetos é necessário localizar e classificar cada objeto presente na imagem. Esta tarefa é denominada detecção de objetos. Modelos computacionais utilizados para esta finalidade têm como saída não apenas a probabilidade de um objeto pertencer a uma classe, mas também as quatro coordenadas espaciais que serão utilizadas para traçado da caixa delimitadora (*Bounding Box*), retângulo que é utilizado para ilustrar a localização de um objeto em uma imagem. Deste modo, o

modelo computacional terá que prever a classe do objeto, coordenadas do canto superior esquerdo, altura e largura do retângulo. Uma forma comum de lidar com este problema consiste no uso de janelas deslizantes. Para realizar a detecção, as imagens são percorridas por janelas de tamanho fixo, que servem de entrada para um classificador. Algumas abordagens têm sido empregadas para este fim, como a Detecção de Objetos utilizando Histograma de Gradientes Orientados [Dalal and Triggs, 2005] e as Redes Neurais Convolucionais. Uma vez que o custo computacional das redes neurais profundas é maior que o de outras abordagens de aprendizado de máquina, é inviável alimentar uma rede com todas as regiões de uma imagem, com escalas diferentes. Sendo assim foram propostas as Redes Neurais Convolucionais Baseadas em Regiões (R-CNN) [Girshick et al., 2015], que trouxeram uma solução denominada busca seletiva (*Selective Search*), onde um algoritmo realiza uma busca na imagem para encontrar regiões com maior probabilidade de conter objetos, diminuindo o número de bounding boxes que serão utilizados como entrada para a rede neural. No entanto, o processo de detecção com CNNs ainda era lento, sendo que outras abordagens que tornaram o processo mais rápido foram propostas por [He et al., 2015], [Girshick, 2015], [Ren et al., 2015].

Outra abordagem existente para o solução do problema de detecção foi proposta por [Redmon et al., 2016], que apresentou a Rede Yolo (do inglês, *You Only Look Once*). Esta abordagem difere das anteriores porque a detecção não é baseada em regiões da imagem, logo não existe o *pipeline* para escolha de regiões da imagem que possuem um maior probabilidade de existência de objetos. Ao contrário das outras redes, a YOLO recebe como entrada a imagem completa e a divide em um *grid* de tamanho SxS. A rede então prevê os *bounding boxes* e coeficiente de confiança (*confidence score*) para cada um, sendo que para a maioria este valor será muito baixo e o bouding box será descartado. Além disso, a rede também prevê a probabilidade de cada divisão do *grid* pertencer a uma determinada classe. Devido ao fato da rede ter como entrada apenas uma imagem a detecção é realizada de forma muito rápida e pode ser executada em tempo real. No entanto, como a rede prevê apenas um tipo de classe para cada divisão do *grid*, esta rede apresenta como limitação a dificuldade de detectar objetos muito pequenos. A rede é formada por camadas de convolução, geralmente sendo escolhida alguma CNN utilizada para classificação de imagens como *backbone*, seguida de duas camadas totalmente conectadas (*fully connected*). A arquitetura e o formato da saída da rede são mostrados na Figura 2.3.

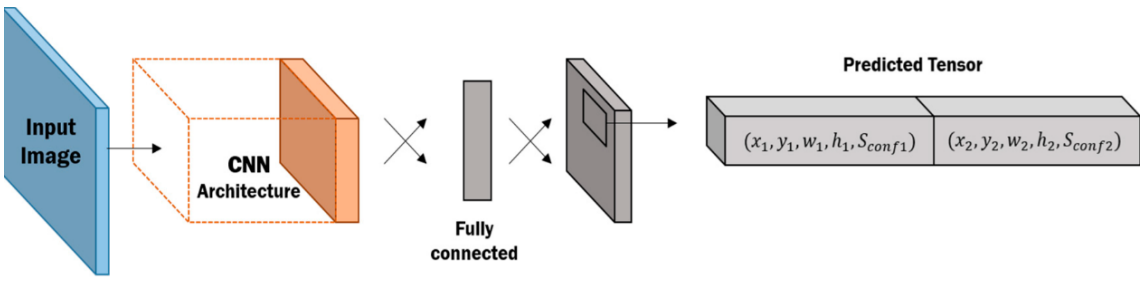

Figura 2.3. Arquitetura da Rede YOLO. [Redmon et al., 2016].

2.5. Placa Arduino Uno

A placa microcontroladora Arduino tem sido amplamente utilizada para o desenvolvimento de diversas aplicações em áreas como automação, robótica e tecnologias assistivas [Kondaveeti et al., 2021]. A placa pode ser entendida como um pequeno computador que pode ser programado para processar entradas e saídas, que são componentes externos conectados a ele [McRoberts, 2018]. Devido à uma maior facilidade para utilização e construção de protótipos, quando comparada à outras tecnologias, esta plataforma passou a ser bastante utilizada por professores e pesquisadores, e mais recentemente por profissionais de áreas citadas anteriormente. Existem diversos tipos de Arduino, sendo um dos mais utilizados, o Arduino Uno, que tem o microcontrolador ATmega328P produzido pela empresa Atmel. A placa, Figura 2.4, possui catorze pinos de E/S (Entrada e Saída), sendo seis que podem realizar a leitura de sinais analógicos e seis com função PWM (do inglês, *Pulse Width Modulation*), porta para comunicação serial, pino para alimentação, dentre outros periféricos.

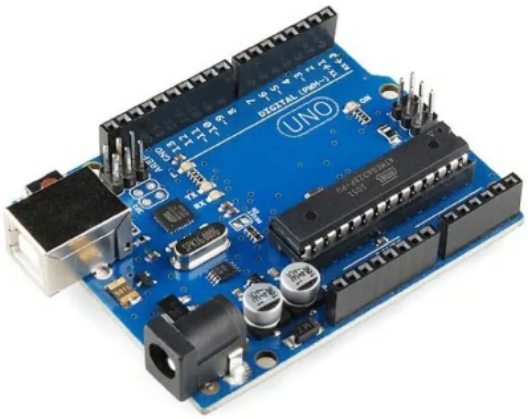

Figura 2.4. Placa Arduino Uno.

Uma das grandes vantagens da utilização de placas Arduino é a existência de módulos de expansão, que em geral são compatíveis com qualquer tipo de placa com necessidade de pouca ou nenhuma adaptação. Existem diferentes módulos, desde sensores de presença, aferição de temperatura e umidade e sensores ultrassônicos, diversos tipos de display, módulos para comunicação via protocolo TCP/IP, dentre outros. Estes módulos facilitam o desenvolvimento de aplicações um pouco mais complexas, que necessitam de sensores ou atuadores específicos. A Figura 2.5 mostra alguns exemplos de módulos de expansão, em sua maioria sensores digitais ou analógicos, juntamente com uma placa Arduino Uno.

2.6. Experimentos

Os experimentos utilizados neste minicurso têm como objetivo principal apresentar uma possibilidade de desenvolvimento de aplicações que utilizem de forma integrada, conceitos de visão computacional e de eletrônica, tendo em vista que grandes avanços têm sido alcançados recentemente nesta área da inteligência artificial, e que a mesma vem sendo cada vez mais explorada para desenvolvimento de produtos que possam ser utilizados

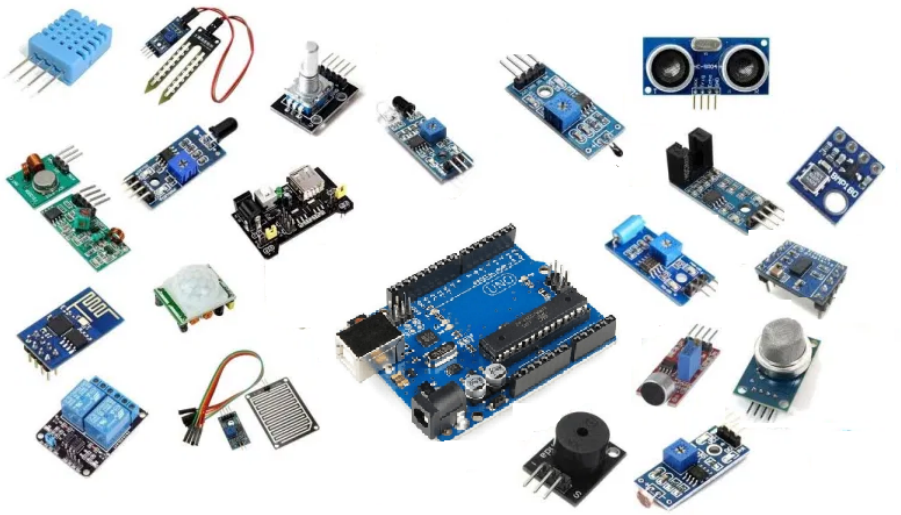

Figura 2.5. Módulos de Expansão.

para trazer melhorias à vida das pessoas, aumento de produtividade de empresas, dentre outros.

Os experimentos apresentam uma forma que possibilita o controle de circuitos eletrônicos através de sinais feitos com as mãos. Para tanto, é necessário um computador equipado com uma webcam, uma placa Arduino, módulos relés, lâmpadas ou outros dispositivos eletrônicos, além de fios e conectores elétricos. Nestes experimentos, as imagens capturadas pela webcam são os sinais de entrada do circuito de controle. As imagens capturadas são processadas para que sejam traçados os *bounding boxes*, permitindo que pontos de referência (*landmarks*) sejam encontrados. Por meio da localização destes landmarks, são geradas interpretações, por exemplo, qual mão foi detectada (direita ou esquerda), quantos dedos estão levantados e o cálculo do valor da distância entre dois dedos. Uma vez que a detecção é capaz de gerar diferentes saídas, basta utilizá-las para enviar diferentes códigos binários (caracteres ou números decimais) para o controlador do circuito eletrônico. A placa controladora Arduino Uno foi utilizada como "cérebro do circuito de controle". A placa recebe esses dados através de sua comunicação serial, e os utiliza como entrada para controle de suas saídas (dispositivos eletrônicos). Uma visão geral do recebimento e envio dos dados de entrada e saída dos experimentos é ilustrada na Figura 2.6.

2.7. Módulo Detector de Mãos

Com os recentes avanços no campo da visão computacional, diversas aplicações têm sido desenvolvidas com propósito de oferecer soluções para diversos problemas do cotidiano. Sistemas de visão têm sido implementados na robótica, veículos autônomos, câmeras, dentre outros. Em geral, estes sistemas buscam detectar um ou mais objetos presentes em imagens. Redes Neurais como a YOLO são treinadas para detectar e classificar objetos presentes em imagens captadas por uma câmera. Várias soluções de campos da Inteligência Artificial são disponibilizadas para desenvolvedores por meio de bibliotecas de alguma linguagem de programação como Python, R ou Matlab. Um exemplo é a biblio-

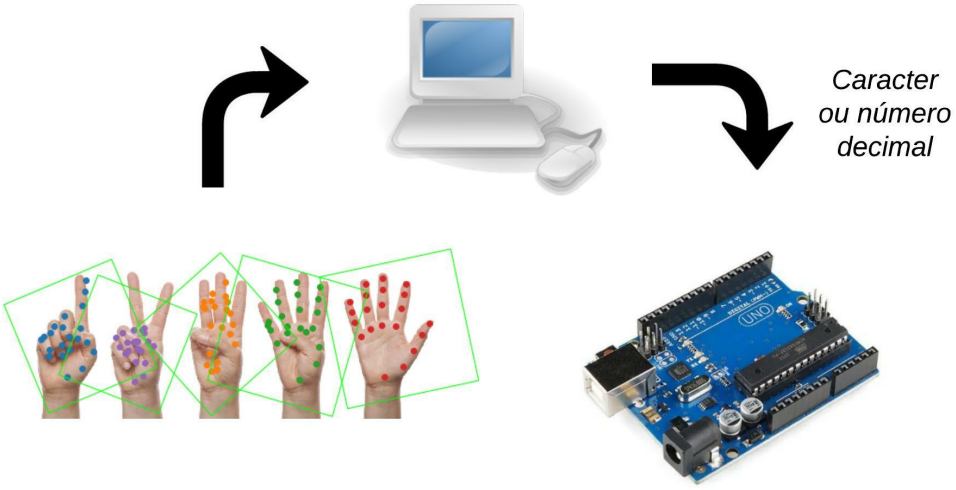

Figura 2.6. Experimento proposto.

teca *opencv*, que é uma biblioteca multiplataforma, totalmente livre ao uso acadêmico e comercial, para o desenvolvimento de aplicativos na área de Visão Computacional.

Uma biblioteca disponível para desenvolvedores que utilizam a linguagem de programação python que vem sendo bastante utilizada em aplicações de visão computacional, é a CVzone (*Computer Vision Zone*), solução criada por [Hassan, 2020], que simplifica a utilização de módulos presentes na biblioteca *opencv*. Estão disponíveis tutoriais, cursos, livros e projetos, que são utilizados como ferramentas para o ensino de diversas aplicações como programação de drones, detecção de textos, detecção de objetos, robótica e realidade aumentada. Através da biblioteca são disponibilizadas diversas ferramentas e módulos que podem ser utilizados para desenvolvimento de aplicações, como a apresentada na Figura 2.7, em que objetos presentes em uma imagem são detectados e são rotulados de acordo com a maior probabilidade de pertencerem à uma classe, dentro de um número *n* de classes possíveis. O uso desta biblioteca facilita a realização de pesquisas para desenvolvimento de sistemas de reconhecimento de gestos e sinais, como o projeto apresentado por [Testa, 2020], em que foi desenvolvido um sistema de reconhecimento da Língua Brasileira de Sinais (LIBRAS) aplicado a robôs de serviço.

Um dos módulos existentes na biblioteca é o módulo detector de mãos. Este módulo fornece uma ferramenta capaz de detectar mãos que estão presentes em uma imagem, contar quantos dedos estão levantados e calcular distância entre dedos ou entre mãos. Cada mão detectada em uma imagem recebe o rótulo de esquerda (left) ou direita (right) e a marcação dos 21 pontos chaves (*landmarks*) utilizados para identificação, que são mostrados na Figura 2.8. Por meio destes pontos é possível realizar o cálculo de distâncias específicas, além do reconhecimento de sinais realizados com as mãos. A Figura 2.9 apresenta um exemplo de detecção realizada por meio do módulo *handDetector*.

A parte inicial do primeiro código-fonte utilizado para controle de dispositivos eletroeletrônicos é mostrado na Figura 2.10. Neste exemplo, são utilizados como sinais de entrada para o circuito de controle, o total de dedos levantados em uma mão, que tem

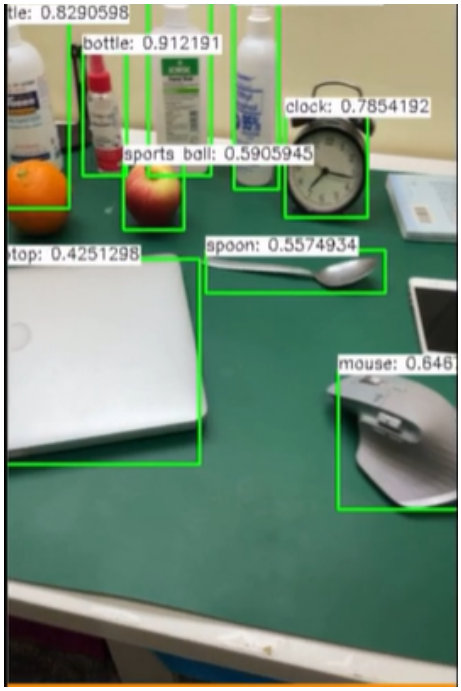

Figura 2.7. Exemplo de aplicação de Visão Computacional.

sua imagem capturada por uma webcam. Deste modo, é possível controlar o acionamento de até 5 dispositivos. Inicialmente, é feita a instalação de algumas bibliotecas (linhas 1 a 6), que são comumente utilizadas e algumas que são requisitos para funcionamento do código. É importante que sejam utilizados ambientes virtuais para execução dos códigos. Estes ambientes permitem que projetos, dependências e bibliotecas sejam isolados em um único local. Deste modo, elimina-se a possibilidade de que a instalação das dependências de um projeto interfira em outros projetos, já que é relativamente comum que no desenvolvimentos de projetos sejam utilizadas versões diferentes de uma mesma biblioteca. O editor de código-fonte escolhido para criação dos projetos foi o Visual Studio Code.

Após a instalação das dependências (bibliotecas), é iniciada a captura de vídeo via webcam (linha 9), seguida da inicialização da comunicação serial (linhas 11 e 12). É importante especificar de forma correta a porta que será utilizada para comunicação serial com a placa Arduino. Para verificar qual porta está sendo utilizada, basta acessar o ícone Ferramentas/Porta na IDE (do inglês, *Integrated Development Environment*) Arduino. A seguir, é necessário configurar a captura do detector. Dois parâmetros são essenciais, o limite de confiança para detecção de objetos (detectionCon) e o número máximo de mãos que será detectado (maxHands). Além disso, é utilizada uma variável que irá armazenar o total de dedos que estarão levantados. A parte final do código, Figura 2.11, é composta for comandos utilizados para contabilizar o total de dedos levantados e pelo envio deste dado à placa Arduino. Uma vez que estes comandos precisam ser executados de forma ininterrupta, é utilizada a instrução *while*. Inicialmente, é realizada a captura de um quadro de vídeo (linha 15), seguido do ajuste do tamanho da janela em que as imagens serão exibidas (linha 16). A próxima instrução (linha 17) é utilizada para obtenção dos landmarks de cada mão detectada na imagem, sendo que esses dados são salvos em uma lista,

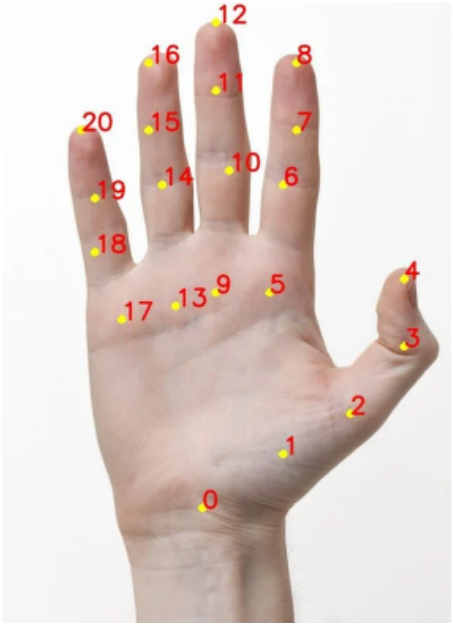

Figura 2.8. Pontos Utilizados para Identificação das Mãos.

denominada *hands*. A seguir, é utilizado o método *fingersUp* (linha 23) que retorna uma lista contendo 5 bits indicando o estado de cada dedo, sendo 1 utilizado para indicar que um determinado dedo está levantado. Por exemplo, se a lista retornada contiver os valores [0,0,1,1,1] significa que os dedos polegar e indicador estão abaixados e os demais estão levantados. A parte final do código contém as instruções utilizadas para contabilizar quantos dedos estão levantados, enviar este dado para a placa Arduino e para escrever a informação na janela que está exibindo a captura feita pela webcam.

O segundo exemplo consiste em controlar um dos sinais PWM da placa Arduino, utilizando como sinal de entrada a distância entre dois pontos chave (landmarks). É possível utilizar a distância entre dois pontos presentes na mesma mão (Figura 2.12) ou em mãos diferentes. A distância calculada será enviada a placa, que utilizará este valor como referência para controle de algum dispositivo conectado a uma de suas saídas analógicas. Utilizando a saída PWM é possível controlar a velocidade de rotação de um motor, a intensidade luminosa de uma lâmpada, assim como outras grandezas analógicas. As instruções utilizadas para que o módulo calcule a distância entre dois pontos são mostradas na Figura 2.13. É necessário obter a lista com os 21 pontos (linha 21), e depois utilizar o método *findDistance* (linha 22), que tem como parâmetros a especificação de quais pontos chave serão utilizados como referência para o cálculo. Também foram utilizadas instruções para desenhar um gráfico de barra, que mostra como o percentual de ciclo de trabalho do sinal de PWM deve variar de acordo com a distância entre os dedos polegar e indicador. Para isso, foram desenhados dois retângulos (linhas 29 e 30), um que tem altura fixa e outro com altura variável, além do valor percentual. É importante ajustar o valor da distância calculada à posição do gráfico desenhado na tela, sendo para isso utilizado o método *interp* da biblioteca *numpy*.

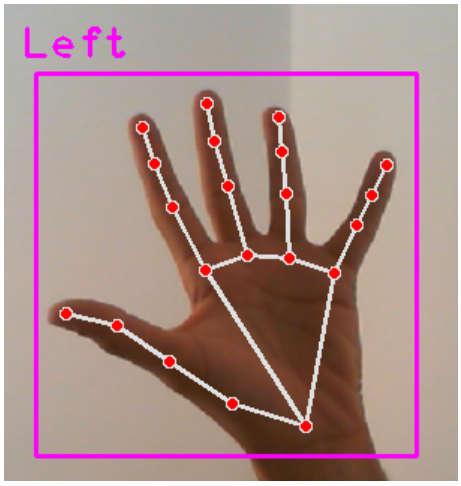

Figura 2.9. Detecção Realizada com o Módulo *HandDetector.*

```
1  import numpy
2  import serial
3  import cvzone
4  import cv2
5  from cvzone.HandTrackingModule import HandDetector
6  import matplotlib.pyplot as plt
7
8  #inicia a captura via webcam
9  cap = cv2.VideoCapture(0)
10 #inicia a comunicação com Arduino
11 serialcomm = serial.Serial('COM16', 9600)
12 serialcomm.timeout = 1
13 #configuração do detector
14 detector = HandDetector(detectionCon=0.8,maxHands=1)
15 #inicializa contador
16 w=0
```

Figura 2.10. Parte Inicial do Código Utilizado no Primeiro Experimento.

2.8. Programação da Placa Arduino

A programação de uma placa Arduino é realizada utilizando-se a linguagem C++ adaptada, e os códigos-fonte são escritos, compilados e enviados para a placa utilizando-se a IDE Arduino. A organização Arduino disponibiliza para consulta alguns tutoriais, últimas atualizações de placas e bibliotecas, além de outros serviços. Todos os serviços são gratuitos, uma vez que a iniciativa Arduino tem como objetivo a disseminação de conhecimento para o maior número de pessoas possível.

As instruções utilizadas para escrita dos algoritmos são em geral simples e intuitivas. Também é possível instalar bibliotecas que permitem utilizar de forma mais simples módulos de expansão. Diariamente, desenvolvedores disponibilizam versões de bibliotecas para tornar cada vez mais simples o desenvolvimento de projetos com a plataforma. Cada código fonte Arduino é composto obrigatoriamente de duas funções: *void setup() e void loop()*. A primeira é formada por comandos que irão configurar os pe-

```
14 while True:
15     success, img = cap.read() #leitura da imagem
16     img=cv2.resize(img,(800, 600)) #dimensiona janela
17     #Detecta mao(s) presentes na imagem
18     hands, img = detector.findHands(img,draw=True)
19     if hands:
20         #escolhe a 1ª mão detectada
21         hand1 = hands[0]
22         #armazena em um vetor quais dedos estão levantados
23         f = detector.fingersUp(hand1)
24         w=0
25         e='\n'#delimitador de dado transmitido
26         for q in f:
27             w +=q
28         #envia valor para arduino
29         serialcomm.write(e.encode())
30         serialcomm.write(str(w).encode())
31     cv2.putText(img, "Dedos: " + str(int(w)), (10,70),
32     cv2.FONT_HERSHEY_PLAIN, 3, (0,0,255), 3)
33     cv2.imshow("Image", img)
34     cv2.waitKey(10)
```

Figura 2.11. Parte Final do Código Utilizado no Primeiro Experimento.

riféricos da placa, como pinos de entrada e saída, comunicação serial, além de ser utilizada inicialização de alguns dispositivos como displays e relógios. A segunda contém as instruções que serão executadas de forma cíclica, durante o tempo em que a placa estiver alimentada.

O código necessário para que a placa receba dados de um computador de forma serial é bastante simples, uma vez que os comandos utilizados para comunicação são nativos da linguagem, além da IDE conter um monitor para comunicação serial. A parte inicial do código, Figura 2.14, contém as variáveis que serão utilizadas. A linha 1 contém a variável que irá armazenar a string recebida pelo Arduino, seguida da definição de conexão das lâmpadas nos pinos da placa (linha 3). A seguir a comunicação serial é iniciada com taxa de transmissão de dados de 9600 pbs, e os pinos onde serão conectadas as lâmpadas são definidos como saídas.

A parte principal, Figura 2.15 do código contém as instruções responsáveis pelo recebimento dos dados enviados pelo computador e pelo acionamento dos dispositivos eletroeletrônicos. Inicialmente, é verificada se a comunicação serial foi estabelecida (linha 18). A seguir a string enviada pelo computador é armazenada (linha 21) e convertida para um valor do tipo inteiro (linha 23). Agora basta que esse valor seja testado para que seja definido de que forma as lâmpadas serão acionadas. Optou-se pela escolha do comando switch para seleção das ações que serão tomadas pelo controlador. É necessário observar que para acionar uma lâmpada utilizou-se o comando *digitalWrite* com argumento LOW (nível lógico baixo), e para apagar o argumento HIGH foi utilizado. Isto deve-se devido ao módulo relé utilizado ser acionado com nível lógico baixo.

Além de ser possível controlar o acionamento de saídas digitais, a placa arduino também permite que saídas sejam controladas de forma analógica. Um exemplo, é o

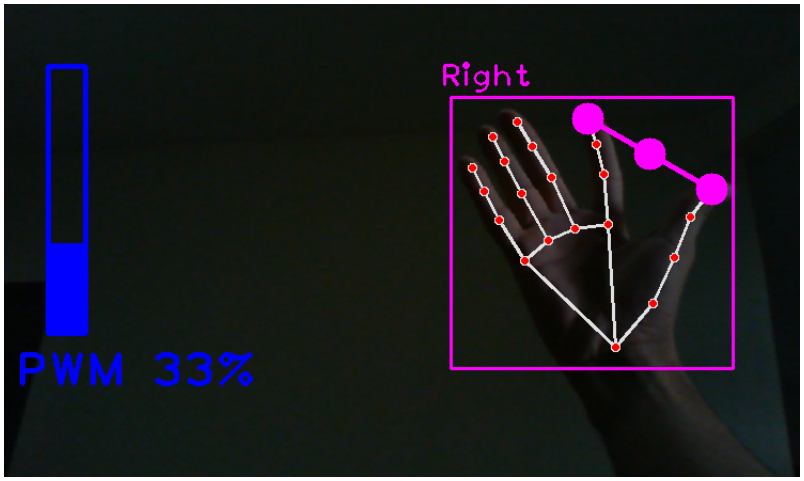

Figura 2.12. Distância entre dois pontos utilizada para controle de um sinal PWM.

controle da velocidade de um motor de corrente contínua ou o controle da intensidade luminosa de um LED ou de uma lâmpada. Para isso, é necessária a utilização de uma das saídas PWM da placa arduino. Para encontrar estes pinos basta procura pelo símbolo semelhante à uma onda. Utilizando-se um sinal PWM, é possível controlar o valor de potência entregue à uma carga. Uma vez que o módulo de detecção de mãos consegue realizar a medição da distância entre dois dedos, ou entre duas mãos, o valor dessa distância é enviado para a placa arduino para ser utilizado como valor de referência para o PWM. O código para este tipo de acionamento é mostrado na Figura 2.16. O código é relativamente simples. Após o recebimento da string e transformação da mesma em um número do tipo inteiro (linha 10), o valor recebido deve ser restringido a um intervalo que varia entre 0 e 255 (linha 13), uma vez que são os valores mínimo e máximo utilizados para configuração do PWM. A seguir, basta enviar este valor para o pino onde está conectado o dispositivo que será controlado de forma analógica (linha 14).

2.9. Circuito de Acionamento

A placa Arduino, assim como outros dispositivos de controle, emitem um sinal de saída capaz de acionar pequenas cargas, como LEDs (Light Emitting Diodes). Em geral os pinos operam em 3,3 ou 5 V, fornecendo uma corrente máxima que costuma variar entre 25 e 40 mA [Banzi and Shiloh, 2022]. Neste caso para que a placa Arduino possa controlar dispositivos alimentados com valores mais elevados de corrente e tensão elétrica é necessário utilizar algum circuito de acionamento ou circuito de potência. Este tipo de circuito é amplamente utilizado na indústria e em outros setores de serviços, uma vez que em uma instalação elétrica existem circuitos que operam com diferentes níveis de tensão [Segundo and Rodrigues, 2015].

Existem diferentes componentes elétricos que podem ser utilizados para permitir a conexão da placa arduino com uma uma carga que opere com níveis de potência mais elevado, ou que sejam alimentados com tensão alternada. Podemos citar o Retificador Controlado de Silício (SCR, do inglês *Silicon Controlled Rectifier*), TRIAC (do inglês, *Triode for Alternating Current*), optoacopladores ou acopladores ópticos, relés do estado sólido,

```python
while True:
    success, img = cap.read()
    img=cv2.resize(img,(800, 600))
    hands, img = detector.findHands(img,draw=True)
    if hands:
        hand1 = hands[0]
        #lista de 21 landmarks - pontos detectados em uma mão
        lmlist1 = hand1["lmList"]
        length, info, img = detector.findDistance(lmlist1[4], lmlist1[8],img)
        e='\n'
        # envia para o arduino
        serialcomm.write(e.encode())
        serialcomm.write(str(length).encode())
    bar = np.interp(length, [50, 300], [400, 150])
    bar_per = np.interp(length, [50, 300], [0, 100])
    cv2.rectangle (img, (50,150), (85,400), (255,0,0), 3)
    cv2.rectangle (img, (50,int(bar)), (85,400), (255,0,0), cv2.FILLED)
    cv2.putText(img, f'PWM {int(bar_per)}%', (20,450),
    cv2.FONT_HERSHEY_PLAIN, 3, (255,0,0), 3)
    cv2.imshow("Image", img)
    cv2.waitKey(10)
```

Figura 2.13. Código python Utilizado no Segundo Experimento (PWM).

transistores, dentre outros. Devido ao fato de não haver componentes mecânicos em sua construção estes dispositivos são denominados chaves estáticas. O símbolo elétrico de alguns destes dispositivos é mostrado na Figura 2.17. A escolha de qual componente deve ser utilizado deve levar em consideração o custo, vida útil do componente, nível de proteção necessária, disponibilidade de mercado. No experimento descrito, é utilizado o relé eletromecânico devido ao seu menor custo e maior disponibilidade. Seu símbolo elétrico e o módulo utilizado são mostrados na Figura 2.18.

A conexão da placa Arduino ao módulo relé e do módulo à lâmpada são mostrados na Figura 2.19. O módulo relé permite que o Arduino controle dispositivos eletro-eletrônicos que são alimentados com tensão superior à tensão de operação dos pinos de saída da placa. Neste caso, a placa envia um sinal para o módulo relé energizando sua bobina, realizando o acionamento do dispositivo conectado a seus terminais alimentados com tensão elétrica mais elevada. O terminal fase do circuito de alimentação deve ser conectado ao terminal comum (COM) do relé. A lâmpada deve ter seus terminais conectados da seguinte forma: um conectado ao contato normalmente aberto do relé (NA, do inglês *Normally open*) e o outro deverá ser conectado diretamente ao terminal Neutro da alimentação. Desta forma, o relé realizará o chaveamento através do terminal da fase, procedimento recomendado pela norma NBR5410 da ABNT (Associação Brasileira de Normas Técnicas) [ABNT, 2004]. É necessário ter atenção com relação a conexão dos terminais do receptáculo da lâmpada, para que os terminais sejam conectados de forma correta, para uma conexão segura. Além do baixo custo, o relé tem como vantagem o fato de seus terminais de tensão baixa (que são conectados diretamente ao Arduino) serem totalmente isolados dos terminais de tensão mais elevada, onde são conectados os dispositivos que têm seu acionamento controlado pela placa Arduino.

```
String dado_byte;
//Pinos onde as lâmpadas serão conectadas
int L_1=4,L_2=5,L_3=6,L_4=7,L_5=8;
int num;
void setup( )
{
//Inicializa porta serial
Serial.begin(9600);
//configura pinos como saídas
pinMode(L_1,OUTPUT);
pinMode(L_2,OUTPUT);
pinMode(L_3,OUTPUT);
pinMode(L_4,OUTPUT);
pinMode(L_5,OUTPUT);
}
```

Figura 2.14. Experimento 1: Parte Inicial do código-fonte para programação da placa Arduino.

2.10. Considerações Finais

Neste capítulo foi descrito o minicurso intitulado “Acionamento de Dispositivos Eletroeletrônicos Utilizando Visão Computacional”que foi aceito para ser ministrado na décima Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI). O curso apresenta de que forma é possível controlar o acionamento de dispositivos eletroeletrônicos, tanto de forma digital quanto de forma analógica, utilizando Visão Computacional, campo de estudo da Inteligência Artificial em que são pesquisadas e desenvolvidas formas que possibilitem aos computadores interpretar dados para extrair informações de imagens. Recentemente, grandes avanços foram alcançados como classificação automática de imagens, e detecção automática de objetos presentes em imagens e reconhecimento facial.

Inicialmente foram apresentados conceitos básicos de Visão Computacional, com uma breve explicação sobre os métodos utilizados para processamento de imagens. O tópico seguinte trouxe de forma detalhada a fundamentação teórica sobre as Redes Neurais Convolucionais, que são os modelos computacionais mais utilizados em tarefas de visão computacional. Foi explicado de que forma uma rede convolucional extrai características presentes em uma imagem para realizar a sua classificação, além de alguns parâmetros que devem ser ajustados para realização do treinamento da rede. A seguir, foi apresentada a placa Arduino Uno, utilizada para confecção dos circuitos elétricos, juntamente com uma visão geral dos experimentos. Detalhes sobre o processo de captura e detecção de mãos presentes em imagens também foram apresentados, assim como figuras que mostram exemplos de detecção realizada com métodos existentes nas bibliotecas opencv e cvzone, sendo que as principais instruções que fazem parte do código foram explicadas. A parte final traz o código-fonte utilizado para programação da placa Arduino e o esquema elétrico do circuito de acionamento.

O curso tem como objetivo principal apresentar uma possibilidade para desenvolvimento de projetos de inovação tecnológica e científica que apliquem de forma integrada conhecimentos das áreas de microeletrônica e inteligência artificial, especificamente da

```
16 void loop( )
17 {
18 if(Serial.available( ) > 0)
19  {
20     //armazena dado recebido
21     dado_byte = Serial.readStringUntil( '\n');
22     //converte para um valor do tipo inteiro
23     num = arrivingdatabyte.toInt();
24     switch (num){
25       case 1:
26       //Aciona Lampada 1
27       digitalWrite(L_1,LOW);
28       digitalWrite(L_2,HIGH);
29       digitalWrite(L_3,HIGH);
30       digitalWrite(L_4,HIGH);
31       digitalWrite(L_5,HIGH);
32       break;
```

Figura 2.15. Experimento 1: Parte Principal do código-fonte para programação da placa Arduino.

```
 7 void loop()
 8 {
 9 while (Serial.available() > 0){
10 int red = Serial.parseInt();
11 if (Serial.read() == '\n')
12   {
13    red = constrain(red, 0, 255);
14    analogWrite(redPin, red);
15    Serial.print(red);}
16   }
17 }
```

Figura 2.16. Experimento 2: Parte Principal do código-fonte para programação da placa Arduino.

área da Visão Computacional , que têm sido amplamente utilizadas em diversas aplicações atualmente. Espera-se que futuramente possam ser desenvolvidos projetos em áreas da automação, e principalmente que possam ser desenvolvidos serviços e recursos de tecnologia assistiva, que possam prover alguma assistência e melhoria a vida de pessoas que possuam alguma necessidade específica.

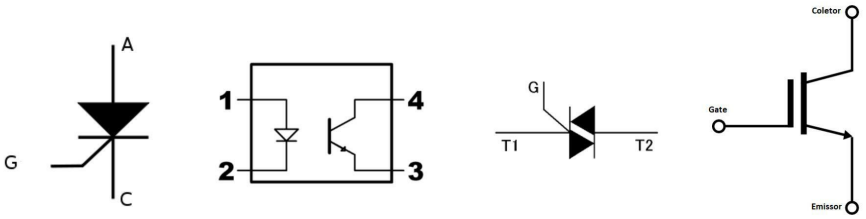

Figura 2.17. Chaves Estáticas de Potência (SCR, Optoacoplador, TRIAC, Transistor).

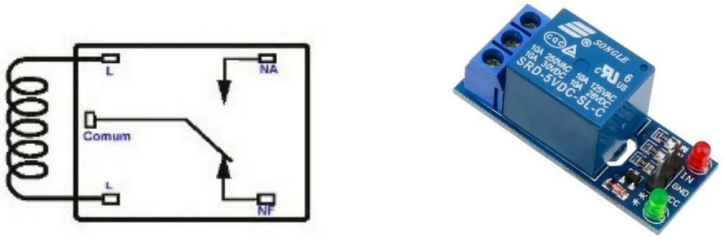

Figura 2.18. Relé: Símbolo elétrico e Módulo de expansão.

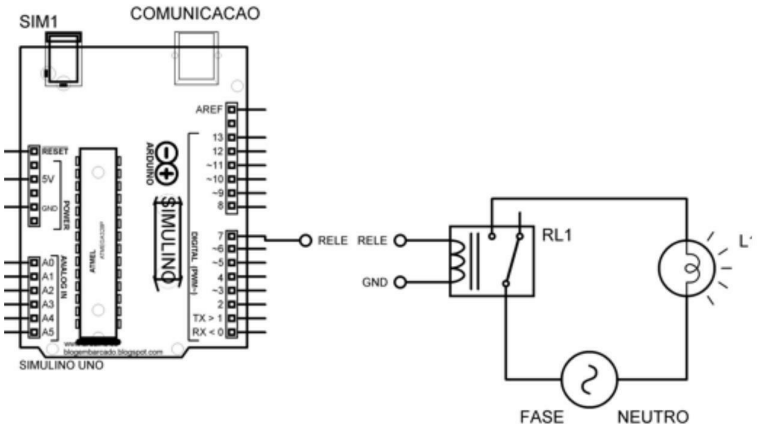

Figura 2.19. Conexão do circuito de acionamento.

Referências

[ABNT, 2004] ABNT (2004). *ABNT NBR 5410: instalações elétricas de baixa tensão*.

[Academy, 2019] Academy, D. S. (2019). Deep learning book. http://deeplearningbook.com.br. Disponível; acessado em 05-Junho-2022.

[Araújo et al., 2017] Araújo, F., Carneiro, A., Silva, R., Medeiros, F., and Ushizima, D. (2017). Redes neurais convolucionais com tensorflow:teoria e prática. 1:382–406.

[Banzi and Shiloh, 2022] Banzi, M. and Shiloh, M. (2022). *Getting started with Arduino*. Maker Media, Inc.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee.

[Ferreira et al., 2020] Ferreira, M. M., Esteve, G. P., Junior, G. B., de Almeida, J. D. S., de Paiva, A. C., and Veras, R. (2020). Multilevel cnn for angle closure glaucoma detection using as-oct images. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 105–110. IEEE.

[Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

[Girshick et al., 2015] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2015). Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[Hassan, 2020] Hassan, M. (2020). Computer vision zone: Cv zone. https://www.computervision.zone. Disponível; acessado em 05-Junho-2022.

[Haykin, 1999] Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916.

[Kaehler and Bradski, 2016] Kaehler, A. and Bradski, G. (2016). *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. "O'Reilly Media, Inc.".

[Kaliyar et al., 2021] Kaliyar, R. K., Goswami, A., and Narang, P. (2021). Fakebert: Fake news detection in social media with a bert-based deep learning approach. *Multimedia Tools and Applications*, 80(8):11765–11788.

[Kondaveeti et al., 2021] Kondaveeti, H. K., Kumaravelu, N. K., Vanambathina, S. D., Mathe, S. E., and Vappangi, S. (2021). A systematic literature review on prototyping with arduino: Applications, challenges, advantages, and limitations. *Computer Science Review*, 40:100364.

[McRoberts, 2018] McRoberts, M. (2018). *Arduino básico*. Novatec Editora.

[Prince, 2012] Prince, S. (2012). *Computer Vision: Models, Learning, and Inference*. Computer Vision: Models, Learning, and Inference. Cambridge University Press.

[Raman et al., 2019] Raman, R., Srinivasan, S., Virmani, S., Sivaprasad, S., Rao, C., and Rajalakshmi, R. (2019). Fundus photograph-based deep learning algorithms in detecting diabetic retinopathy. *Eye*, 33(1):97–109.

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

[Saha, 2018] Saha, S. (2018). A comprehensive guide to convolutional neural networks. http://www.towardsdatascience.com. Disponível; acessado em 20-Junho-2022.

[Segundo and Rodrigues, 2015] Segundo, A. K. R. and Rodrigues, C. L. C. (2015). Eletrônica de potência e acionamentos elétricos. *Ouro Preto: Ifmg-Instituto Federal de Ciencia, Educação e Tecnologia de Minas Gerais*.

[Shang et al., 2019] Shang, Q., Zhao, Y., Chen, Z., Hao, H., Li, F., Zhang, X., and Liu, J. (2019). Automated iris segmentation from anterior segment oct images with occludable angles via local phase tensor. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4745–4749, Berlin, Germany. IEEE.

[Testa, 2020] Testa, B. (2020). Sistema de reconhecimento de língua brasileira de sinais aplicado a robôs de serviço.

[Zhang et al., 2018] Zhang, L., Wang, S., and Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.