



UFRR

UNIVERSIDADE FEDERAL DE RORAIMA
PRÓ-REITORIA DE ENSINO E GRADUAÇÃO
CENTRO DE CIÊNCIA E TECNOLOGIA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

JOÃO PAULO VERÇOSA PINTO

HAND.IO: UMA LUVA PARA CONTROLE DE DISPOSITIVOS ELETRO-ELETRÔNICOS
UTILIZANDO RECONHECIMENTO DE GESTOS

Boa Vista - RR

2019

JOÃO PAULO VERÇOSA PINTO

**HAND.IO: UMA LUVAS PARA CONTROLE DE DISPOSITIVOS
ELETRO-ELETRÔNICOS UTILIZANDO RECONHECIMENTO DE GESTOS**

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Roraima como requisito para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. DSc. Herbert Oliveira Rocha

Boa Vista - RR

2019

Dados Internacionais de Catalogação na publicação (CIP)
Biblioteca Central da Universidade Federal de Roraima

P659h Pinto, João Paulo Verçosa.

Hand.io : uma luva para controle de dispositivos eletro-eletrônicos utilizando reconhecimento de gestos / João Paulo Verçosa Pinto. – Boa Vista, 2019.
60 f. : il.

Orientador: Prof. Dr. Herbert Oliveira Rocha.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Roraima, Curso de Ciência da Computação.

1 - Controle de dispositivos. 2 - Acelerômetro. 3 - Giroscópio. 4 - Ambientes inteligentes. 5 - Luva inteligente. I - Título. II - Rocha, Herbert Oliveira (orientador).

CDU - 681.057.8

Ficha Catalográfica elaborada pela Bibliotecária/Documentalista:
Maria de Fátima Andrade Costa - CRB-11/453-AM

JOÃO PAULO VERÇOSA PINTO

HAND.IO: UMA LUVA PARA CONTROLE DE DISPOSITIVOS ELETRO-ELETRÔNICOS
UTILIZANDO RECONHECIMENTO DE GESTOS

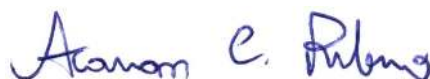
Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Roraima como requisito para a obtenção do grau de Bacharel em Ciência da Computação. Defendido em 10 de Julho de 2019 e aprovado pela seguinte banca:



Prof. DSc. Herbert Oliveira Rocha
Orientador / Curso de Ciência da Computação
- UFRR



Prof. DSc. Leandro Nelinho Balico
Curso de Ciência da Computação - UFRR



Prof. MSc. Acauan Cardoso Ribeiro
Curso de Ciência da Computação - UFRR

Dedico este trabalho à todos aqueles que acreditaram em mim e abriram as portas para que eu chegue onde eu jamais havia imaginado.

AGRADECIMENTOS

Agradeço à minha família, em especial meus pais, Raul e Cleire, que me deram condições de chegar à graduação e finalmente estar finalizando esta etapa da minha vida, que não é o fim, mas o começo das grandes coisas que estão por vir.

Agradeço à minha digníssima namorada Joana Harana, que esteve comigo durante os meus momentos de fraqueza e sempre me mostrou a luz nos momentos em que eu estava perdido nos meus pensamentos e na minha ansiedade.

Agradeço aos meus amigos e que sempre estiveram comigo durante esta jornada, em especial meus amigos de laboratório que passaram comigo momentos de incerteza e me ajudaram a supera-los juntos.

Agradeço ao meu orientador, Herbert Rocha, que me guiou durante os momentos finais, e me permitiu desenvolver um excelente trabalho depois de muito esforço e dedicação.

E finalmente, agradeço aos meus professores, que me apontaram o caminho do conhecimento, que eu irei utilizar para resolver problemas no mundo real e trazer frutos que vão engrandecer a sociedade como um todo.

À todos o meu mais sincero obrigado.

“Se você quiser fazer uma torta de maçã do nada, primeiro tem que inventar o universo.”

–Carl Sagan.

RESUMO

A crescente adoção da internet das coisas, criou novos paradigmas de interação entre usuário e sistemas. Interações mais naturais e personalizadas como comando por voz e por gestos parecem ser mais viáveis do que apenas a utilização de telas sensíveis ao toque. A Hand.io é um sistema vestível (composto por sensores de movimentos, como acelerômetro e giroscópio) de reconhecimento de gestos manuais que utiliza algoritmos de aprendizado de máquina para classificar sinais enviados pelo usuário, de maneira mais natural possível com o mínimo de dificuldade por parte do usuário, a fim de enviar comandos para manipulação de dispositivos eletrônicos em dado ambiente interconectado. Para o desenvolvimento do protótipo do sistema proposto, optou-se pelo classificador *Gaussian Naive Bayes* (NB), por apresentar uma taxa de acurácia de 81.1%, a maior dentre os testados, e possibilitou através de experimentos, que o protótipo atingisse uma taxa máxima de sucesso de 93.3% nos testes realizados.

Palavra-chaves: controle de dispositivos, acelerômetro, giroscópio, ambientes inteligentes, vestíveis, luva inteligente.

ABSTRACT

The increasing adoption of the internet of things has created new paradigms of interaction between users and systems. More natural and personalized interactions like voice commands and gestures seem to be more viable than just the use of touch screens. Hand.io is a wearable system (composed of motion sensors, such as accelerometer and gyroscope) of manual gesture recognition that uses machine learning algorithms to classify signals sent by the user, in a more natural way with minimal user effort in order to send commands for manipulation of electronic devices in a given interconnected environment. For the development of the prototype of the proposed system, the *Gaussian Naive Bayes* (NB) was chosen, as it had a rate of accuracy of 81.1%, the best among all the tested, and it allowed through experimentation, that the prototype achieved the highest rate of success of 93.3% in the tests performed.

Keywords: device control, accelerometer, gyroscope, smart environments, wearables, smart glove.

LISTA DE FIGURAS

Figura 1 – Diagrama de um sistema em tempo real genérico.	18
Figura 2 – Raspberry Pi 3.	20
Figura 3 – Pinagem do microcontrolador ATmega328P.	21
Figura 4 – Máquina de estados finita de uma máquina de venda de doces simples.	25
Figura 5 – Resultado da classificação de pontos em uma base de dados.	28
Figura 6 – Visão geral do protótipo.	34
Figura 7 – Fluxograma da Hand.io.	35
Figura 8 – Exemplos de gestos.	37
Figura 9 – Esquemático do protótipo da luva.	39
Figura 10 – Esquemático da central de processamento de sinais e execução de ações.	39
Figura 11 – Projeto da luva.	42
Figura 12 – Central de controle.	42
Figura 13 – Diagrama de classe da Hand.io.	44
Figura 14 – Classificador Gaussian Naive Bayes.	47
Figura 15 – Máquina de estados da Hand.io.	48
Figura 16 – Cenário experimental.	49

LISTA DE TABELAS

Tabela 1 – Estimativa de preços dos componentes.	38
Tabela 2 – Comparativo dos classificadores.	46
Tabela 3 – Dados coletados do experimento com um usuário experiente.	50
Tabela 4 – Dados coletados do experimento com voluntários inexperientes.	51
Tabela 5 – Dados coletados do questionário realizado.	52

SUMÁRIO

1	INTRODUÇÃO	12
1.1	Definição do Problema	14
1.2	Objetivos	14
1.3	Organização do trabalho	15
2	CONCEITOS E DEFINIÇÕES	16
2.1	Sistemas Embarcados	16
2.1.1	Restrições de um Sistema Embarcado	16
2.1.2	Sistemas de Tempo Real	18
2.1.3	Microcontroladores e Microprocessadores	19
2.2	Modelagem e verificação de sistemas	22
2.2.1	Unified Modeling Language	22
2.2.2	Máquina de Estados	24
2.3	Internet das Coisas	25
2.4	Reconhecimento de Padrões	25
2.4.1	Aprendizagem de Máquina	26
2.4.2	<i>Frameworks</i> de aprendizado de máquina	27
2.4.2.1	Scikit-learn	27
2.4.2.2	TensorFlow	28
3	TRABALHOS CORRELATOS	29
3.1	Accelerometer-based gesture control for a design environment	29
3.2	I'm home: Defining and evaluating a gesture set for smart-home control	30
3.3	uWave: Accelerometer-based personalized gesture recognition and its applications	32
4	MÉTODO PROPOSTO	34
4.1	Visão geral do método da Hand.io	34
4.2	Fluxo de execução da Hand.io	35
4.3	Captura de Dados Baseado em Movimentos de Amplitude de Punho e Mão	36
4.3.1	Conexão com a central de processamento	36
4.4	Reconhecimento de Padrões Baseado em Movimentos e Ações	36
4.5	Modelo de Conexão Entre Dispositivos Eletrônicos	37
4.6	Modelo de Prototipação	38

5	AVALIAÇÃO EXPERIMENTAL	41
5.1	Análise da implementação	41
5.1.1	Protótipo da luva de captura de movimentos	41
5.1.2	Protótipo da central de processamento de sinais	42
5.1.3	Análise do software do protótipo Hand.io	43
5.1.3.1	Análise dos softwares embarcados nos Arduinos	43
5.1.3.2	Análise do software da central de processamento	44
5.1.4	Fluxo de execução do Hand.io	47
5.2	Planejamento e projeto do cenário experimental	48
5.3	Execução dos experimentos e análise dos resultados	50
5.3.1	Teste com um usuário experiente	50
5.3.2	Teste com voluntários inexperientes	51
5.3.3	Aplicação do questionário ao grupo de voluntários	52
6	CONCLUSÕES E TRABALHOS FUTUROS	55
	REFERÊNCIAS	57

1 INTRODUÇÃO

A complexidade de sistemas computacionais cresce exponencialmente, visando aplicações em diferentes domínios, tais como: agricultura, automação residencial e robótica. Neste sentido, Sistemas Embarcados (SE) que são sistemas computacionais integrados a outros sistemas, usualmente são encontrados no nosso dia a dia. Geralmente, o principal propósito dos SE é o controle e provimento de informações para uma função específica (RAMESH et al., 2012) sendo estes extremamente interativos com seu ambiente, operam geralmente em tempo real, e estão disponíveis continuamente.

Segundo Yurish (2016), robôs se tornam mais inteligentes com respostas em tempo real. Sistemas como o Controle Integrado de Forças processam as variações com sensibilidade humana, melhorando a performance, e reduzindo assim o tempo de codificação. Isso torna os robôs mais inteligentes e capazes, como os humanos, de manusear itens com interações externas em tempo real. Dentre outras aplicações, robôs podem auxiliar pessoas com restrições motoras a realizar atividades cotidianas, sendo que para isso os mesmos devem interagir com rostos humanos

Levando em consideração estes avanços no campo de sensores robóticos, desenvolver sistemas embarcados com características robóticas, por exemplo, a capacidade de interação com o ambiente, pode ser uma solução viável para que humanos possam interagir com dispositivos eletro-eletrônicos (splints de ar-condicionado, televisões, computadores e outros) ao seu redor sem a necessidade do toque presencial. Um modo de tornar esta solução viável seria a utilização de luvas inteligentes, as quais contam com utilização de: acelerômetros, sensores de força, um microcontrolador e transmissores (NAVAS et al., 2012) para reconhecimento de padrões e então comunicação com os dispositivos eletro-eletrônicos (O'FLYNN et al., 2013) (BERNIERI et al., 2015) (CHOUDHARY et al., 2015). Adicionalmente, tais características robóticas podem auxiliar na recuperação de doenças ou mesmo prover suporte complementar em casos de necessidades especiais dos humanos, exemplo, deficiência visual.

No trabalho de Choudhary et al. (2015) é proposta uma nova abordagem para auxiliar a comunicação e a interação de indivíduos surdos-cegos, aumentando sua independência. Este inclui uma luva inteligente que traduz o alfabeto Braille, que é o mais universalmente utilizado pela população surda-cega alfabetizada, em texto e vice-versa, e comunica mensagens via SMS para contato remoto. Isto permite ao usuário transmitir mensagens simples através de sensores de toque capacitivos como sensores de entrada colocados na palma da luva e convertidos para texto pelo computador/telefone móvel. O usuário pode entender e interpretar mensagens recebidas através de padrões de retornos

táteis de pequenos motores de vibração no dorso da luva. A implementação bem-sucedida da tradução bidirecional em tempo real entre Inglês e Braille, e a comunicação entre o dispositivo vestível e o computador/telefone móvel abrem novas oportunidades para a troca de informação que até então estavam indisponíveis para indivíduos surdos-cegos, como a comunicação remota, assim como transmissões paralelas de um pra muitos.

O'Flynn et al. (2013) apresenta o desenvolvimento de uma luva inteligente para facilitar o processo de reabilitação de Artrite Reumatoide (PLASQUI, 2008) através da integração de sensores, processadores e tecnologia sem fio para medir empiricamente o alcance do movimento. Medições tradicionais de artrite precisam de exames pessoais intensamente trabalhosos realizados por uma equipe médica que através de suas medições objetivas podem dificultar a determinação e a análise da reabilitação da artrite. A luva proposta (O'FLYNN et al., 2013) usa uma combinação de 20 sensores de dobras, 16 acelerômetros de três eixos, e 11 sensores de força para detectar o movimento de juntas. Todos os sensores são posicionados em uma placa de circuito impresso flexível para permitir um alto nível de flexibilidade e estabilidade do sensor.

Analisando o processo de desenvolvimento destes sistemas, verifica-se que existem diferentes graus de complexidades tanto no software quanto no hardware. Desta forma, é necessário que as aplicações sejam projetadas considerando os requisitos de previsibilidade e confiabilidade, principalmente em aplicações de sistemas embarcados críticos, onde diversas restrições (por exemplo, tempo de resposta e precisão dos dados) devem ser atendidas e mensuradas de acordo com os requisitos do usuário, caso contrário uma falha pode conduzir a situações catastróficas. Por exemplo, o erro de cálculo da dose de radiação no Instituto Nacional de Oncologia do Panamá que resultou na morte de 23 pacientes (WONG et al., 2010). Contudo, erros durante o desenvolvimento de sistemas computacionais tornam-se mais comuns, principalmente quando se tem curto espaço de tempo de liberação do produto ao mercado, e estes sistemas precisam ser desenvolvidos rapidamente e atingir um alto nível de qualidade.

Visando contribuir com metodologias de desenvolvimento de sistemas embarcados, o contexto deste trabalho está situado em demonstrar métodos e técnicas no processo de codificação e prototipação de sistemas embarcados, tais como: Redes de Petri (BENDERS; STEVENS, 1992) (GIRAULT; VALK, 2002), UML (CUNHA et al., 2011); e Máquinas de Estados (LAMPKA et al., 2009). A utilização destas ferramentas garante à um sistema uma confiabilidade muito maior, por serem métodos padronizados com um certo formalismo matemático que podem ser testados de forma automatizada através de simulações. Sistemas que são utilizados praticamente durante o dia todo como peças de roupas precisam ter seu funcionamento garantido para que tenham utilidade ao usuário.

Sistemas integrados à roupa do usuário passarão a ser comuns na interação entre um usuário e seu ambiente, seja ele residencial ou urbano. Segundo Weiser (1999) as

redes ubíquas já são capazes de realizar certas operações em um contexto onde diversos dispositivos inteligentes estão em constante comunicação. Em um futuro próximo objetos cotidianos e aparelhos eletro-eletrônicos passarão a ser capazes de se comunicarem uns com os outros, e compartilhar dados de seus sensores com outros sistemas que podem ser utilizados para mudar completamente a relação dos usuários com o seu ambiente. Existem aplicações nos campos de: medicina, automação industrial, gerenciamento inteligente de energia, assistência à idosos, e muitos outros. (ZANELLA et al., 2014)

Neste contexto, este trabalho visa o desenvolvimento de uma luva inteligente de baixo custo para controle de dispositivos eletrônicos por meio do reconhecimento de padrões de movimentos, visto que tais aplicações se mostram de grande utilidade para o usuário em um ambiente inteligente, assim provendo conforto e facilidades ao seu usuário. Adicionalmente, para o desenvolvimento do sistema proposto, este trabalho visa projetar e analisar componentes de hardware e software com o foco em IoT que tenham um baixo consumo de energia; baixo custo de implementação; e aplicação de métodos para garantir a qualidade da execução do sistema computacional proposto.

1.1 Definição do Problema

O problema considerado neste trabalho é expresso na seguinte questão: Como projetar e desenvolver um sistema embarcado, para uma luva inteligente de baixo custo, utilizando métodos e técnicas para garantir os requisitos de previsibilidade e confiabilidade do sistema, de tal forma que o sistema proposto auxilie o seu usuário na comunicação com dispositivos eletrônicos?

1.2 Objetivos

O objetivo principal deste trabalho é projetar e avaliar um sistema computacional para uma luva inteligente efetuar a comunicação com dispositivos eletrônicos por meio do reconhecimento de padrões de gestos manuais, utilizando métodos e técnicas no processo de codificação e prototipação deste dado sistema, visando garantir os requisitos de previsibilidade; confiabilidade; e baixo custo. Assim, focando na criação de uma interface única e intuitiva entre um usuário e um ambiente inteligente, visando facilitar o uso de dispositivos através de gestos, que são um meio natural de comunicação.

Os objetivos específicos são:

1. Identificar métodos para a modelagem do software e hardware;
2. Projetar e desenvolver uma central de controle que será o meio de comunicação entre os dispositivos eletrônicos disponíveis em um ambiente e a luva para obtenção dos

- dados de movimentos;
3. Propor um algoritmo para reconhecimento e classificação dos movimentos/sinais enviados pela luva na mão do usuário;
 4. Desenvolver um protótipo da luva e da central de controle; e
 5. Validar o método proposto pela análise da prototipação do sistema proposto, a fim de examinar a sua eficácia e aplicabilidade.

1.3 Organização do trabalho

Este trabalho é organizado em capítulos que servem de base para a resposta dos problemas de pesquisa deste trabalho.

No **Capítulo 1: Introdução** foi apresentada uma contextualização do trabalho, a definição do problema e os objetivos deste trabalho.

No **Capítulo 2: Conceitos e definições** são apresentadas as ferramentas que serão utilizadas no método.

No **Capítulo 3: Trabalhos correlatos** são apresentados trabalhos similares a este e como estes trabalhos contribuíram para a definição do método.

No **Capítulo 4: Método proposto** é definido como os objetivos deste trabalho serão atingidos.

No **Capítulo 5: Resultados experimentais** é apresentados os resultados obtidos a partir da implementação do método proposto da Hand.io.

No **Capítulo 6: Conclusões e trabalhos futuros** é realizada uma análise crítica do trabalho juntamente com considerações que o autor deixou para futuras versões da Hand.io.

2 CONCEITOS E DEFINIÇÕES

Este capítulo tem como objetivo apresentar os principais conceitos e definições que serão utilizados para o entendimento e desenvolvimento deste trabalho.

2.1 Sistemas Embarcados

De acordo com [Vahid e Givargis \(2001\)](#) um sistema embarcado é um sistema computacional desenvolvido para um propósito específico, que em contraposição a sistemas de propósito geral, realiza apenas uma tarefa específica. Existem certas características (exemplo, restrição de memória e tempo de execução) que distinguem um sistema embarcado dos demais, que apesar de nem sempre serem atendidas, servem de referência para a classificação de tais sistemas. Sistemas embarcados geralmente realizam apenas uma atividade repetidas vezes, contam com restrições mais apertadas que um sistema normal, são reativos e muitas vezes funcionam em tempo real. Estes sistemas geralmente fazem parte de sistemas maiores, e quase sempre funcionam sem o conhecimento do usuário.

Dispositivos como *smartphones* e computadores pessoais, são compostos por diversos sistemas embarcados que realizam apenas uma função, mas que quando estão juntos passam a realizar diversas atividades, passando a ser classificado como um sistema de propósito geral. Um exemplo de um sistema embarcado que faz parte de um sistema maior seria a placa de rede de um smartphone ([QUALCOMM, 2017](#)), que por si só é um sistema que apenas recebe e envia dados através de uma rede sem fio, mas que quando contextualizado em um smartphone possibilita o acesso a internet, o que promove a realização de diversas atividades disponíveis online, como acesso à redes sociais ou a reprodução de vídeos ([VAHID; GIVARGIS, 2001](#)).

2.1.1 Restrições de um Sistema Embarcado

Um sistema embarcado geralmente trabalha em condições restritas nas quais certas métricas devem ser cumpridas para que seu funcionamento se dê de maneira eficiente. [Marwedel \(2011\)](#) define uma série de métricas utilizadas para medir a eficiência de um dado sistema como: consumo energético, eficiência em tempo de execução, tamanho de código, peso e custo. Em alguns casos existem situações onde estas métricas entram em conflito, neste caso cabe ao projetista do sistema definir qual a melhor configuração abrindo mão de certas funcionalidades para melhor atender as necessidades da aplicação.

O consumo de energia em um sistema embarcado muitas vezes é limitado devido à tecnologia atual de baterias e ao tipo do circuito utilizado. Circuitos Integrados de

Aplicações Específicas (ASICs, em inglês) tendem a ter uma eficiência energética muito superior aos demais, no entanto abrem mão de flexibilidade no desenvolvimento do software, circuitos reconfiguráveis como os Arranjos de Portas Programáveis em Campo (FPGAs, em inglês) tem sua eficiência energética inferior aos ASICs, mas oferecem uma flexibilidade de desenvolvimento muito maior apesar de serem limitados pelo tamanho dos circuitos reconfiguráveis disponíveis (MARWEDEL, 2011).

A limitação entre flexibilidade e eficiência também se aplica aos processadores. Processadores desenvolvidos especificamente para processamento de sinais analógicos, por exemplo, são exponencialmente mais eficientes energeticamente que processadores de propósito geral, que contam com a pior eficiência energética dentre os circuitos apresentados, mas permitem uma grande gama de possibilidades durante o desenvolvimento do sistema (MARWEDEL, 2011).

As limitações em tempo de execução devem ser reduzidas ao máximo para aproveitar o hardware disponível da melhor maneira possível. Problemas acarretados por compiladores que geram binários que não utilizam todo o potencial da arquitetura devem ser corrigidos a fim de eliminar desperdícios de instruções por ciclo de processamento (MARWEDEL, 2011).

O tamanho do código e as limitações em espaço de armazenamento são um desafio recorrente no mundo dos sistemas embarcados. Em diversas aplicações onde não existe a possibilidade de carregamento dinâmico de dados, como ocorrem nos *smartphones*, todos os dados devem ser armazenados na memória limitada dos chips, como nos casos de *Systems on a Chip* (SoC), nos quais todos os componentes de processamento de dados se encontram em um único chip. Nesse tipo de situação onde o espaço se torna um recurso precioso é necessário um cuidado maior durante a construção do código que será executado na plataforma. Tal limitação pode ser contornada com a utilização de memórias *flash*, mas em certas aplicações devido à outras limitações este tipo de recurso pode não estar disponível (MARWEDEL, 2011).

O peso dos componentes pode ser o fator decisivo na aplicação de um sistema embarcado, existem situações onde o sistema deve ser portátil, logo um peso elevado acabaria por dificultar o manuseio do protótipo. Nestes casos o projetista deve ter preferência pelos componentes de tamanho mais reduzido possível (MARWEDEL, 2011).

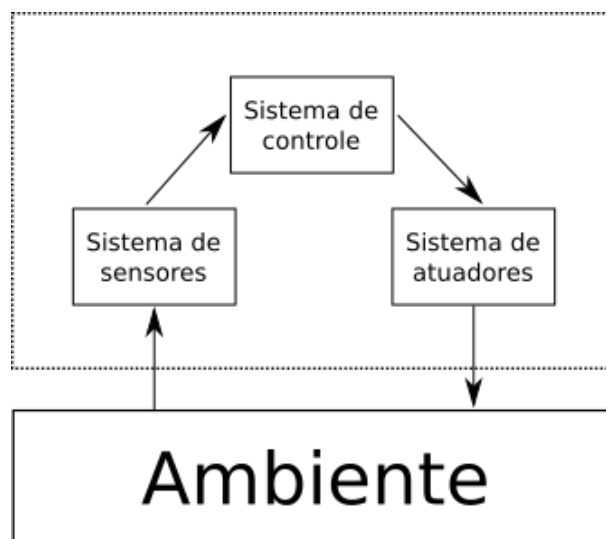
Em sistemas voltados para o mercado de consumo onde existe um orçamento limitado para o desenvolvimento do sistema, o custo passa a ser uma métrica crucial para o projeto, componentes que não melhoram de maneira significativa a eficiência do pior caso do sistema devem ser descartados a fim de reduzir o consumo total de energia. Os requisitos da aplicação devem ser cumpridos utilizando o menor número de componentes possível e com os componentes mais baratos possíveis desde que não comprometam a qualidade do produto final. (MARWEDEL, 2011)

Muitas vezes quando um sistema é desenvolvido em um ambiente com recursos limitados, se faz necessária a utilização de modelos formais para garantir que os recursos disponíveis sejam utilizados da melhor maneira possível. Geralmente o tamanho de código e o tamanho físico de um componente estão em conflito direto com o seu desempenho, o que pode gerar problemas quando um sistema embarcado precisa reagir em tempo real. Modelos formais contam com certos índices de desempenho que podem ser medidos com valores numéricos, o que permite aos desenvolvedores realizar otimizações precisas nas escolhas componentes que serão utilizados em um projeto. [Edwards et al. \(1997\)](#)

2.1.2 Sistemas de Tempo Real

Conforme [Buttazzo \(2011\)](#), um sistema em tempo real deve ter o tempo do sistema medido utilizando a mesma escala de tempo do mundo real, isto ocorre pelo fato de que o sistema deve ter ciência do ambiente no qual ele irá operar. [Buttazzo \(2011\)](#) realiza uma comparação entre sistemas biológicos e a velocidade das suas reações em seus habitats, um gato e uma tartaruga, por exemplo, podem ter velocidades de reação diferentes, no entanto em seus respectivos ambientes esta velocidade se mostra suficiente para a sua sobrevivência.

Figura 1 – Diagrama de um sistema em tempo real genérico.



Fonte: Adaptado de [Buttazzo \(2011\)](#).

Tal exemplo demonstra que o conceito de tempo não é natural aos sistemas, sejam eles biológicos ou artificiais, mas que na verdade está relacionado com o ambiente no qual estes sistemas irão atuar como apresentado na [Figura 1](#), que demonstra um modelo básico de como um sistema em tempo real deve ser reativo em relação ao ambiente no qual ele está implantado. Captando sinais através de sensores que são processados por sistemas de

controle que então realiza ações baseadas nestes dados através de atuadores (BUTTAZZO, 2011).

Em termos de processamento, a computação mais rápida reduz o tempo de resposta de um sistema, entretanto ela não necessariamente garante que o tempo de resposta de tarefas individuais será atingido de maneira correta, um sistema em tempo real não deve apenas ser rápido, ele deve ser previsível. Analisando os sistemas do ponto de vista de processos, os processos de um sistema em tempo real contam com um componente ausente em processos de sistemas normais, o chamado *Deadline*, que é o prazo máximo para a finalização de uma determinada tarefa (BUTTAZZO, 2011).

Em aplicações críticas o retorno de operações fora do *deadline* não é apenas atrasado, mas sim incorreto, o que pode ocasionar em perdas significativas dependendo da criticidade do sistema (BUTTAZZO, 2011). Em um contexto de um sistema de banco de dados de tempo real, a perda de deadlines pode acarretar em sobrecarregamentos em requisições ao banco de dados, o que tornaria o sistema inutilizável (HARITSA et al., 1991). A criticidade de uma aplicação depende das consequências ocasionadas devido ao atraso no tempo de resposta esperado do sistema, sendo este classificado em *Hard*, *Firm* e *Soft* (BUTTAZZO, 2011).

- Sistemas críticos *Hard* são aqueles onde caso não haja resposta no tempo definido podem ocorrer eventos devastadores, muitas vezes com perda de vidas;
- Sistemas críticos *Firm* são aqueles onde o atraso na resposta torna o sistema inútil, no entanto nenhum dano é gerado; e
- Sistemas críticos *Soft* são aqueles onde resultados após o tempo de resposta definido ainda tem alguma utilidade para o sistema, mesmo gerando perda de desempenho.

Usualmente os sistemas embarcados trabalham de maneira híbrida quanto a sua criticidade, onde certas atividades podem ser consideradas como *Hard* e outras podem ser *Soft* ou *Firm*. Atividades com criticidade *Hard* incluem: coleta de dados utilizando sensores, detecção de condições críticas, filtragem de dados, etc. Atividades que contam com uma criticidade *Firm* podem ser encontradas em aplicações de redes e multimídia, por exemplo: processamento de imagem *on-line*, execução de vídeos e decodificação de áudio e vídeo. Já atividades com criticidade *Soft* geralmente estão relacionadas à interação com o usuário como: a exibição de mensagens em uma tela, o processamento de sinais de teclado e o armazenamento de dados de utilização (BUTTAZZO, 2011).

2.1.3 Microcontroladores e Microprocessadores

As diferenças entre microcontroladores e microprocessadores são apresentadas por Ayala (1991) em seu livro, onde fica claro que apesar de terem surgido da mesma ideia e

serem fabricados pelo mesmo grupo de pessoas, seu funcionamento e aplicação diferem grandemente. Existem diferenças fundamentais no design de tais dispositivos, enquanto microcontroladores por si só contam com todos os componentes necessários para o seu funcionamento, microprocessadores necessitam de outros componentes (exemplo, memória) e periféricos para funcionar como um computador completo.

Microprocessadores são conhecidos popularmente como Unidades Centrais de Processamento (CPU, em inglês), e tem como sua função principal buscar e modificar extensivamente dados da memória para que sejam armazenados ou exibidos para o usuário. Por si só um microprocessador não constitui um microcomputador completo, para se tornar uma máquina capaz de executar programas de propósito geral se faz necessária a utilização de memórias RAM, memórias de armazenamento massivo e diversos dispositivos de entrada e saída externos (AYALA, 1991).

Nos microprocessadores o hardware destes componentes é desenvolvido de tal maneira a permitir o desenvolvimento de sistemas grandes ou pequenos dependendo da demanda da aplicação. O projeto do microprocessador é desenvolvido de modo a atender as suas expectativas no mercado de consumo em massa.

Um exemplo de microprocessador seria o Broadcom BCM2837 (RASPBERRY PI FOUNDATION, 2018a), que conta com quatro cores ARM que funcionam em uma frequência de 1.2GHz. Este processador é encontrado no Raspberry Pi 3 (RASPBERRY PI FOUNDATION, 2018b), exibido na Figura 2, que é um computador de placa única (SBC, em inglês). Por se tratar de um microprocessador, o Broadcom BCM2837 necessita de diversos outros componentes para funcionar como um computador completo.

Figura 2 – Raspberry Pi 3.



Fonte: [Raspberry Pi Foundation \(2018b\)](#).

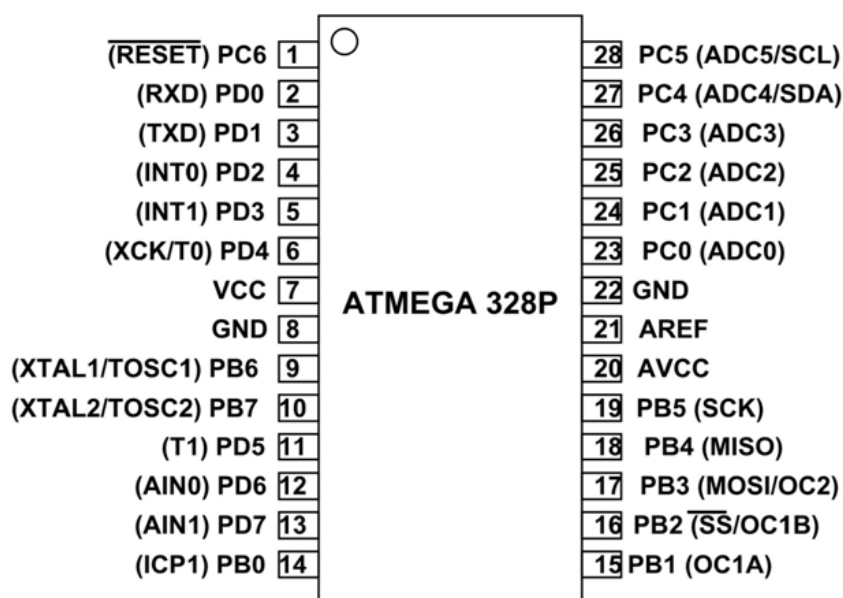
Os microcontroladores funcionam de maneira similar aos microprocessadores, no entanto, além de terem os componentes encontrados em um microprocessador eles também contam com todos os componentes necessários para o funcionamento de um computador completo, como memórias RAM, memórias ROM e portas paralelas e seriais de entrada e

saída. Como os microprocessadores, os microcontroladores também são componentes de propósito geral, mas o seu foco deixa de ser apenas computar os dados encontrados na memória e armazená-los, e passa também controlar o ambiente em que se encontram se baseando nos resultados do processamento dos dados disponíveis (AYALA, 1991).

Os programas utilizados em microcontroladores são armazenados na memória ROM e não tem seu funcionamento alterado durante o ciclo de vida do sistema. As instruções de máquina encontradas nos microcontroladores geralmente envolvem buscar dados na memória interna e que também realizam operações envolvendo os pinos de conexão inclusos na plataforma, o que permite que cada pino tenha seu propósito programado de acordo com a vontade do desenvolvedor (AYALA, 1991)

Um exemplo de microcontrolador seria o ATmega328P (ATMEGA328P, 2018), apresentado na Figura 3, que é o chip utilizado no Arduino Uno (ARDUINO, 2018). Por si só o ATmega328P conta com memória interna de 32 kilobytes para a gravação de programas de propósito geral, capacidade de controlar seus pinos e realizar cálculos complexos com os dados providos por eles. Quando integrado à uma placa de microcontrolador como o Arduino o desenvolvimento passa a ser muito mais simplificado sendo necessário apenas um cabo micro USB para a gravação de programas na memória.

Figura 3 – Pinagem do microcontrolador ATmega328P.



Fonte: ATmega328P (2018).

2.2 Modelagem e verificação de sistemas

Segundo [Edwards et al. \(1997\)](#), os sistemas atualmente, em sua maioria, são desenvolvidos com uma abordagem *ad hoc*, que é baseada em experiências prévias dos desenvolvedores com produtos similares aos que serão utilizados durante a implementação do sistema. Uma abordagem utilizando modelos formais e métodos de síntese automatizada de implementação é uma maneira mais efetivas de garantir um sistema seguro, visto que tais abordagens adotam modelos matemáticos para certificar propriedades de execução do sistema desenvolvido.

Um modelo é uma representação abstrata e simplificada de uma coisa real. No desenvolvimento de sistemas um modelo permite aos desenvolvedores a possibilidade de distinguir o que realmente é necessário e o que é supérfluo, além de permitir que seja realizada uma análise mais rápida do sistema do que se cada componente do sistema tivesse que ser analisado individualmente ([MILES, 2006](#)).

A utilização de um ou mais modelos formais para descrever os comportamentos do sistema antes de tomar decisões mais específicas da implementação, como por exemplo, qual plataforma de hardware ou qual linguagem de programação utilizar, podem reduzir consideravelmente as falhas durante o funcionamento do sistema ([EDWARDS et al., 1997](#)).

Estes modelos permitem que as propriedades do sistema sejam validadas de maneira simples, ainda durante as fases iniciais do desenvolvimento, através de ferramentas de simulação e verificação automatizadas, evitando eventuais entraves que podem comprometer a eficiência do sistema ([EDWARDS et al., 1997](#)).

2.2.1 Unified Modeling Language

De acordo com [Miles \(2006\)](#) a Linguagem de Modelagem Unificada (UML, em Inglês) é um padrão amplamente utilizado por desenvolvedores para modelagem de software e hardware de sistemas em geral. A UML é uma ferramenta muito útil durante o desenvolvimento de um sistema, pois permite aos desenvolvedores manter um certo controle sobre a complexidade do sistema, permitindo que as partes mais importantes do sistema se sobressaiam em relação às outras, evitando eventuais confusões durante a implementação.

Existem diferentes maneiras de utiliza a UML, alguns desenvolvedores a utilizam como um esboço do sistema frisando os componentes mais importantes para a implementação, alguns utilizam como planta baixa do sistema mantendo qualquer mudança realizada no código sincronizada com os modelos e outros utilizam como linguagem de programação, graças à natureza orientada a objetos da UML a conversão de diagramas em código de máquina passa a ser uma realidade no cotidiano de alguns desenvolvedores. ([MILES, 2006](#))

Diversos diagramas são utilizados para a especificação de um sistema. Cada um deles sendo utilizado para descrever alguma parte do modelo, sem conter informações demais relacionadas às outras partes do projeto o que poderia causar confusão durante a implementação do sistema. Abaixo estão listados os diagramas UML e suas respectivas funções. (MILES, 2006).

- **Diagrama de caso de uso:** utilizado para modelar interações entre usuário e sistema, também sendo usado para levantamento de requisitos.
- **Diagrama de atividade:** utilizado para modelar atividades sequenciais e paralelas do sistema.
- **Diagrama de classe:** utilizado para modelar classes, tipos, interfaces e as relações entre eles.
- **Diagrama de objeto:** utilizado para modelar as instâncias das classes definidas no diagrama de classes.
- **Diagrama de sequência:** utilizado para modelar as interações entre objetos quando a ordem delas é importante.
- **Diagrama de comunicação:** utilizado para modelar a maneira como os objetos interagem e as conexões necessárias para as interações.
- **Diagrama de tempo:** utilizado para modelar as interações entre objetos quando o tempo é uma preocupação importante
- **Diagrama de visão geral de interação:** utilizado para unir os diagramas de sequência, comunicação e tempo com a finalidade de ter uma visão geral do sistema.
- **Diagrama de estrutura composta:** utilizado para modelar os componentes internos de uma classe e descrever as relações entre objetos em uma dada situação.
- **Diagrama de componente:** utilizado para modelar os componentes importantes do sistema e as interfaces que eles utilizam para comunicação;
- **Diagrama de pacote:** utilizado para descrever a organização hierárquica de grupos de classes e componentes.
- **Diagrama de estados:** utilizado para modelar o estado de um objeto durante seu período de vida e os eventos que podem gerar mudanças de estados.
- **Diagrama de implantação:** utilizado para descrever como um sistema pode ser implantado em uma situação real.

2.2.2 Máquina de Estados

Segundo [Hopcroft et al. \(2001\)](#) máquinas de estados finitos, também conhecidas como autômatos finitos, são uma maneira muito útil e visual de modelar o comportamento de um sistema de maneira que seus componentes ou subsistemas estejam sempre realizando um conjunto de tarefas finitas que podem ser representadas e rastreadas através de estados que são alterados em resposta à entradas externas.

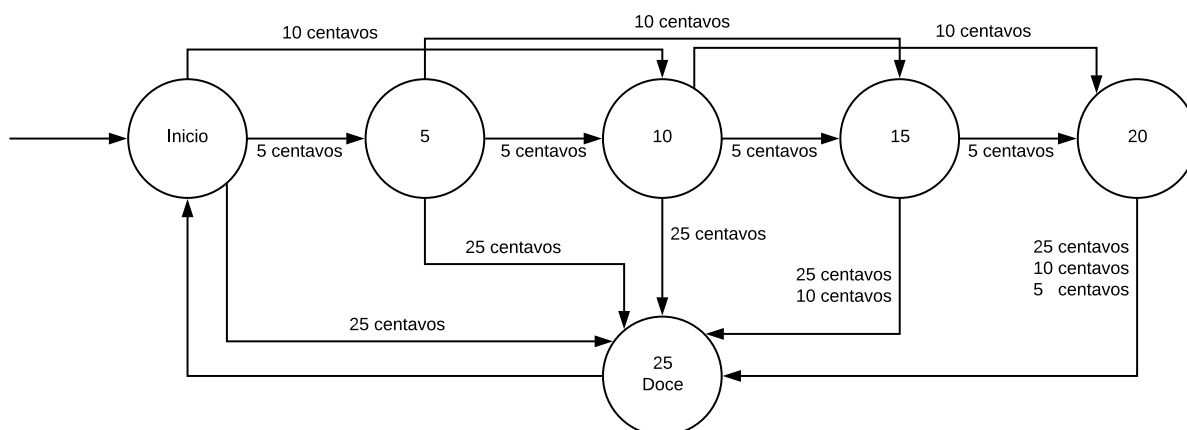
De acordo com [Hopcroft et al. \(2001\)](#), formalmente uma máquina de estados finitos é definida por uma quintupla $A = (Q, \Sigma, \delta, q_0, F)$ onde:

- Q é um conjunto de estados.
- Σ é um conjunto finito de símbolos de entrada.
- δ é uma função de transição que recebe um estado e um símbolo e retorna um outro estado.
- q_0 é um estado inicial que faz parte de Q
- F é um conjunto estados finais que é subconjunto de Q

Existem duas grande categorias de máquinas de estados, as determinísticas, onde apenas um estado é atingido por vez, e não determinísticas, que permite que vários estados podem estar sendo acessados ao mesmo tempo. ([HOPCROFT et al., 2001](#))

Um exemplo simples de modelagem de sistemas com máquinas de estados finitos seria a modelagem de uma máquina de doces apresentada na [Figura 4](#), onde cada moeda inserida pelo usuário realiza uma mudança de estado dependendo do valor da moeda. Quando o valor inserido excede os 25 centavos, a máquina retorna ao usuário um doce e retorna ao seu estado inicial até que sejam inseridos novos 25 centavos para a liberação de um novo doce.

Figura 4 – Máquina de estados finita de uma máquina de venda de doces simples.



Fonte: Adaptado de [Hopcroft et al. \(2001\)](#).

2.3 Internet das Coisas

No trabalho de [Atzori et al. \(2010\)](#) é demonstrado que em sua ideia fundamental a internet das coisas é um ambiente de computadores pervasivos que interagem um com os outros de maneira cooperativa para atingir metas em comum, este conceito permite a idealização de ambientes inteligentes onde todos os objetos estão em constante comunicação: para auxiliar o usuário a atingir uma qualidade de vida maior.

Nos ambientes residenciais e empresariais, a internet das coisas pode tornar a vida muito mais confortável e eficiente com sensores e atuadores, por exemplo, coletando dados sobre a preferência dos usuários e sobre o clima é possível que a temperatura seja adequada automaticamente levando em consideração o gasto de energia elétrica para uma maior economia. ([ATZORI et al., 2010](#))

2.4 Reconhecimento de Padrões

Segundo [Bishop \(2006\)](#) o problema de encontrar padrões em dados é um problema famoso e antigo que desde sempre intrigou cientistas mundo a fora. A observação de padrões na natureza serviu de fundação para a criação de modelos matemáticos que servem de base para a física moderna que até hoje é utilizada para resolver problemas reais.

Atualmente o campo de reconhecimento de padrões é focado na identificação automática de regularidades em grandes conjuntos de dados, utilizando algoritmos computadorizados. Os resultados destes algoritmos podem ser utilizados para classificar estas regularidades em diferentes grupos. ([BISHOP, 2006](#)). Vale ressaltar que com o uso de sistema em IoT uma grande quantidade de dados é gerado pelos sensores adotados, logo técnicas de reconhecimento de padrões podem ser uma solução para classificar e prever

determinadas ações nestes sistemas.

2.4.1 Aprendizagem de Máquina

Segundo [Russell e Norvig \(2010\)](#) o aprendizado ocorre quando um agente, seja ele uma máquina ou um humano, melhora seu desempenho em tarefas futuras após realizar observações do ambiente no qual ele atua se baseando em pares de entradas e saídas para prever novas saídas a partir de novas entradas.

A utilização do aprendizado de máquina se justifica pela necessidade de criar sistemas adaptativos, pois durante o desenvolvimento muitas vezes não é possível prever mudanças que podem ocorrer no ambiente no qual o sistema irá atuar, ou mesmo porque não é possível de maneira simples desenvolver um sistema que possa atender as necessidades da aplicação utilizando métodos tradicionais de programação ([RUSSELL; NORVIG, 2010](#)).

De acordo com [Russell e Norvig \(2010\)](#) existem três tipos de aprendizado que diferem no tipo de *feedback* informado ao agente que busca se aprimorar. No aprendizado não-supervisionado o agente busca aprender padrões nas entradas de dados sem que qualquer tipo de *feedback* seja informado de maneira explícita. O tipo mais comum de atividade deste tipo de aprendizado seria a criação de grupos a partir de entradas, por exemplo, um cachorro pode desenvolver uma noção sobre dias de passear ou dias de ficar em casa de acordo com o clima do dia sem que o seu dono tenha explicitamente dito que era a hora de passear.

No aprendizado por reforço o agente aprende a partir de incentivos ou punições que são aplicados de acordo com o resultado da predição de saídas a partir de novas entradas. Um exemplo deste tipo de aprendizado seria quando o dono de um cachorro tenta ensinar novos truques a ele oferecendo um biscoito quando ele realiza o truque de maneira correta, e falando com uma voz mais firme sem oferecer o biscoito quando ele realiza o truque de maneira incorreta. ([RUSSELL; NORVIG, 2010](#))

No aprendizado supervisionado, o agente observa pares de entradas e saídas e tenta criar novas funções para reproduzir estes resultados a partir de novas entradas. Um exemplo deste tipo de aprendizado acontece quando uma criança é ensinada sobre como realizar cálculos matemáticos simples e o professor apresenta exemplos de cálculos corretos esperando que a criança realize o mesmo cálculo com valores diferentes e resultados diferentes por si só, sendo informada se o resultado estava correto ou não ([RUSSELL; NORVIG, 2010](#)).

Um exemplo da aplicabilidade de algoritmos de aprendizado de máquina seria o reconhecimento de escrita. Devido a grande variedade de tipos de escrita, este problema conta com um alto grau de complexidade de implementação. Para este problema é utilizada uma imagem de 28 por 28 pixels como representação de um caractere escrito a mão, que é

apresentada para o algoritmo como um vetor composto por 784 números reais. A criação de um algoritmo utilizando heurísticas e regras feitas para casos específicos acabaria criando um grande volume de regras e exceções o que levaria a resultados ruins no reconhecimento. (BISHOP, 2006)

Resultados melhores podem ser atingidos utilizando algoritmos de aprendizado de máquina. Um grande volume de dígitos escritos a mão representados por um conjunto de vetores e um vetor com valores representando categorias manualmente escolhidas por um desenvolvedor são conhecidos como *training sets*, e são utilizados como referência para a realização de ajustes nos parâmetros de um algoritmo de aprendizado de máquina. A ideia por trás deste método é que o algoritmo passe a categorizar novos vetores que não fazem parte do *training set* original nas categorias definidas previamente. A habilidade de reconhecer estes novos padrões corretamente é conhecida como *generalização* (BISHOP, 2006).

2.4.2 Frameworks de aprendizado de máquina

Como o foco deste trabalho não é o desenvolvimento e a implementação de algoritmos de aprendizado de máquina, serão utilizados *frameworks* desenvolvidos em Python prontos para desenvolvimento.

2.4.2.1 Scikit-learn

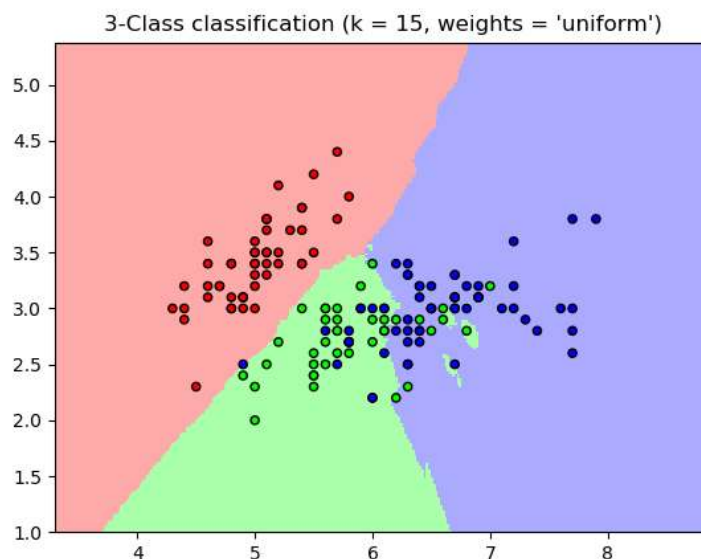
O *Scikit-learn* é um módulo de código aberto para a linguagem *Python* que conta com implementações do estado da arte de diversos algoritmos de aprendizado de máquina para análise de grandes volumes de dados. Por contar com código compilado a execução dos algoritmos acontece de maneira muito mais rápida que em outras plataformas *Python* que utilizam interpretadores para realizar cálculos complexos. (PEDREGOSA et al., 2011)

Graças à sua API simples integrada à linguagem *Python*, o *Scikit-learn* pode ser utilizado por diversos profissionais de diversas áreas sem conhecimentos avançados da implementação de algoritmos de aprendizado de máquina. A natureza padronizada da implementação dos algoritmos, permite que comparações de desempenho sejam facilmente realizadas. Algoritmos que são tidos como padrões na indústria como a *SVM* (CORTES; VAPNIK, 1995), *kNN* (COVER; HART, 1967) e *k-means* (MACQUEEN et al., 1967) estão implementados no módulo. (PEDREGOSA et al., 2011)

O algoritmo de classificação *k* Nearest Neighbors (kNN) está implementado no *Scikit-learn* e busca definir categorias baseadas em pesos definidos pelo usuário. O *Scikit-learn* implementa o algoritmo kNN como *KNeighborsClassifier*, que define categorias aos dados se baseado na quantidade *k* de vizinhos de cada valor definidos a partir de cada valor definido em uma base de dados. A Figura 5 apresenta o resultado do processamento

dos dados pelo algoritmo kNN, as três categorias de dados estão definidas pelas cores vermelha, verde e azul.

Figura 5 – Resultado da classificação de pontos em uma base de dados.



Fonte: Pedregosa et al. (2011).

2.4.2.2 TensorFlow

O TensorFlow é uma plataforma de aprendizado de máquina de código aberto desenvolvida por [Abadi et al. \(2016\)](#), possibilita ao desenvolvedor realizar o treinamento de um algoritmo em uma plataforma extremamente escalável podendo ser utilizado tanto em grandes *clusters* de servidores quanto em dispositivos móveis. As computações sendo realizadas no algoritmo e o estado no qual este se encontra podem ser visualizadas em tempo real através de uma interface visual em alto nível. Tal interface permite ao desenvolvedor tomar decisões rápidas sobre como otimizar o algoritmo que está sendo treinado. Diversos algoritmos de estado-da-arte estão implementados no TensorFlow como: Redes Neurais Recorrentes ([TENSORFLOW, 2018b](#)) e Redes Neurais Convolucionais ([TENSORFLOW, 2018a](#)).

Durante o trabalho de [Ertam e Aydin \(2017\)](#), foram utilizadas Redes Neurais Convolucionais (CNN, em inglês) e um classificador *SoftMax* implementados no TensorFlow para a identificação de escrita em uma base de dados contendo 60000 imagens de caracteres escritos a mão para serem utilizados como *training set* para os algoritmos. O reconhecimento de escrita foi realizado com sucesso em 98.43% dos casos apresentados como testes.

3 TRABALHOS CORRELATOS

Neste capítulo serão expostos trabalhos que contam com métodos de captura de movimentos de mão utilizando dispositivos com funções similares ao trabalho aqui proposto para a construção da luva Hand.io. Para uma melhor comparação, os trabalhos selecionados contam com sensores de acelerômetros e giroscópios como principal método de captura de sinais, assim como a luva Hand.io.

Os trabalhos aqui analisados demonstram a viabilidade do desenvolvimento da Hand.io e sua aplicação em um ambiente real. As principais particularidades da Hand.io seguem listadas abaixo e servem de parâmetro de comparação com as demais aplicações:

- Captura passiva de gestos;
- Sensores presos à mão;
- Aplicado em um ambiente residencial;
- Presença de sensor acelerômetro;
- Presença de sensor giroscópio;
- Conexão sem fio;
- Controle por infravermelho; e
- Personalização de gestos.

3.1 Accelerometer-based gesture control for a design environment

O trabalho de [Kela et al. \(2006\)](#) conta com dois estudos principais, sendo o primeiro a aplicação e o estudo da viabilidade da utilização de gestos em um estúdio de projetos inteligentes, em específico em aplicações de Desenho Assistido por Computador (CAD) e o segundo uma avaliação e comparação da interface por gestos em relação a outras modalidades como: controle por voz, objetos físicos utilizando RFID, uma tela de toque através de um *tablet*, e um dispositivo conhecido como IntelliPen, que funciona com um laser apontado para uma tela que realiza funções similares à um mouse. Foi utilizado um Modelo Oculto de Markov (HMM) ([RABINER, 1989](#)) a partir de sinais discretos para o reconhecimento dos padrões, o algoritmo conta com duas fases distintas de treino e de reconhecimento.

O dispositivo utilizado para captura de movimentos foi uma *SoapBox* ([TUULARI; YLISAUKKO-OJA, 2002](#)), que é uma placa miniaturizada do tamanho de uma caixa de

fósforos que conta com um processador, um giroscópio, um acelerômetro, um compasso eletrônico, e comunicação com e sem fios. O protótipo em questão conta com dois botões que são pressionados no início e no fim da realização de um gesto, além do modo de reconhecimento de gestos o protótipo conta com um modo de leitura contínua de movimentos, que pode ser acionado pressionando ambos os botões. Este modo realiza medições diretas dos valores fornecidos pelos sensores e os utiliza para ações de zoom e rotação de objetos virtuais 3D, permitindo que o usuário sinta que está segurando o objeto em suas mãos.

Os resultados do estudo apontam que existe um grande potencial para a utilização de controle por gestos em um ambiente de trabalho, em específico para o aumento da produtividade em determinadas atividades. De acordo com o grau de conhecimento dos participantes dos testes houveram diferentes preferências quanto a interface de utilização do estúdio inteligente, participantes com um grau maior de experiência com design se mostraram mais favoráveis a IntelliPen, devido a similaridade à um mouse, já os demais também demonstraram um interesse maior pela interface por gestos. O estudo conclui que a utilização de gestos combinados com outros métodos de controle pode ser benéfica para o aumento da produtividade em um ambiente de design, por permitir um controle mais natural e intuitivo de objetos 3D e facilitar a realização de atalhos em programas CAD.

Existem diversos pontos de divergência entre o trabalho apresentado e a Hand.io, sendo o mais predominante o fato dos sensores da Hand.io estarem presos diretamente à mão do usuário. A natureza vestível da luva Hand.io permite ao usuário controlar os dispositivos ao seu redor de maneira mais natural, sem que seja necessário um dispositivo a parte. [Kela et al. \(2006\)](#) demonstra que existe espaço para controle por gestos em diversos ambientes e que sistemas como a Hand.io podem aumentar consideravelmente a produtividade em ambientes de trabalho.

3.2 I'm home: Defining and evaluating a gesture set for smart-home control

[Kühnel et al. \(2011\)](#) apresenta uma metodologia extensa para definição e avaliação de gestos para controle de ambientes inteligentes, em específico casas inteligentes. O dispositivo escolhido para a realização da captura dos gestos foi um *Smartphone Iphone*, tal dispositivo foi escolhido devido a presença de um sensor acelerômetro e giroscópio e a sua grande disponibilidade. Os sinais enviados pelos sensores foram processados por um algoritmo Fast Dynamic Time Warping - FastDTW ([SALVADOR; CHAN, 2007](#)) que conta com uma complexidade de tempo de espaço linear. Foram realizados três estudos que buscavam definir um vocabulário de gestos simples e coesos, definir o grau de distinção entre diferentes gestos e o grau de dificuldade na memorização de cada gesto.

No primeiro estudo os participantes foram colocados em um quarto com diversos

dispositivos de maneira que todos os dispositivos controláveis estivessem no ângulo de visão dos voluntários em todos os momentos. As ações realizadas pelo sistema foram realizadas pelos experimentadores através de uma interface gráfica seguidos por uma descrição do comando executado. Os participantes então, foram perguntados qual gesto seria mais apropriado para o comando realizado, os movimentos foram gravados com duas câmeras e foram guardados para avaliação do tempo e articulação de cada gesto. Após a realização de cada gesto foi realizada uma pesquisa para saber a compatibilidade do gesto com a ação realizada e para saber a facilidade da execução do gesto em questão. Foram realizadas 23 ações diferentes que controlavam dispositivos diferentes.

Os resultados deste estudo apontam que a utilização de gestos pré definidos e gestos personalizados são a melhor abordagem no controle por gestos. A utilização de uma grande base de usuários durante o desenvolvimento facilita grandemente a concepção de gestos que possam ser simples e coesos para as atividades propostas. Experiências prévias com interfaces de controle, como a interruptores e reguladores rotativos, podem influenciar bastante o tipo de gesto escolhido pelos usuários. Foi constatado também que nem o tamanho do gesto nem o tempo necessário para a realização são boas referências para a distinção entre gestos realizados pelos usuários.

No segundo estudo foram exibidas gravações dos gestos realizados no estudo anterior e logo depois foram perguntadas quais as melhores ações poderiam ser realizadas a partir daquele gesto. O ideal seria que os gestos apresentados remetesse aos comandos para os quais eles foram criados. Os gestos que fossem corretamente relacionados às ações mais vezes seriam os mais adequados para a atividade em questão.

No estudo apresentado foi constatado que gestos mais simples tendem a ser relacionados com mais facilidade com a ação para qual estes foram criados. Movimentos intuitivos como para cima ou para baixo foram relacionados com mais facilidade à atividade que tinham sido propostos, gestos simbólicos como sinais de interrogação também tiveram bons resultados.

No terceiro estudo participantes que não tiveram relação nenhuma com os estudos anteriores. No primeiro momento os voluntários foram convidados a assistir um vídeo para cada gesto e sua respectiva ação. Cada gesto foi repetido 5 vezes por cada participante utilizando um *smartphone* que registrou os sinais dos gestos que vão ser utilizados como referência para um classificador. Os participantes então, foram perguntados qual o grau de compatibilidade entre o gesto e a ação relacionada a ele.

No segundo momento foram exibidos aos participantes somente as ações que cada gesto realizava em ordem aleatória e durante os 5 segundos seguintes foi requisitado que o gesto correspondente à ação fosse realizado. Caso o gesto fosse correto a apresentação continuaria, caso contrário, o gesto correto seria mostrado e a ação seria exibida ao final da apresentação, que continuaria até que todos os gestos fossem realizados corretamente.

Foram registradas a quantidade de erros para cada gesto.

O terceiro estudo conclui que os usuários conseguem distinguir gestos intuitivos de gestos menos intuitivos. Algo que foi comprovado pela quantidade de erros durante o teste de memorabilidade. Gestos mais simples e que rápidos são os mais indicados.

A metodologia de definição de gestos apresentada por Kühnel et al. (2011) e o estudo da viabilidade de cada um, servem de guia para a definição dos gestos que foram utilizados na Hand.io. Testes com grupos de voluntários com diversos graus de conhecimento foram realizados durante o desenvolvimento dos gestos e em um momento posterior a viabilidade dos gestos realizados foi medida.

3.3 uWave: Accelerometer-based personalized gesture recognition and its applications

O trabalho realizado por Liu et al. (2009) propõe um algoritmo de reconhecimento de gestos chamado *uWave*, que é baseado no algoritmo *Dynamic Time Warping* (DTW) (MYERS; RABINER, 1981), diferentemente de métodos estatísticos como o Modelo Oculto de Markov - HMM (RABINER, 1989) que requer um volume muito grande de amostras de referência, o *uWave* necessita de apenas uma amostra para realizar o reconhecimento de um gesto, algo que é muito vantajoso para a criação de gestos personalizados especificamente para cada usuário. O algoritmo foi criado para ser aplicado em qualquer dispositivo que conte com um sensor acelerômetro de três eixos, tal sensor foi escolhido devido à sua grande disponibilidade em dispositivos eletrônicos de consumo, o que aumenta consideravelmente a aplicabilidade do algoritmo em um ambiente real.

Liu et al. (2009) apresenta duas aplicações distintas do algoritmo afim de demonstrar a sua versatilidade e usabilidade em cenários reais, a primeira tem como objetivo a identificação simples e precisa de um usuário através de gestos personalizados, analogamente à uma senha alfanumérica. Nesta aplicação foi utilizado um controle de um Nintendo Wii (NINTENDO, 2018) como método de entrada de gestos. O estudo constata, através de uma pesquisa realizada com os usuários, que um gesto personalizado é um meio de identificação muito mais fácil de memorizar do que uma senha composta por vários caracteres.

A segunda aplicação é a utilização de gestos para navegação de um ambiente 3D, foi desenvolvida uma rede social baseada em compartilhamento de vídeos utilizando um *smartphone* da marca Motorola, que conta com um acelerômetro integrado. A aplicação permite que o usuário realize um remapeamento dos gestos para qualquer função específica do aplicativo, o que reforça a natureza de personalização do algoritmo.

Liu et al. (2009) discorre sobre os desafios encontrados em definir um vocabulário de gestos que possam ser intuitivos para a grande maioria dos usuários. Outro ponto

frisado é o fato do algoritmo depender apenas de um acelerômetro, o que dificulta no registro de certos gestos devido à falta de dados sobre a inclinação e o ângulo das forças realizada sobre o sensor, gerando uma certa confusão nos dados dos movimentos.

Diferentemente do uWave, a Hand.io utiliza irá utilizar um módulo MPU 6050 (INVENSENSE, 2018) que conta com um acelerômetro e um giroscópio integrados, o que permite que seja realizada uma medição mais precisa da direção dos movimentos realizados com a luva. Liu et al. (2009) demonstra que existe uma variedade enorme de algoritmos como o Dynamic Time Warping (DTW) (MYERS; RABINER, 1981), que utiliza métodos exatos de classificação de dados necessitando apenas de poucos valores no *training set* e o Modelo Oculto de Markov (HMM, em inglês) (RABINER, 1989), que utiliza estatística na classificação e necessita de um grande volume de dados no *training set*. Diferentes algoritmos apresentam diferentes resultados variando com *training sets* de tamanhos diferentes, logo na Hand.io foram realizados testes de desempenho com diversos classificadores até que se atinja um resultado ótimo.

4 MÉTODO PROPOSTO

Este capítulo apresenta as etapas do método proposto para o funcionamento do sistema computacional da luva Hand.io e está explicado como os objetivos deste trabalho serão satisfeitos.

4.1 Visão geral do método da Hand.io

A Hand.io é uma luva de controle que permite que quem a use realize gestos para controlar os dispositivos eletrônicos disponíveis em um dado ambiente. O projeto conta com uma luva de pano com um dispositivo preso ao tecido, que lê os movimentos da mão do usuário e uma central de processamento posicionada em um local estratégico, que recebe os sinais dos gestos através de uma conexão sem fio e executa as ações previamente definidas correspondentes a eles, conforme [Figura 6](#).

Figura 6 – Visão geral do protótipo.



Fonte: Elaborada pelo autor.

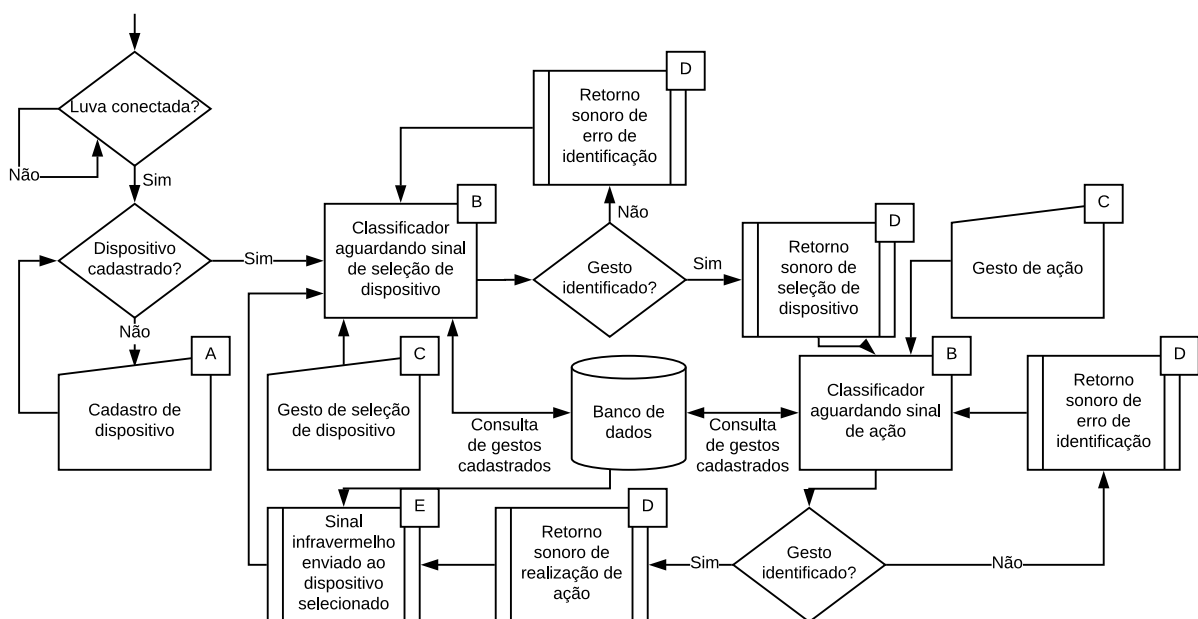
Os sinais dos movimentos são processados pela central utilizando algoritmos de aprendizado de máquina, que classificam o gesto realizado e realizam as ações correspondentes que estão definidas no banco de dados do sistema.

4.2 Fluxo de execução da Hand.io

A Figura 7, descreve o método proposto para a luva Hand.io através de um fluxograma, mostrando as diferentes etapas de funcionamento da luva de maneira sequencial. Os itens do fluxograma apresentado abaixo significam:

- A posição inicial do fluxograma é definida por uma **seta sem origem** a partir da qual são realizadas as checagens iniciais do sistema.
- Os **losangos** representam condições onde dependendo do resultado o fluxo do sistema pode ser alterado, no primeiro momento é verificada a conexão com a luva de controle e logo depois o sistema verifica se existem dispositivos cadastrados, caso não existam o sistema aguarda pela inserção de dispositivos no banco de dados.
- Os **trapézios** representam ações externas que são realizadas pelo usuário, seja a partir da luva ou a partir do dispositivo que será utilizado para realizar o cadastro de novos dispositivos.
- **Retângulos** representam processos internos do sistema e quadrados com barras laterais representam os processos predefinidos do sistema.
- O banco de dados que armazena os dados de gestos e códigos de infra-vermelhos de controle de dispositivos, está representado por um **cilindro** encontrado na parte central da figura.

Figura 7 – Fluxograma da Hand.io.



Fonte: Elaborada pelo autor.

4.3 Captura de Dados Baseado em Movimentos de Amplitude de Punho e Mão

A ideia principal da luva Hand.io é que o usuário controle seu ambiente utilizando gestos da maneira mais natural possível, para isso os movimentos do usuário são capturados constantemente e enviados para a central em tempo real até que seja identificado um gesto correspondente a um dispositivo cadastrado, como pode ser observado na [Figura 7](#).

A luva Hand.io utiliza dois sensores para realizar a captura dos movimentos da mão do usuário, um acelerômetro e um giroscópio. Estes dois sensores foram escolhidos visando evitar problemas encontrados no trabalho de [Liu et al. \(2009\)](#), exposto na [Seção 3.3](#), que sofre problemas de precisão devido a presença de apenas um sensor acelerômetro. Os sensores escolhidos realizam diversas amostras nas mudanças na aceleração e no giro realizados na luva, que serão enviados em tempo real à central de processamento.

4.3.1 Conexão com a central de processamento

A central de processamento (semelhante ao um pequeno aparelho de TV a cabo) inicia a sua operação esperando o recebimento de algum sinal da luva e até que a conexão tenha sido confirmada. A conexão entre a luva e a central é realizada utilizando o protocolo de redes sem fio IEEE 802.11 ([CROW et al., 1997](#)), conhecida popularmente como Wi-Fi, através de uma rede LAN. Este protocolo foi escolhido por ter uma boa relação custo benefício de implementação levando também em consideração velocidade de transmissão de dados e alcance. A necessidade de permitir que o usuário se movimente livremente em um ambiente requer que a conexão sem fio tenha um alcance amplo.

4.4 Reconhecimento de Padrões Baseado em Movimentos e Ações

Os sinais recebidos pela central de processamento servem de entrada para algoritmos de aprendizado de máquina, como ocorre no trabalho de [Liu et al. \(2009\)](#) encontrado na [Seção 3.3](#), que ficam em constante execução tentando classificar os sinais recebidos em categorias previamente escolhidas pelo desenvolvedor do sistema.

Os possíveis gestos reconhecidos pela luva estão definidos durante a fase de implementação e contaram com a ajuda de um grupo de voluntários que serviram de referência para o desenvolvimento de gestos simples e coesos, como os apresentados na [Figura 8](#). O trabalho de [Kühnel et al. \(2011\)](#) encontrado na [Seção 3.2](#) demonstra que um grande grupo de voluntários é fundamental para a criação de um vocabulário de gestos eficiente.

Gestos pré definidos e gestos criados pelos usuários são a abordagem mais efetiva para controlar um ambiente. No entanto uma quantidade muito grande de gestos, e

Figura 8 – Exemplos de gestos.



Fonte: Kela et al. (2006).

movimentos muito complexos não são a melhor escolha, pois existem uma quantidade limitada de gestos que podem ser lembrados com precisão sem que haja confusão durante a realização de um determinado gesto.

Neste método foi desenvolvido um extensivo *training set* com dados recolhidos de voluntários realizando os gestos. Estes dados são utilizados como modelo de treinamento para algoritmos de aprendizado de máquina, como os apresentados por (LIU et al., 2009) na Seção 3.3, que irão reconhecer os gestos realizados pelo usuário. Foi utilizado o *framework Python* Scikit-learn (PEDREGOSA et al., 2011), levando em consideração o tamanho do *training set* e a capacidade computacional da central de processamento.

Existem dois momentos durante o fluxo de execução da Hand.io onde estes algoritmos são utilizados, quando o sistema aguarda que o usuário selecione o dispositivo que ele deseja controlar e quando o sistema aguarda um comando que será realizado no dispositivo selecionado, ambos os momentos podem ser vistos respectivamente na Figura 7 na letra B. A cada vez que os algoritmos finalizam a classificação dos sinais tidos como entrada, o sistema retorna para o usuário um sinal sonoro de confirmação.

4.5 Modelo de Conexão Entre Dispositivos Eletrônicos

O controle dos dispositivos se dá utilizando um LED infravermelho, seguindo um padrão de controle remoto bem estabelecido pela indústria de eletrônicos de consumo. Cada fabricante define um protocolo de códigos infravermelhos específicos para cada dispositivo como o protocolo SIRC que é utilizado pela Sony em suas TVs (SONY, 2018) e o protocolo RC5 que é utilizado pela Philips (ATILUM, 2018). Protocolos diferentes foram criados visando impedir a interferência entre controles e dispositivos de fabricantes diferentes, como por exemplo o usuário tentar ligar o ar condicionado e acabar ligando a TV por acidente.

Os códigos de controle remoto de cada dispositivo cadastrado no sistema ficam armazenados em um banco de dados localizado na central de processamento. Cada código estará relacionado à um gesto específico que quando classificado pelo sistema será executado. Durante o cadastro de dispositivos, exibido na Figura 7 letra A, o sistema pede ao usuário que pressione os botões do controle remoto do dispositivo a ser controlado correspondentes às ações que ele deseja realizar por gestos. A intenção é que a base de dados dos sinais de

infravermelho seja populada ao ponto de que sejam criados perfis pré definidos para cada dispositivo, para que quando um dispositivo já presente na base de dados seja cadastrado, não seja necessário realizar o cadastro dos sinais manualmente.

4.6 Modelo de Prototipação

Os componentes utilizados para a confecção do protótipo da luva e da central buscam ter o menor custo possível, sem comprometer os recursos computacionais necessários para o funcionamento nominal do sistema, uma relação de preços do projeto pode ser vista na [Tabela 1](#).

Tabela 1 – Estimativa de preços dos componentes.

Componentes	Preços
Raspberry pi 3	R\$ 239,90
Sensor MPU-6050	R\$ 23,90
Módulo WiFi ESP8266-01	R\$ 26,90
Receptor Infravermelho TSOP4838	R\$ 6,90
LED Emissor Infravermelho	R\$ 0,90
Total	R\$ 298,50

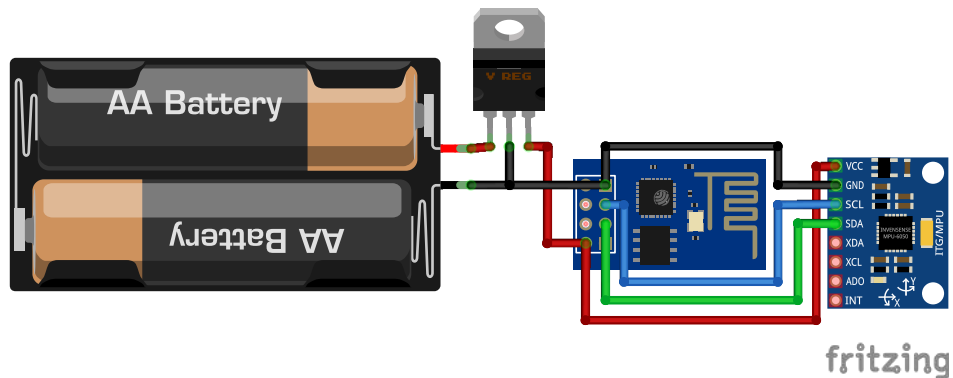
Fonte: [FilipeFlop \(2018\)](#).

O protótipo é composto pela luva em si que conta com sensores que irão capturar os movimentos do usuário apresentado na [Figura 9](#), e pela central de processamento que irá realizar as ações correspondentes aos gestos exibida na [Figura 10](#), nos dispositivos conectados.

A luva conta uma placa GY-521 que tem um sensor MPU-6050 ([INVENSENSE, 2018](#)), apresentado no canto direito da [Figura 9](#). Esta placa é utilizada em diversos projetos que utilizam captura de movimentos devido a sua boa relação custo benefício em relação a precisão dos sensores. O sensor MPU-6050 ([INVENSENSE, 2018](#)) conta com um acelerômetro e um giroscópio de três eixos cada, e um processador de sinais digitais integrado que pode ser utilizado para pré processar os sinais da luva antes que eles sejam enviados para a central de processamento, reduzindo a carga de trabalho realizada pela central.

A comunicação com a central de processamento ocorre utilizando uma placa Wi-Fi ESP-8266, segundo componente da direita para a esquerda na [Figura 9](#), que pode enviar sinais de maneira independente à qualquer computador conectado a uma rede LAN. Como fonte de energia a luva irá utilizar um conjunto de baterias que terão sua voltagem ajustada por um regulador, já que tanto a placa GY-521 quanto a placa Wi-Fi ESP-8266 podem funcionar com uma voltagem nominal de 3.3v.

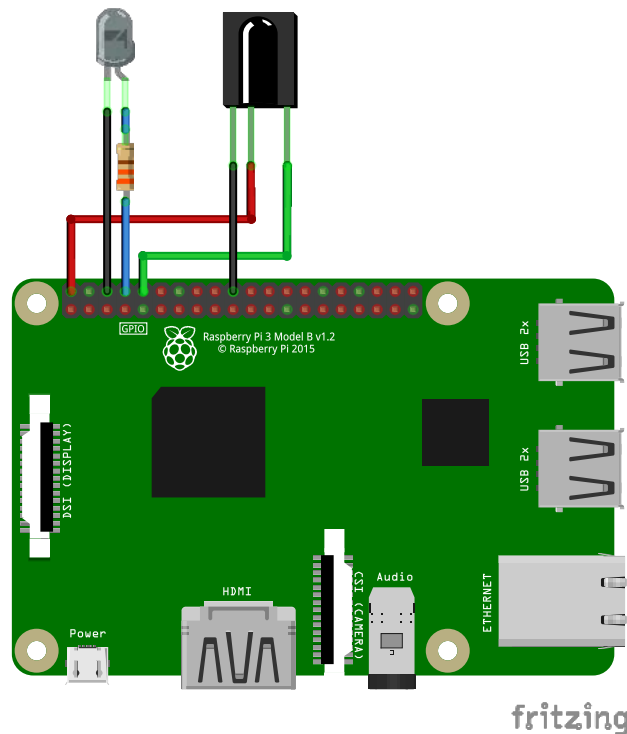
Figura 9 – Esquemático do protótipo da luva.



Fonte: Elaborada pelo autor.

Os componentes da central de processamento, apresentados na [Figura 10](#), são compostos por um Raspberry Pi 3 ([RASPBERRY PI FOUNDATION, 2018b](#)) que ficará conectado à internet esperando os sinais da luva e realizará o reconhecimento dos sinais, um receptor infravermelho TSOP4838 que serve para a realização do cadastro de novos dispositivos e um LED emissor de sinais infravermelhos que podem ser reconhecidos por dispositivo que utilizam este protocolo como meio de controle.

Figura 10 – Esquemático da central de processamento de sinais e execução de ações.



Fonte: Elaborada pelo autor.

O Raspberry Pi 3 ([RASPBERRY PI FOUNDATION, 2018b](#)) foi escolhido devido ao seu tamanho compacto e sua alta capacidade de processamento. Por se tratar de um computador completo em uma placa é possível instalar uma distribuição de Linux com capacidade de se conectar à internet com certa facilidade. Graças à sua unidade de processamento gráfico (GPU, em inglês) integrada, o Raspberry Pi 3 consegue executar algoritmos de aprendizado de máquina de maneira eficaz, frameworks como o TensorFlow conseguem tirar todo proveito destes componentes através de paralelismo.

5 AVALIAÇÃO EXPERIMENTAL

Este capítulo tem como objetivo apresentar o planejamento, execução e análise dos resultados obtidos a partir implementação do método proposto no sistema Hand.io. Esta avaliação teve como foco o funcionamento do protótipo da luva por meio de experimentações em cenários reais levando em consideração a eficácia e a eficiência do sistema em situações de utilização normal.

5.1 Análise da implementação

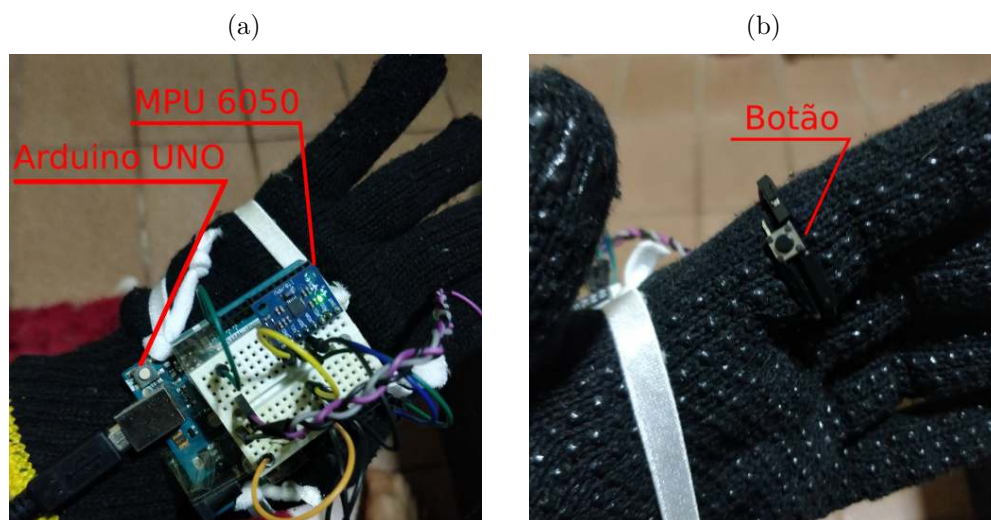
Nesta seção é detalhado e analisado de maneira extensiva as particularidades da implementação da Hand.io, tanto no projeto do *hardware* quanto de *software*. Como definido no método proposto no [Capítulo 4](#), o protótipo da Hand.io consiste em duas partes: a luva de captura de movimentos e a central de processamento de sinais que controla os dispositivos. Ambas as partes serão avaliadas de forma individual nas seções a seguir.

5.1.1 Protótipo da luva de captura de movimentos

A luva é composta por três principais componentes: microcontrolador Arduino UNO, que decodifica os sinais do sensor e os envia pela porta serial conectada à central de processamento (Figura 11a); um sensor MPU 6050, composto por um acelerômetro e um giroscópio, que é utilizado para realizar a leitura dos movimentos (Figura 11a); e um botão, acionado pelo usuário para definir o início e o fim da captura dos movimentos (Figura 11b). Os três componentes podem ser vistos na [11](#).

Neste protótipo adotou-se uma luva de pano para reduzir o contato do Arduino com a pele. O Arduino foi posicionado no dorso da mão de maneira firme, com fitas elásticas para evitar movimentações indesejadas que levariam à uma leitura imprecisa dos movimentos. O sensor MPU 6050 está fixado em uma *mini protoboard* presa no topo do Arduino com fitas adesivas de maneira estável. O botão de acionamento de captura de movimentos conta com os seus fios trançados, a fim aumentar a rigidez da construção do protótipo. Ao final da trança os fios são ligados ao botão de modo a formar um anel em volta do dedo indicador do usuário. Este posicionamento visa aumentar a ergonomia da luva, pois o botão estará convenientemente localizado no alcance do polegar do operador.

Figura 11 – Projeto da luva.

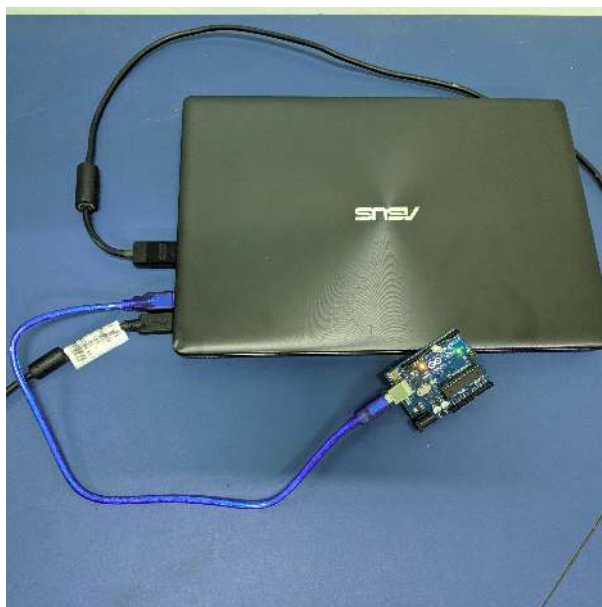


Fonte: Elaborada pelo autor.

5.1.2 Protótipo da central de processamento de sinais

O esquemático da central apresentado na [Figura 10](#) do [Capítulo 4](#) define que a central é composta por um computador de propósito geral equipado com um microprocessador, capaz de executar algoritmos de aprendizado de máquina e enviar e receber sinais infravermelhos. Componentes com as mesmas funcionalidades dos propostos inicialmente no método foram utilizados no protótipo final, de maneira que foi possível validar o método proposto sem que fossem realizadas mudanças significativas.

Figura 12 – Central de controle.



Fonte: Elaborada pelo autor.

O protótipo da central exibido na [Figura 12](#), é composto por um Notebook Asus com um processador *Intel Core I5*, 6GB de memória RAM, rodando o sistema operacional Manjaro Linux 18.0.4, conectado à luva através de uma conexão serial via cabo por uma porta USB, que processa os sinais recebidos e executa as ações correspondentes. Através de uma outra porta USB do Notebook, uma outra conexão serial ligada à um segundo Arduino UNO como atuador, é realizada. Este Arduino é a interface para emissão e recepção de sinais infravermelhos para o controle de dispositivos no ambiente.

5.1.3 Análise do software do protótipo Hand.io

O sistema é composto por três softwares que estão sendo executados simultaneamente de maneira independente no protótipo. Um no Arduino acoplado à luva, um no Arduino conectado à central e um na central de processamento. O código-fonte deste projeto está disponível no repositório online GitHub <<https://github.com/JohnPinto/Hand.io>>.

5.1.3.1 Análise dos softwares embarcados nos Arduinos

Os softwares em execução nos dois Arduinos funcionam de maneira distinta. Durante a modelagem do sistema, foi definido que o Arduino presente na luva funcionaria apenas como sensor, enviando os sinais dos movimentos para a central, enquanto o encontrado na central serviria de atuador, enviando os sinais infravermelhos para os dispositivos a serem controlados.

No decorrer do funcionamento normal do sistema o Arduino embutido na luva lê os sinais do sensor em tempo real. Enquanto o botão de captura de movimentos presente na luva não for pressionado pelo usuário, é enviado para a central apenas uma sequência de pontos (.) com um intervalo de 5ms entre os envios.

O comportamento, de envio intercalado de pontos (.) e sinais de movimento, serve de condição de marca parada para o classificador de movimentos encontrado na central que será detalhado nas seções a seguir. Foi utilizada a biblioteca `SoftwareSerial`¹ para a realização de comunicação serial.

Quando o botão na luva é pressionado, o sensor passa a enviar, o sinal do acelerômetro e do giroscópio, pela porta serial, formatado seguindo o padrão, representado na sequência de caracteres \$ ax ay az gx gy gz \n, onde as componentes x, y e z dos sensores são indicadas pelas letras a para acelerômetro e g para giroscópio, o \$, indica o início da leitura e o \n, uma quebra de linha, indica o fim.

O Arduino conectado à central que serve de atuador, permanece ocioso até que um comando indicando qual sinal infravermelho deverá ser enviado seja recebido. Os códigos infravermelhos ficam armazenados diretamente no Arduino, em função do desacoplamento

¹ <<https://www.arduino.cc/en/Reference/SoftwareSerial>>

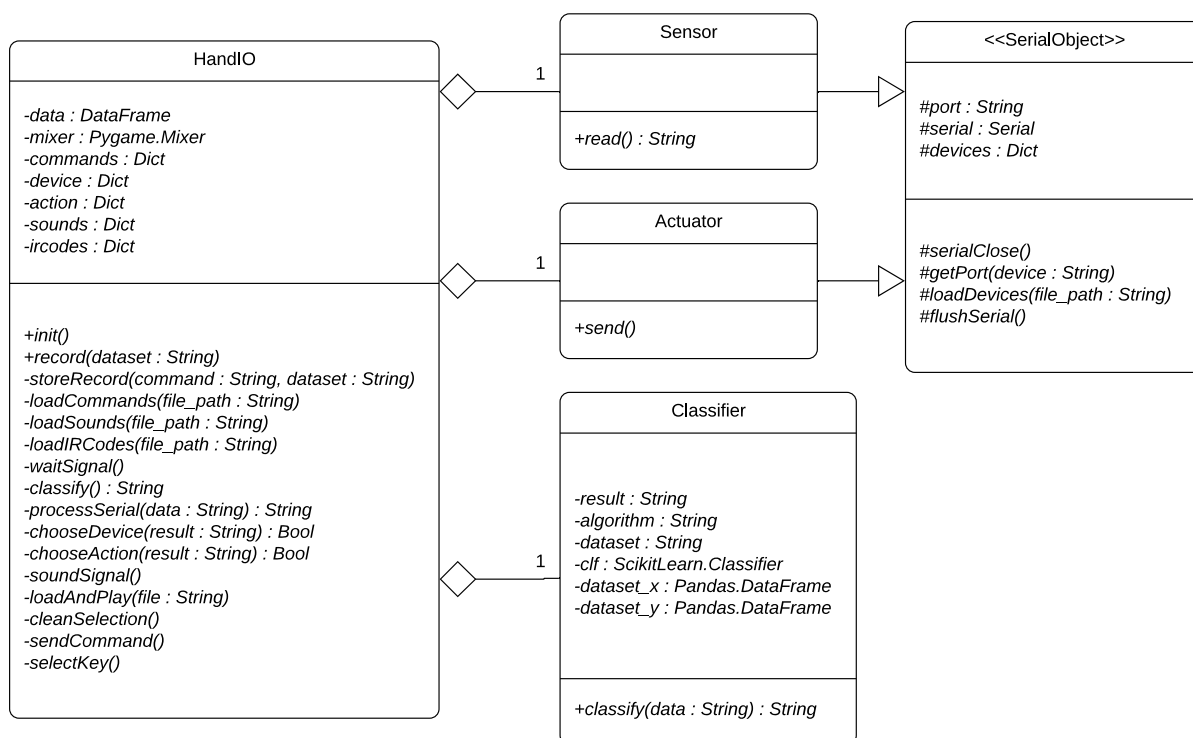
dos códigos da memória RAM para a flash², isto permite ao usuário realizar o cadastro de centenas de códigos infravermelhos de dispositivos diferentes.

Devido à natureza estacionária da central, é possível que múltiplos atuadores estejam inseridos em diversos cômodos diferentes de uma casa, por exemplo, o que serviria de aplicação para este trabalho. Na implementação deste software foram utilizadas as bibliotecas SoftwareSerial¹, para comunicação serial, e IRremote³, para envio de sinais infravermelhos a partir do Arduino.

5.1.3.2 Análise do software da central de processamento

O software em execução na central de processamento tem a implementação detalhada na Figura 13, onde é apresentado o diagrama de classes que define as especificidades do sistema e detalha as relações entre as classes que o constituem.

Figura 13 – Diagrama de classe da Hand.io.



Fonte: Elaborada pelo autor.

O código-fonte foi desenvolvido na linguagem de programação Python 3, utilizando conceitos de programação orientada a objetos, como classes e herança, visando compartimentalizar o código com o objetivo de aumentar a previsibilidade, e evitar retrabalhos por parte do desenvolvedor.

² <<https://www.arduino.cc/reference/en/language/variables/utilities/progmem/>>

³ <<https://github.com/z3t0/Arduino-IRremote>>

A linguagem em questão foi escolhida por sua praticidade de implementação por causa da sua natureza de alto nível, o que permite uma implementação rápida e simples em comparação com as demais linguagens. Outro fator motivador é a presença de uma vasta gama de bibliotecas externas que agilizaram o desenvolvimento do protótipo, como o Scikit-learn⁴ que conta com uma série de classificadores que utilizam algoritmos de aprendizado de máquina já implementados, e a PySerial⁵ que realiza o interfaceamento entre a linguagem Python e as portas seriais que conectam o sistema aos sensores e atuadores.

No diagrama de classes da Figura 13 a `HandIO` é a classe principal do sistema. Esta classe conta com definições de sensor, atuador e classificador. Ambos, os sensores e atuadores, implementados nas classes de nomes correspondentes, herdam da classe abstrata `SerialObject`, com a diferença de que o sensor tem a capacidade de enviar sinais e o atuador apenas de recebe-los. Internamente a classe `SerialObject` utiliza a biblioteca PySerial, para a transmissão de dados.

A classe `HandIO` conta com diversos dicionários que agem como uma especie de banco de dados do sistema, esta abordagem foi escolhida para que haja a possibilidade do desenvolvedor do sistema modificar ou adicionar novos gestos ou dispositivos de maneira simplificada. Todos os dicionários utilizados são carregados a partir de seus arquivos `.json` correspondentes que podem ser encontrados na pasta `json/` da raiz do sistema. Foram implementados métodos privados que iniciam com a palavra `load`, que carregam cada dicionário individualmente, estes métodos são chamados no construtor do objeto.

O classificador, implementado na classe `Classifier`, utiliza a biblioteca Scikit-learn. Esta biblioteca foi escolhida por suprir as necessidades do protótipo, e contar com uma API de utilização bastante simplificada. Os datasets com registros de movimentos utilizados para geração do modelo de classificação dos gestos ficam armazenados na pasta `datasets/` do programa.

A geração destes datasets se dá utilizando o método público `record()` da classe `HandIO`, que pede que o usuário da luva insira uma classe, que define qual gesto será registrado, e realize a captura de 10 gestos. Este procedimento se repete até que o usuário pare o programa. Para o protótipo foi levantado um `dataset` com 360 leituras recolhidas de nove voluntários diferentes, distribuídas entre quatro classes de gestos correspondentes à movimentos nas direções cima, baixo, direita e esquerda. Os voluntários foram orientados a realizar os gestos com a palma da mão voltada para a direita. Estes gestos foram escolhidos devido ao seu baixo grau de complexidade o que facilitaria os experimentos detalhados nas seções seguintes.

Para a definição de qual classificador seria o mais eficaz para a classificação de gestos, foi realizado um comparativo, com os principais classificadores oferecidos pela

⁴ <<https://scikit-learn.org/>>

⁵ <<https://github.com/pyserial/pyserial>>

biblioteca Scikit-learn, sendo estes: a *Logistic Regression* (LR) (SCIKIT-LEARN, 2019i), a *Linear Discriminant Analysis* (LDA) (SCIKIT-LEARN, 2019h), o *K-Neighbors Classifier* (KNN) (SCIKIT-LEARN, 2019g), o *Decision Tree Classifier* (CART) (SCIKIT-LEARN, 2019c), a *Gaussian Naive Bayes* (NB) (SCIKIT-LEARN, 2019e), a *Support Vector Machine* (SVM) (SCIKIT-LEARN, 2019k), o *Ada Boost Classifier* (ADB) (SCIKIT-LEARN, 2019a), o *Random Forest Classifier* (RFC) (SCIKIT-LEARN, 2019j), o *Extra Trees Classifier* (ETC) (SCIKIT-LEARN, 2019d), e o *Gradient Boosting Classifier* (GBC) (SCIKIT-LEARN, 2019f).

Este comparativo foi realizado utilizando validação cruzada (SCIKIT-LEARN, 2019b), técnica que divide o dataset em grupos de treinamento e grupos de teste. O grupo de treinamento é utilizado para a criação do modelo de classificação que serve de referência para classificações futuras, e o grupo de testes é utilizado como entrada para o classificador, esperando que a classificação seja bem sucedida, já que ambas as partes fazem parte de um mesmo dataset. Para uma medição mais precisa foi utilizada a técnica conhecida como *K-fold* (SCIKIT-LEARN, 2019b), que realiza K testes de validação cruzada pegando K partes, como treinamento e teste, a cada K interações. Para a obtenção de resultados consistentes foi utilizado um $K = 10$.

Após a execução do comparativo, os resultados constatados na Tabela 2, indicam que o classificador com melhor desempenho foi o *Gaussian Naive Bayes* (NB), que obteve uma acurácia de 81.1% durante a validação cruzada, sem que houvesse uma perda considerável na pontuação de treino no decorrer do aumento da quantidade de registros de movimento no *dataset*.

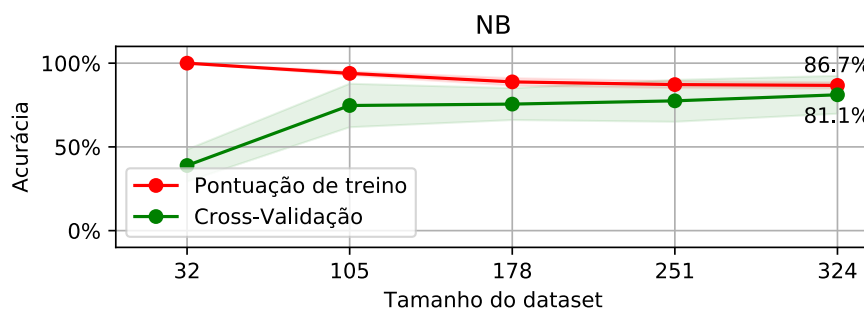
Tabela 2 – Comparativo dos classificadores.

Classificador	LR	LDA	KNN	CART	NB
Acurácia	62.7%	73.8%	77.5%	71.3%	81.1%
Classificador	SVM	ADB	RFC	ETC	GBC
Acurácia	79.7%	49.1%	77.7%	78.8%	74.1%

Fonte: Elaborada pelo autor.

Apesar do gráfico do NB, apresentado na Figura 14, com 32 valores valores no *dataset* ter atingido uma pontuação de treino alta em relação à uma quantidade maior, o modelo gerado com apenas esta quantidade de valores não é genérico o suficiente para classificar os movimentos de um segundo usuário com precisão, por isso é necessária uma quantidade maior de leituras realizadas por usuários diferentes, o que justifica a escolha deste classificador como o classificador padrão deste sistema.

Figura 14 – Classificador Gaussian Naive Bayes.



Fonte: Elaborada pelo autor.

5.1.4 Fluxo de execução do Hand.io

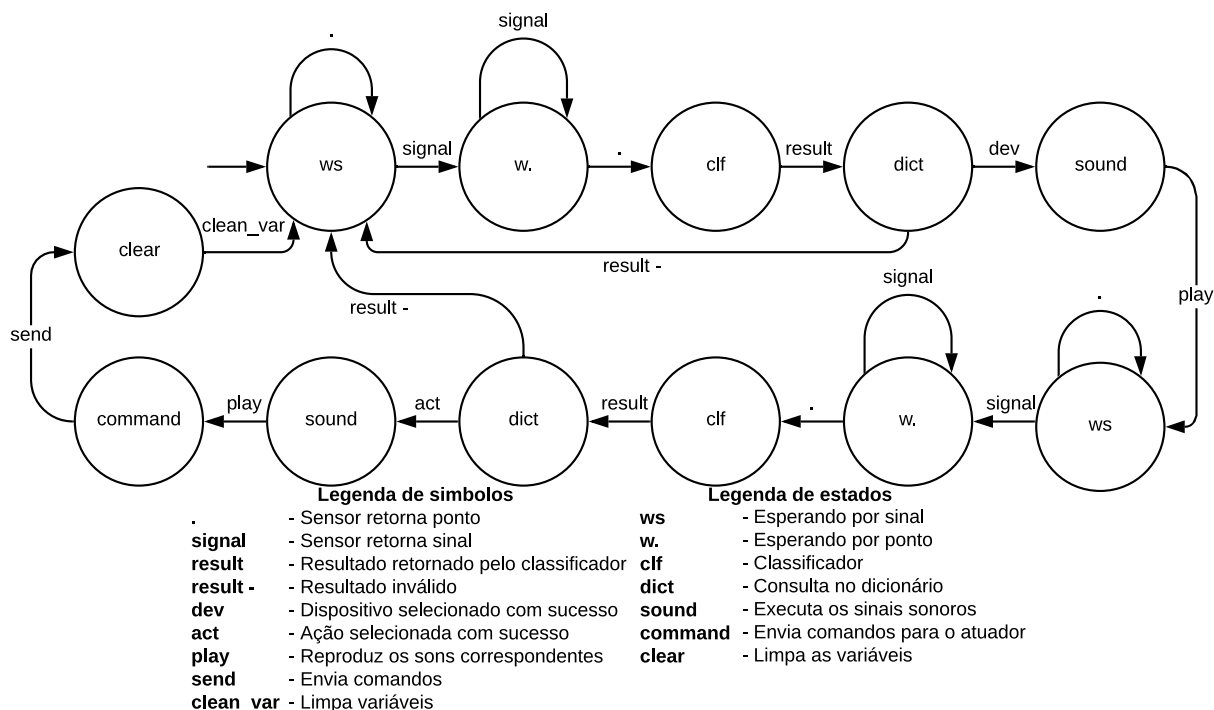
O sistema conta com duas grandes fases apresentadas na máquina de estados da Figura 15. A fase de seleção de dispositivo, definida na parte superior da figura, e a fase de seleção de ação, na parte inferior. O fluxo de execução será abordado do ponto de vista da máquina de estados, realizando anotações sobre quais métodos das classes, expostos na Figura 13, são chamados durante um dado estado. Na classe `HandIO` está implementado o método público `init()`, que é responsável pelo funcionamento nominal da central. Os procedimentos dentro deste método, refletem o comportamento definido na máquina de estados apresentada anteriormente.

Durante os estágios iniciais da primeira e da segunda fase, os estados `ws` e `w.`, que foram implementados no método `waitSignal()`, o sistema fica aguardando os sinais de início e parada de captura de movimentos. Em um primeiro momento, ao receber pontos pela serial, a central permanece no estado `ws`, quando for constatado o recebimento de um sinal de movimento, o sistema passará para o estado `w.`

O programa permanecerá neste estado até que voltem a ser recebidos pontos, para que, então, o ultimo sinal recebido seja salvo no atributo `data`, e enviado para o classificador no estado `clf` que chama o método `classify()`. Após o retorno da classificação, o sistema passará para o estado `dict`, onde os resultados serão avaliados.

Neste estado o resultado da classificação serve como entrada para o método `chooseDevice()` ou para o método `chooseAction()`, dependendo da fase que o programa se encontra. Nestes métodos o resultado é utilizado como chave no dicionário `commands` que retorna o dispositivo ou ação correspondente ao gesto. Caso o dicionário retorne um sinal válido o sistema então passa para o estado `sound`, que chama o método `soundSignal()` caso o retorno seja inválido o sistema retornará ao estado inicial. Durante a segunda fase do programa os possíveis retornos do dicionário ficam restritos às possíveis ações que dispositivo previamente selecionado tem mapeados pelo sistema.

Figura 15 – Máquina de estados da Hand.io.



Fonte: Elaborada pelo autor.

O estado **command**, implementado no método `sendCommand()`, analisa a ação selecionada no estado anterior e as realiza no dispositivo selecionado. Ao final das duas fases o sistema entra no estado **clear**, que limpa a seleção de dispositivo e gesto, para que um novo comando seja realizado.

5.2 Planejamento e projeto do cenário experimental

Um cenário, no qual serão realizados os experimentos, foi criado para avaliar a capacidade da Hand.io de controlar dispositivos em um ambiente real. Será utilizado, nesta avaliação, o protótipo discutido na seção anterior deste capítulo. Como referência do progresso dos experimentos, as questões experimentais levantadas abaixo serão respondidas conforme o andamento dos experimentos:

QE1 - Qual a taxa de sucesso do sistema Hand.io em detectar os gestos corretamente e enviar sinais aos dispositivos correspondentes?

QE2 - Um novo usuário do sistema Hand.io, usando um roteiro, consegue controlar os dispositivos de um dado ambiente?

QE3 - Qual a avaliação de um novo usuário sobre o estado atual do protótipo do sistema Hand.io?

O cenário experimental, apresentado na [Figura 16](#), conta com os seguintes dispositivos eletrônicos: um ar-condicionado, do fabricante Carrier de 36.000 *BTU/h*; uma TV da marca Philips de 32 polegadas; e um software para apresentação de slides em um *notebook*, definido na [subseção 5.1.2](#). Para controle de dispositivos que contam com uma interface de controle por meio de sinais infravermelhos, como é o caso do ar-condicionado e da TV selecionadas, é necessário que os sinais enviados pelos controles remotos destes dispositivos sejam capturados e carregados no Arduino que serve como atuador, no sistema Hand.io, esta etapa é feita durante o processo de instalação.

Para controle da apresentação de slides, não é necessária intervenção por parte do instalador do sistema, pois o sistema já possui as funcionalidades para este tipo de operação, tendo os comandos do teclado que realizam o controle da apresentação já mapeados. O protótipo da Hand.io está localizado no alcance dos receptores infravermelhos dos dispositivos. O usuário que realizará os experimentos ficará em uma posição central no cenário.

Figura 16 – Cenário experimental.



Fonte: Elaborada pelo autor.

Diante do proposto, foram realizados três experimentos, medindo a eficácia e execução do sistema proposto de maneiras diferentes. Com a finalidade de responder a primeira questão experimental (**QE1**), foi realizado um primeiro experimento onde um voluntário, que teve os seus dados de movimentos previamente cadastrados no sistema, reproduziu duas situações e teve a sua taxa de sucesso registrada quantitativamente.

Para a análise da segunda questão experimental (**QE2**), um grupo de voluntários, que nunca utilizou o sistema e não teve seus dados de movimentos previamente registrados, foi instruído a seguir um roteiro escrito de uso do Hand.io e teve a sua taxa de sucesso

registrada para avaliação posterior. A descrição do experimento e seus resultados são apresentados na [seção 5.3](#).

O grupo anteriormente descrito foi utilizado em um outro experimento, a fim de responder a terceira questão experimental (**QE3**), após a execução do roteiro, os participante responderam um questionário, composto por perguntas sobre o estado atual do sistema proposto. Adicionalmente, foram coletadas sugestões dos participantes do experimento visando identificar quais possíveis melhorias poderiam ser feitas para a ergonomia do sistema a partir do ponto de vista de um eventual usuário final.

5.3 Execução dos experimentos e análise dos resultados

Durante esta seção será detalhada a execução dos experimentos, e os seus resultados serão analisados com a finalidade de responder as questões experimentais levantadas na seção anterior.

5.3.1 Teste com um usuário experiente

Após a execução do primeiro experimento, os resultados foram registrados e estão expostos na [Tabela 3](#), onde a primeira coluna representa o número de vezes que o experimento proposto foi executado. A segunda e a terceira denotam os resultados das situações 1 e 2, respectivamente. Na situação 1, o usuário do Hand.io foi instruído a ligar e desligar sequencialmente a TV e o ar-condicionado presentes no cenário experimental. Já na situação 2, o usuário foi instruído a avançar ou retroceder a apresentação de slides reproduzida na TV. Cada resultado de um experimento n está assinalado com um **OK**, para bem sucedido, e **F** para mal sucedido.

Tabela 3 – Dados coletados do experimento com um usuário experiente.

n	Situação 1				Situação 2	
	Ligar TV	Ligar AC	Desligar TV	Desligar AC	→	←
1	OK	OK	OK	OK	OK	OK
2	OK	OK	OK	OK	OK	OK
3	OK	OK	OK	OK	OK	OK
4	OK	OK	OK	OK	OK	OK
5	OK	OK	OK	OK	OK	OK
6	OK	OK	OK	F	OK	OK
7	OK	F	OK	OK	OK	F
8	OK	OK	OK	OK	OK	OK
9	OK	F	OK	OK	OK	OK
10	OK	OK	OK	OK	OK	OK
Taxa de sucesso	100%	80%	100%	90%	100%	90%
Taxa média de sucesso: 93.3%						

Fonte: Elaborada pelo autor.

A partir do total de execuções do sistema distribuídos entre as situações 1 e 2, foi encontrada uma taxa média de sucesso de 93.3%, o que apresenta o funcionamento correto do sistema. Apesar da taxa de falha ter sido pequena, ela não é desprezível em um sistema desenhado para um usuário final, logo se faz necessário um aprimoramento do classificador, a fim de reduzir ainda mais esse percentual. Os resultados apresentados por este experimentos servem de resposta para a questão experimental **QE1**.

5.3.2 Teste com voluntários inexperientes

Um grupo com cinco voluntários que não tiveram contato anterior com o sistema, foi instruído a seguir um roteiro com a finalidade de avaliar a usabilidade e funcionamento da Hand.io. Os resultados deste experimento estão expostos na [Tabela 4](#), onde **V1**, **V2**, **V3**, **V4**, **V5** representam os voluntários que participaram do experimento e a primeira coluna as ações propostas para o experimento.

O roteiro com as ações propostas para o experimento é composto por 6 passos, onde é solicitado, respectivamente, aos voluntários: Ligar o ar-condicionado presente no cenário experimental (**1**); ligar a TV (**2**); avançar cinco slides em uma determinada apresentação (de **3.1** à **3.5**); voltar dois slides do passo anterior (**4.1** e **4.2**); desligar a TV (**5**); e desligar o ar-condicionado (**6**).

Tabela 4 – Dados coletados do experimento com voluntários inexperientes.

	V1	V2	V3	V4	V5
1	OK	OK	OK	OK	P ¹
2	P ¹	OK	OK	OK	OK
3.1	P ¹	OK	OK	OK ³	OK
3.2	OK	OK	OK	OK ³	OK
3.3	OK	OK	OK	OK ³	OK
3.4	OK	OK	OK	OK ³	OK
3.5	OK	OK	OK	OK ³	F
4.1	P	OK	OK	P	OK
4.2	OK	OK	OK	OK	OK
5	OK	OK ²	OK	OK	OK
6	OK	OK	OK	OK	OK
Taxa de sucesso	72.7%	100%	100%	90.9%	81.8%
Taxa média de sucesso: 89.08%					

¹ Problemas por falta de familiaridade com o sistema.

² Dificuldade em lembrar dos gestos.

³ Problemas com o botão de acionamento de leitura.

Fonte: Elaborada pelo autor.

Os resultados foram quantificados utilizando **OK** para bem sucedido; **P** para sucesso parcial, onde foi permitida uma tentativa para a realização do comando; e **F** para

falha na execução. Quando houveram problemas durante a execução dos experimentos, o aplicador do experimento realizou anotações quanto à possível razão.

Após a análise dos resultados deste experimento foi constatada uma taxa média de sucesso de 89.08%, considerando apenas os testes bem sucedidos. Apesar do desempenho ter sido inferior em relação ao experimento anterior, neste caso vale salientar que os voluntários não tiveram qualquer contato prévio com o sistema e não tiveram seus dados de movimentos inseridos no classificador, o que demonstra que o modelo obtido a partir do *dataset* inicial, se mostra genérico o suficiente para ser utilizado por terceiros. Logo, o modelo de previsão de movimentos não foi calibrado para os novos usuários, assim para a versão final pretende-se propor na instalação do sistema uma calibragem do sistema proposto.

5.3.3 Aplicação do questionário ao grupo de voluntários

Após a realização do experimento anterior o grupo de voluntários respondeu à um questionário que avalia o sistema como um todo. As perguntas realizadas podem ser vistas na lista abaixo:

- Q1** - O quão satisfeito você está quanto à eficácia do sistema? (de 0 a 10)
Q2 - Como você avalia os gestos utilizados para controlar o ambiente? (de 0 a 10)
Q3 - Qual o seu interesse em adquirir uma versão finalizada do sistema? (de 0 a 10)
Q4 - O quê você acha que poderia ser aprimorado no sistema?

As primeiras três perguntas **Q1**, **Q2**, e **Q3**, fazem uma análise quantitativa do sistema, onde 0 seria uma resposta completamente negativa e 10, completamente positiva. Já a **Q4** espera uma resposta mais subjetiva. Os resultados do questionário está expostos na [Tabela 5](#).

Tabela 5 – Dados coletados do questionário realizado.

	V1	V2	V3	V4	V5	Média
Q1	10	10	10	10	9	9.8
Q2	9	10	9	9	10	9.4
Q3	10	10	10	10	10	10

Fonte: Elaborada pelo autor.

A média das respostas da primeira pergunta (**Q1**), de 9.8 aponta que o sistema em seu estado atual atende às expectativas dos usuários. Este resultado positivo fortalece a ideia proposta por este trabalho, que apesar de ser um protótipo, já recebe avaliações positivas em relação ao seu funcionamento. Já a segunda pergunta (**Q2**) contou com uma média de resposta de 9.4, a menor entre as respostas.

Isso pode ser explicado pela falta de familiaridade dos voluntários em relação a sistemas de controle por movimentos, um problema que seria resolvido caso os voluntários tivessem mais tempo de uso do sistema, ou pela baixa variedade de gestos, isso significa que devem ser feitos mais testes quanto à definição dos gestos. As respostas da terceira pergunta (**Q3**) teve média 10, a mais alta de todas. Isso demonstra que existe mercado para um eventual produto final que possa surgir a partir deste trabalho.

A quarta pergunta (**Q4**) gerou respostas diferentes de cada voluntário, o primeiro voluntário respondeu que seria interessante que a luva deixe de utilizar um botão para definir o início e o fim dos gestos e passe a realizar uma leitura contínua dos movimentos. Esta é uma abordagem interessante, no entanto o grau de complexidade em definir os momentos onde o usuário deseja realizar comandos é alto, talvez com um estudo mais aprofundado este tipo de funcionalidade possa ser possível, mas isso está fora do escopo deste trabalho.

O segundo voluntário sugeriu que pudessem ser realizados mais de um gesto por dispositivo selecionado, isto é, ligar e mudar de canal ao selecionar a TV, sem que fosse preciso selecionar novamente a TV entre cada umas das ações citadas. Esta funcionalidade pode ser implementada no sistema, no entanto seria necessário que o desenvolvedor criasse uma solução simples para esta mecânica sem que a complexidade na utilização aumentasse.

Foi sugerido pelo terceiro voluntário que caso um dispositivo tivesse sido selecionado por acidente, houvesse a possibilidade do usuário retornar para a seção de seleção sem que um comando seja realizado. Assim como na sugestão do segundo voluntário é necessário que o desenvolvedor faça um levantamento de como implementar esta funcionalidade sem reduzir a eficácia do sistema.

A resposta dada pelo quarto voluntário sugere que se adicionassem mais opções de gestos para que mais dispositivos possam ser controlados e mais comandos possam ser realizados. Na implementação atual do sistema isso já é possível, mas um aumento na quantidade de gestos poderia dificultar na memorização por parte do usuário de qual gesto realiza qual ação. Seria necessária a realização de um estudo afim de definir qual o número ideal de gestos poderiam ser inseridos sem que o usuário começasse a se confundir.

O quinto e último voluntário respondeu que era do interesse de um eventual usuário que fosse possível a personalização do sistema, através do remapeamento dos gestos com as ações. O sistema já permite este tipo de operação, no entanto não existe uma interface para que um usuário final possa realizar estas mudanças. Os arquivos que definem estas funcionalidades são externos, logo um programa externo executado em um outro dispositivo, como por exemplo um *smartphone*, poderia sem maiores dificuldades realizar estas alterações, basta que uma interface de rede seja incluída na central.

As respostas recebidas neste questionário servem de guia para os trabalhos futuros

a serem implementados para que uma versão final do sistema se torne viável. O *feedback* positivo em relação ao protótipo da Hand.io, demonstra o potencial de um sistema deste tipo, que ocuparia uma fatia de mercado que não é explorada pelas grandes empresas do setor de tecnologia. A partir dos resultados obtidos por este experimento, a terceira questão experimental (**QE3**) pode ser respondida de forma positiva quanto à avaliação do protótipo.

6 CONCLUSÕES E TRABALHOS FUTUROS

Durante este trabalho foi apresentada uma nova abordagem de controle de dispositivos eletro-eletrônicos, mais natural e ergonômica que as apresentadas pelas grandes empresas fabricantes de eletrônicos de consumo. Esta abordagem foi detalhada através de um método proposto, que teve a sua efetividade validada através de experimentos em um cenário muito próximo ao que um eventual usuário encontraria em sua casa.

O método em questão desenhou um conjunto de passos que permitiram à este trabalho atingir todos os seus objetivos propostos. Definindo a identificação de métodos de modelagem do fluxo do sistema; a definição de um algoritmo ótimo para a classificação dos movimentos; particularidades físicas, e de protocolos de comunicação; e os componentes necessários para a construção do protótipo.

A utilização de sensores de movimento, como acelerômetros e giroscópios, presos à mão do utilizador, permitiu a geração de um grande volume de dados que foram utilizados em algoritmos de aprendizado de máquina, para a classificação de padrões nos movimentos. Este tipo de algoritmo reduziu consideravelmente a complexidade do desenvolvimento, já que uma abordagem algorítmica que levasse em consideração todas as possibilidades possíveis de um movimento não conta com uma implementação trivial.

A fase experimental do trabalho coletou resultados positivos em relação ao funcionamento do protótipo. Usuários com graus de familiaridade diferentes em relação ao sistema, obtiveram taxas de sucesso e satisfação igualmente altas. Estes resultados comprovam que o método desenvolvido por este trabalho é viável, e indica que existe muito espaço para estudos desenvolvidos na área de dispositivos vestíveis, ainda muito recente e pouco explorada, dentro das áreas de sistemas embarcados e ambientes inteligentes.

Para que um eventual produto final proveniente deste trabalho se torne viável, é necessária a implementação de alguns aprimoramentos, a fim de garantir uma experiência de uso boa o suficiente de modo que o sistema se equipare aos os meios de controle de dispositivos encontrados no mercado. A miniaturização e integração dos componentes à vestimenta dos usuários, e a utilização de conexões sem fio, são pontos essenciais a serem implementados, pois sem estas funcionalidades, nenhum usuário utilizaria um sistema desse tipo sem que fosse algo natural e imperceptível, o grau interesse por parte dos usuários seria reduzido consideravelmente.

Outro ponto a ser melhorado é a classificação dos movimentos, o sistema atualmente classifica apenas um único momento do movimento, o ideal seria que o gesto como um todo

fosse considerado, no entanto com a biblioteca de classificação atual isso não é possível. Seria necessária a migração para uma biblioteca que permitisse que séries temporais fossem classificadas, como é o caso do *TensorFlow*, no entanto, esta biblioteca tem um grau de dificuldade de implementação consideravelmente mais elevado que a usada atualmente, o que inviabilizou a sua utilização durante o desenvolvimento com tempo limitado deste trabalho.

Após a realização do questionário com voluntários que não conheciam o sistema, detalhado na seção experimental, vários outros ajustes de usabilidade podem ser aprimorados como: a realização da captura dos movimentos de maneira contínua, a possibilidade da inserção de novos gestos, um fluxo de execução com mais opções para o usuário, e a criação de uma interface que permita a personalização do sistema.

REFERÊNCIAS

- ABADI, M. et al. Tensorflow: a system for large-scale machine learning. In: **OSDI**. [S.l.: s.n.], 2016. v. 16, p. 265–283. Citado na página 28.
- ARDUINO. **Arduino Uno**. Arduino, 2018. Disponível em: <<https://store.arduino.cc/arduino-uno-rev3>>. Citado na página 21.
- ATILUM. **Philips RC5 Infrared Transmission Protocol**. Atilum, 2018. Disponível em: <<https://techdocs.altium.com/printpdf/212920>>. Citado na página 37.
- ATMEGA328P. Microchip Technology Inc, 2018. Disponível em: <<https://www.microchip.com/wwwproducts/en/atmega328p>>. Citado na página 21.
- ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer Networks**, v. 54, n. 15, p. 2787 – 2805, 2010. ISSN 1389-1286. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1389128610001568>>. Citado na página 25.
- AYALA, K. J. **The 8051 microcontroller: architecture, programming, and applications**. [S.l.]: West Publishing Co., 1991. Citado 3 vezes nas páginas 19, 20 e 21.
- BENDERS, L. P. M.; STEVENS, M. P. J. Petri net modelling in embedded system design. In: **CompEuro 1992 Proceedings Computer Systems and Software Engineering**. [S.l.: s.n.], 1992. p. 612–617. Citado na página 13.
- BERNIERI, G.; FARAMONDI, L.; PASCUCCI, F. A low cost smart glove for visually impaired people mobility. In: **2015 23rd Mediterranean Conference on Control and Automation (MED)**. [S.l.: s.n.], 2015. p. 130–135. Citado na página 12.
- BISHOP, C. M. **Pattern Recognition and Machine Learning**. 1st ed. 2006. corr. 2nd printing. ed. [S.l.]: Springer, 2006. (Information science and statistics). ISBN 9780387310732,0387310738. Citado 2 vezes nas páginas 25 e 27.
- BUTTAZZO, G. **Hard real-time computing systems: predictable scheduling algorithms and applications**. [S.l.]: Springer Science & Business Media, 2011. v. 24. Citado 2 vezes nas páginas 18 e 19.
- CHOUDHARY, T.; KULKARNI, S.; REDDY, P. A braille-based mobile communication and translation glove for deaf-blind people. In: **2015 International Conference on Pervasive Computing (ICPC)**. [S.l.: s.n.], 2015. p. 1–4. Citado na página 12.
- CORTES, C.; VAPNIK, V. Support vector machine. **Machine learning**, v. 20, n. 3, p. 273–297, 1995. Citado na página 27.
- COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE transactions on information theory**, IEEE, v. 13, n. 1, p. 21–27, 1967. Citado na página 27.
- CROW, B. P. et al. Ieee 802.11 wireless local area networks. **IEEE Communications Magazine**, v. 35, n. 9, p. 116–126, Sept 1997. ISSN 0163-6804. Citado na página 36.

CUNHA, E. et al. Formal verification of uml sequence diagrams in the embedded systems context. In: **2011 Brazilian Symposium on Computing System Engineering**. [S.l.: s.n.], 2011. p. 39–45. ISSN 2324-7886. Citado na página 13.

EDWARDS, S. et al. Design of embedded systems: formal models, validation, and synthesis. **Proceedings of the IEEE**, v. 85, n. 3, p. 366–390, Mar 1997. ISSN 0018-9219. Citado 2 vezes nas páginas 18 e 22.

ERTAM, F.; AYDIN, G. Data classification with deep learning using Tensorflow. In: **2017 International Conference on Computer Science and Engineering (UBMK)**. [S.l.: s.n.], 2017. p. 755–758. Citado na página 28.

FILIFELOP. **FILIFELOP Componentes Eletrônicos**. FilipeFlop, 2018. Disponível em: <<https://www.filieflop.com/>>. Citado na página 38.

GIRAULT, C.; VALK, R. **Petri Nets for Systems Engineering**. Springer, 2002. ISBN 3540412174. Disponível em: <<https://www.amazon.com/Petri-Systems-Engineering-Claude-Girault/dp/3540412174%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D3540412174>>. Citado na página 13.

HARITSA, J. R.; LIVNY, M.; CAREY, M. J. Earliest deadline scheduling for real-time database systems. In: [1991] **Proceedings Twelfth Real-Time Systems Symposium**. [S.l.: s.n.], 1991. p. 232–242. Citado na página 19.

HOPCROFT, J. E.; MOTWANI, R.; ULLMAN, J. D. **Introduction to automata theory, languages, and computation**. 2nd ed. ed. [S.l.]: Addison-Wesley, 2001. ISBN 9780201441246,0201441241. Citado 2 vezes nas páginas 24 e 25.

INVENSENSE. **MPU-6050 Six-Axis (Gyro Accelerometer)**. TDK, 2018. Disponível em: <<https://www.invensense.com/products/motion-tracking/6-axis/mpu-6050/>>. Citado 2 vezes nas páginas 33 e 38.

KELA, J. et al. Accelerometer-based gesture control for a design environment. **Personal and Ubiquitous Computing**, v. 10, n. 5, p. 285–299, Aug 2006. ISSN 1617-4917. Disponível em: <<https://doi.org/10.1007/s00779-005-0033-8>>. Citado 3 vezes nas páginas 29, 30 e 37.

KÜHNEL, C. et al. I'm home: Defining and evaluating a gesture set for smart-home control. **International Journal of Human-Computer Studies**, v. 69, n. 11, p. 693 – 704, 2011. ISSN 1071-5819. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1071581911000668>>. Citado 3 vezes nas páginas 30, 32 e 36.

LAMPKA, K.; PERATHONER, S.; THIELE, L. Analytic real-time analysis and timed automata: A hybrid method for analyzing embedded real-time systems. In: **Proceedings of the Seventh ACM International Conference on Embedded Software**. New York, NY, USA: ACM, 2009. (EMSOFT '09), p. 107–116. ISBN 978-1-60558-627-4. Disponível em: <<http://doi.acm.org/10.1145/1629335.1629351>>. Citado na página 13.

LIU, J. et al. uwave: Accelerometer-based personalized gesture recognition and its applications. In: **2009 IEEE International Conference on Pervasive Computing and Communications**. [S.l.: s.n.], 2009. p. 1–9. Citado 4 vezes nas páginas 32, 33, 36 e 37.

- MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: **Proceedings of the fifth Berkeley symposium on mathematical statistics and probability**. [S.l.: s.n.], 1967. v. 1, n. 14, p. 281–297. Citado na página 27.
- MARWEDEL, P. **Embedded system design: embedded systems foundations of cyber-physical systems**. [S.l.]: Springer, 2011. Citado 2 vezes nas páginas 16 e 17.
- MILES, K. H. R. **Learning UML 2.0**. 1. ed. [S.l.]: O'Reilly Media, 2006. ISBN 0596009828,9780596009823. Citado 2 vezes nas páginas 22 e 23.
- MYERS, C. S.; RABINER, L. R. A comparative study of several dynamic time-warping algorithms for connected-word recognition. **The Bell System Technical Journal**, v. 60, n. 7, p. 1389–1409, Sept 1981. ISSN 0005-8580. Citado 2 vezes nas páginas 32 e 33.
- NAVAS, V. X. et al. Smart glove. In: **2012 IEEE Long Island Systems, Applications and Technology Conference (LISAT)**. [S.l.: s.n.], 2012. p. 1–4. Citado na página 12.
- NINTENDO. **Nintendo's Wii Video Game Console**. Nintendo, 2018. Disponível em: <<http://wii.com/>>. Citado na página 32.
- O'FLYNN, B. et al. Novel smart sensor glove for arthritis rehabilitation. In: **2013 IEEE International Conference on Body Sensor Networks**. [S.l.: s.n.], 2013. p. 1–6. ISSN 2376-8886. Citado 2 vezes nas páginas 12 e 13.
- PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011. Citado 3 vezes nas páginas 27, 28 e 37.
- PLASQUI, G. The role of physical activity in rheumatoid arthritis. **Physiology & Behavior**, v. 94, n. 2, p. 270 – 275, 2008. ISSN 0031-9384. *Traces in Metabolism and Nutrition*. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0031938407005057>>. Citado na página 13.
- QUALCOMM. **VIVE Chipsets**. 2017. Disponível em: <<https://www.qualcomm.com/products/vive/chipsets>>. Citado na página 16.
- RABINER, L. R. A tutorial on hidden markov models and selected applications in speech recognition. **Proceedings of the IEEE**, v. 77, n. 2, p. 257–286, Feb 1989. ISSN 0018-9219. Citado 3 vezes nas páginas 29, 32 e 33.
- RAMESH, U. B. K.; SENTILLES, S.; CRNKOVIC, I. Energy management in embedded systems: Towards a taxonomy. In: **2012 First International Workshop on Green and Sustainable Software (GREENS)**. [S.l.: s.n.], 2012. p. 41–44. Citado na página 12.
- RASPBERRY PI FOUNDATION. **BCM2837**. Raspberry Pi Foundation, 2018. Disponível em: <<https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2837/README.md>>. Citado na página 20.
- RASPBERRY PI FOUNDATION. **Raspberry Pi 3 Model B+**. 2018. Disponível em: <<https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>>. Citado 3 vezes nas páginas 20, 39 e 40.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3rd. ed. [S.l.]: Prentice Hall, 2010. (Prentice Hall Series in Artificial Intelligence). ISBN 0136042597,9780136042594. Citado na página 26.

SALVADOR, S.; CHAN, P. Toward accurate dynamic time warping in linear time and space. **Intell. Data Anal.**, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 11, n. 5, p. 561–580, out. 2007. ISSN 1088-467X. Disponível em: <<http://dl.acm.org/citation.cfm?id=1367985.1367993>>. Citado na página 30.

SCIKIT-LEARN. **Ada Boost Classifier**. scikit-learn, 2019. Disponível em: <<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>>. Citado na página 46.

SCIKIT-LEARN. **Cross-validation: evaluating estimator performance**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/cross_validation.html>. Citado na página 46.

SCIKIT-LEARN. **Decision Tree Classifier**. scikit-learn, 2019. Disponível em: <<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>>. Citado na página 46.

SCIKIT-LEARN. **Extra Trees Classifier**. scikit-learn, 2019. Disponível em: <<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html>>. Citado na página 46.

SCIKIT-LEARN. **GaussianNB**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html>. Citado na página 46.

SCIKIT-LEARN. **Gradient Boosting Classifier**. scikit-learn, 2019. Disponível em: <<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>>. Citado na página 46.

SCIKIT-LEARN. **KNeighbors Classifier**. scikit-learn, 2019. Disponível em: <<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>>. Citado na página 46.

SCIKIT-LEARN. **Linear Discriminant Analysis**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html>. Citado na página 46.

SCIKIT-LEARN. **Logistic Regression**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>. Citado na página 46.

SCIKIT-LEARN. **Random Forest Classifier**. scikit-learn, 2019. Disponível em: <<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>>. Citado na página 46.

SCIKIT-LEARN. **SVC**. scikit-learn, 2019. Disponível em: <<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>>. Citado na página 46.

SONY. **SIRC based IR Remote Control Information**. Sony, 2018. Disponível em: <<http://picprojects.org.uk/projects/sirc/>>. Citado na página 37.

TENSORFLOW. **Convolutional Neural Networks**. 2018. Disponível em: <https://www.tensorflow.org/tutorials/deep_cnn>. Citado na página 28.

TENSORFLOW. **Recurrent Neural Networks**. 2018. Disponível em: <<https://www.tensorflow.org/tutorials/recurrent>>. Citado na página 28.

TUULARI, E.; YLISAUKKO-OJA, A. Soapbox: A platform for ubiquitous computing research and applications. In: MATTERN, F.; NAGHSHINEH, M. (Ed.). **Pervasive Computing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 125–138. ISBN 978-3-540-45866-1. Citado na página 29.

VAHID, F.; GIVARGIS, T. D. **Embedded System Design: A Unified Hardware/Software an Introduction**. Wiley, 2001. ISBN 0471386782. Disponível em: <<https://www.amazon.com/Embedded-System-Design-Hardware-Introduction/dp/0471386782%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0471386782>>. Citado na página 16.

WEISER, M. The computer for the 21st century. **SIGMOBILE Mob. Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 3, n. 3, p. 3–11, jul. 1999. ISSN 1559-1662. Disponível em: <<http://doi.acm.org/10.1145/329124.329126>>. Citado na página 13.

WONG, W. E. et al. Recent catastrophic accidents: Investigating how software was responsible. In: **2010 Fourth International Conference on Secure Software Integration and Reliability Improvement**. [S.l.: s.n.], 2010. p. 14–22. Citado na página 13.

YURISH, S. **Advances in Sensors: Reviews, Vol. 3**. Ifsa Publishing, 2016. ISBN 8460877043. Disponível em: <<https://www.amazon.com/Advances-Sensors-Reviews-Vol-3/dp/8460877043>>. Citado na página 12.

ZANELLA, A. et al. Internet of things for smart cities. **IEEE Internet of Things Journal**, v. 1, n. 1, p. 22–32, Feb 2014. ISSN 2327-4662. Citado na página 14.