TRABALHO DE GRADUAÇÃO

Sistema Inteligente para Captura e Interpretação de Dados Temporais da Língua Brasileira de Sinais a partir de uma Luva Instrumentada

Abdullah Zaiter

Lucas dos Santos Schiavini

Brasília, Dezembro de 2020

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

Sistema Inteligente para Captura e Interpretação de Dados Temporais da Língua Brasileira de Sinais a partir de uma Luva Instrumentada

Abdullah Zaiter

Lucas dos Santos Schiavini

Relatório submetido como requisito parcial de obtenção de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Dr. Marcus Vinicius Lamar, CIC/UnB _____
Orientador

Prof. Dr. Jones Yudi Mori Alves da Silva, ENM/UnB _____
Examinador interno

Prof. Dr. Marcelo Grandi Mandelli, CIC/UnB _____
Examinador interno

Brasília, Dezembro de 2020

FICHA CATALOGRÁFICA

ZAITER, ABDULLAH; SCHIAVINI, LUCAS
Sistema Inteligente para Captura e Interpretação de Dados Temporais da Língua Brasileira de Sinais a partir de uma Luva Instrumentada

[Distrito Federal] 2020.

xv, 102p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2020). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

I. Mecatrônica/FT/UnB II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

ZAITER, ABDULLAH; SCHIAVINI, LUCAS. (2020). Sistema Inteligente para Captura e Interpretação de Dados Temporais da Língua Brasileira de Sinais a partir de uma Luva Instrumentada em Engenharia de Controle e Automação, Publicação FT.TG-$n°$05, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 102p.

CESSÃO DE DIREITOS

Abdullah Zaiter

Casa do Estudante Universitário, Bloco B, Apt 208. Universidade de Brasília

70908-022 Brasília – DF – Brasil.

Lucas dos Santos Schiavini

Casa do Estudante Universitário, Bloco B, Apt 223. Universidade de Brasília

70908-022 Brasília – DF – Brasil.

Dedicatórias

Dedico esse trabalho a meus pais que sempre me apoiaram e que me ensinaram o valor de me dedicar em tudo que eu me proponho a fazer.

Lucas dos Santos Schiavini

Dedico este trabalho aos meus pais e meus irmãos, que sempre me apoiaram nos altos e baixos desta vida.

Abdullah Zaiter

Agradecimentos

Agradeço a todos que fizeram parte da minha jornada dentro ou fora da universidade. Em primeiro lugar, aos meus pais, Tayser Zaiter e Safaa Mahaini, meus irmãos Anas, Baraa, Dana, Obaid e Moaz... o que eu aprendi com vocês na vida eu jamais conseguiria aprender em outro lugar ou com outras pessoas.Vocês sempre acreditaram em mim e nas minhas habilidades, apostaram no que acreditei e sempre foram o motivo que me incentivava a me superar dia após dia.
Aos meus sobrinhos Safa, Mariam, Sana, Yousof, Amal e Zaina, Hatem, Nabil, Sara, Omar e Lubna vocês são o motivo que sempre me faz me esforçar para ser um bom exemplo e um motivo de orgulho para vocês.
Agradeço também a DROID, a equipe na qual me tornei um profissional muito mais competente, devo a eles muito do meu conhecimento e o mínimo que posso ter por essa equipe é a gratidão.

Abdullah Zaiter

Agradeço a todos os envolvidos na minha jornada na Universidade de Brasília, primeiro meus pais que estiveram em cada etapa me apoiando e acreditando em mim. Sou grato também ao meu amigo Tiago, que me trouxe para Brasília a tempo de me matricular na UnB.
Além deles agradeço a meus colegas de turma com quem compartilhei muitas horas de trabalho e apoio para melhorar cada dia.
Agradeço ainda aos membros da Equipe DROID com quem aprendi como desenvolver projetos implementando conteúdos de sala de aula. Foi a equipe que mais me deu perspectivas em relação ao que aprendemos no curso e como posso levar esses conteúdos para minha vida profissional. Foi bom trabalhar com vocês.
Por fim, agradeço a você, o leitor.

Lucas dos Santos Schiavini

RESUMO

A dificuldade da comunidade surda é histórica. Incapazes por anos de usar uma linguagem própria, foram excluídos do resto da sociedade e julgados incapazes intelectualmente de exercer funções como cidadãos. Hoje, após anos de conquistas e a recuperação de línguas próprias e reconhecidas, ainda se veem à margem da sociedade ouvinte. Tecnologias de tradução, aplicativos educativos e sistemas vestíveis tem demonstrado o quanto é possível integrar essa comunidade com esforço direcionado e pensado para realidade deles. LIBRAS é a língua de sinais usada no Brasil pela comunidade surda e é reconhecida como uma língua completa por linguistas, além de ter embasamento legal como uma língua oficial do país.

Dessa forma, este trabalho visa discutir sobre a implementação de uma luva capaz de traduzir sinais da Língua Brasileira de Sinais para texto via aplicativo em dispositivo móvel. O intuito é de facilitar o estudo dos sinais, tanto de ouvintes quanto de surdos. Assim, o sistema proposto engloba o estudo de diferentes métodos de treinamento de classificadores para reconhecimento de sinais com redes neurais recorrentes. Também visa analisar a frequência de captura de dados para implementação de protocolos de comunicação entre protótipo e dispositivos móvel. As decisões de sensoriamento e captura de dados foram feitas de forma que o protótipo permitisse medir os parâmetros que compõe um sinal manual, exceto expressão corporal e facial que necessitariam de análise visual.

O trabalho demonstra a dependência entre a acurácia e a frequência de amostragem dos sensores do protótipo físico e obtém acurácias médias para os sinais propostos.

Palavras Chave: Luva Instrumentalizada, Sistema Embarcado, Micro controlador, Redes Neurais Recorrentes, Aprendizado de Máquina, Sistema de tempo real.

ABSTRACT

The deaf community has been a victim of negligence throught history. They've been forbidden of their natural language, and have been cast away from society and deemed intelectually uncapable of being citizens. Now, after years of earning back their rights as members of society and having their natural language recognized, they are still not fully integrated in society. Translation, education and wearable systems have been demonstrating usefulness in integrating this community. LIBRAS is the legally official Sign Language in Brasil. It is valued as a complete language by linguists.

This work has the focus on discussing implementation of a glove that is capable of translating Brazilian Sign Language into text through an application. The goal is to make it easier for listeners and deaf people to study signs. In order to do that, the proposed system analyses different classification training methods for sign recognition. It also studies how the rate of data capture affects the training models. In this way, there is a study on multiple rates to open up communication protocol ideas between data gathering devices and simultaneous translation applications. Sensoring and data capture decisions have been proposed to allow measurements of all parameters that constitute signs except facial and body expressions. Those parameters have been ruled out because of higher costs of application due to the need of visual analysis.

Also this work demonstrates the dependence between the accuracy and the sampling frequency of the sensors of the physical prototype and resultes in mid-levels of accuracy for the proposed signals.

Keywords: Instrumentalized Glove, Embedded Systems, Microcontroller, Neural Networks, Machine Learning, Real time System.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

v	Velocidade linear	[m/s]
a	Aceleração	[m/s^2]
m	Massa	[Kg]
F	Força	[N]

Símbolos Gregos

ω	Velocidade angular	[rad/s]

Grupos Adimensionais

i, k, t	Contador

Subscritos

ref	Referência
fer	Ferramenta
sis	Sistema
des	Desejado

Sobrescritos

$\cdot$	Variação temporal
$-$	Valor médio
$\hat{}$	Valor estimado

Siglas

API	do inglês *Application Programming Interface*
ASL	do inglês, *American Sign Language*
BLE	do inglês, *Bluetooth Low Energy*
DNN	do inglês, *Deep Neural Networks*
I²C	do inglês, *Inter-Integrated Circuit*
IBGE	Instituto Brasileiro de Geografia e Estatística
IDE	do inglês, *Integrated development environment*
IoT	do inglês, *Internet of Things*
IMU	do inglês, *Inertial Measure Unit*
ISM	do inglês, *Industrial, Scientific and Medical Applications*
GATT	do inglês, *General Attribute Table*
GPIO	do inglês, *General Purpose Input/Output*
GND	do inglês, *Ground*
GRU	do inglês, *Gated Recurrent Unit*
LiPo	do inglês, *Lithium Polymer Battery*
LSTM	do inglês, *Long Short-Term Memory*
LIBRAS	Língua Brasileira de Sinais
MEMS	Micro-Eletro-Mecânicos
NVS	do inglês, *Non-Volatile Storage*
RNA	Redes Neurais Artificiais
RNN	do inglês, *Recurrent Neural Networks*
SCL	do inglês, *Serial Clock*
SDA	do inglês, *Serial Data*
SRN	do ingles, *Simple Recurrent Network*
SPI	do inglês, *Serial Peripheral Interface*
UART	do inglês, *Universal Asynchronous Receiver-Transmitter*
UUID	do inglês, *Universally Unique Identifier*

Capítulo 1

Introdução

De forma a tratar o tema de tecnologia assistiva para pessoas da comunidade surda, o seguinte capítulo resume as pesquisas da área que ajudaram a embasar o tema do trabalho, assim como as motivações do trabalho.

Segundo o IBGE (Instituto Brasileiro de Geografia e Estatística), a população brasileira de surdos era de cerca de 10 milhões no ano de 2010 [7], correspondendo a cerca de 5% da população brasileira. Grande parte dessa população tem a Língua Brasileira de Sinais (LIBRAS) como primeiro idioma, geralmente aprendendo o português posteriormente nas escolas para surdos. Dessa forma, soluções de integração da comunidade surda com a população não-surda têm-se visto necessárias.

Tradução de língua de sinais para língua oral utilizando redes neurais tem sido uma das frentes de desenvolvimento de tecnologias assistivas. Essa frente é um problema de interesse acadêmico e social. Os resultados têm sido promissores na área, abordagens de sensoriamento utilizando métodos óticos sensoriais com tecnologia *wearable* tem capturado e estimado com precisão a cinemática da mão humana de forma satisfatória com potencial de ampliação [6]. Linhas de integração e fusão de sensores, junção de visão computacional com medidas inerciais, além de melhorias de hardware ainda se veem necessárias para a utilização futura desse tipo de projeto.

1.1 Definição do problema

O problema a ser resolvido visa de facilitar comunicação entre a comunidade surda e pessoas que não entendem línguas de sinais por meio de um projeto de classificação de sinais em LIBRAS para texto. Para tal problema, existem abordagens de captura de sinais por meio de sensores inerciais juntos ao corpo, e de visão computacional. No entanto, dado o alto custo de captura e processamento de dados da abordagem de visão computacional e do desenvolvimento de sensores inerciais com baixo custo, foi escolhida a abordagem sensorial.

Escolheu-se a criação de um hardware de fácil uso que permita surdos e não surdos aprenderem

a se comunicar facilmente via uma luva instrumentalizada e um aplicativo de celular. O projeto consiste num estudo de capacidade de mapear sinais com movimentos em texto, treinar utilizando um protótipo que indique o acerto dos sinais realizados, retornando para o usuário qual sinal foi reconhecido.

Outro problema a ser resolvido é a falta de grandes *data sets* de pessoas usando língua de sinais de forma a treinar algoritmos de *machine-learning* para projetos em hardware específicos. Dessa forma o trabalho se divide em 5 frentes: Criação do projeto físico; Integração com um aplicativo em dispositivo móvel; Criação de um banco de dados; Módulo de subamostragem de dados para simular diferentes taxas de amostragem de sensores; E por fim, treinamento de rede neural .

1.2 Objetivos do projeto

O propósito do projeto é expandir o conhecimento de integração sensorial para medidas cinemáticas de juntas humanas na forma de um protótipo de luva capaz de ler sinais em LIBRAS e transmiti-los para um aplicativo móvel externo. Além disso, o sistema visa ser agnóstico à frequência de amostragem entre o protótipo da luva e o aplicativo móvel.

A luva age na forma de coleta e construção de um banco de dados que pode ser expandido futuramente. Já a aplicação externa tem foco de obter dados e ser capaz de identificar o sinal correspondente. Dessa forma, com o propósito de realizar o projeto, foram definidos os seguintes objetivos específicos.

- **Projeto e implementação do Hardware da Luva**:

  Para desenvolver o projeto, seguimos os preceitos:

  - Poucas partes móveis que possam colidir entre si quando utilizado pelo usuário;
  - Possuir material agradável à pele humana, de forma a facilitar o uso prolongado;
  - Hardware de fácil uso, com dificuldade semelhante a de vestir uma luva comum.

  Por fim, o protótipo com placa de desenvolvimento deve permanecer no pulso com sensores em cada dedo da mão. Deve ainda, satisfazer taxas de amostragem de dados suficiente para capturar sinais realizados rapidamente por usuários experientes.

- **Desenvolvimento do Aplicativo**:

  Utilização de um aplicativo de captura e processamento de dados. O aplicativo irá ser desenvolvido utilizando *frameworks* que facilitem a portabilidade para outras plataformas. Assim, mesmo um aplicativo Android pode ser adaptado em iOS futuramente de forma a aumentar alcance de utilização.

- **Criação do Banco de Sinais**:

  O Banco de Dados de Sinais (*Data Set*) deve ser grande o suficiente de forma a permitir resultados significativos no treinamento de redes neurais.

- **Divisor de Frequência**:

  Os dados devem ser independentes da frequência de transmissão de dados, com a única limitação sendo a frequência de captura de sensores pelo microcontrolador. Dessa forma é possível realizar uma análise de desempenho de classificadores por frequência. Com isso é realizável uma comparação de diferentes taxas de amostragem de dados.

- **Projeto de Rede Neural**:

  O projeto final da rede deve tornar possível a predição de sinais comuns em LIBRAS selecionados para uma única mão.

  Além disso, a rede treinada deve possuir o menor tamanho possível de forma a otimizar o processamento dos dados no aplicativo, devido às limitações do dispositivo móvel utilizado.

1.3 Estrutura da Monografia

A forma como o texto foi organizado e estruturado segue:

- Capítulo 2: Fundamentos - Seguirá com a fundamentação teórica necessária para entendimento dos conceitos utilizados neste trabalho.
- Capítulo 3: Metodologia - Detalha as decisões de projeto e como foi pensada a implementação do protótipo.
- Capítulo 4: Resultados - Apresenta os testes realizados durante o desenvolvimento do sistema e discussão dos resultados finais.
- Capítulo 5: Conclusões - Conclui a realização do projeto e realiza um sumário do trabalho produzido. Por fim, apresenta uma breve discussão com propostas para trabalhos futuros.

Capítulo 2

Fundamentos

Este capítulo apresenta o conhecimento agregado necessário à compreensão deste trabalho.

O capítulo que segue tem como objetivo detalhar conhecimentos base para o desenvolvimento do projeto de uma luva de captura de cinemática de uma mão humana para o contexto de frases comuns em LIBRAS e classificação de sinais a partir destes dados.

De início será tratada sobre a história da comunidade surda, discorrendo sobre a educação e qualidade de vida dos surdos e a dificuldade de comunicação com a comunidade ouvinte. Em seguida há a introdução das tecnologias utilizadas.

Partindo do contexto histórico, foram definidos parâmetros que caracterizam a LIBRAS e o espaço escolhido de análise. Foi dado destaque quanto às formas de medição, tipos de sensores, microcontroladores utilizados, protocolos de transferência de dados, arquitetura de software, *frameworks* utilizados e o meio de treinamento do sistema utilizando redes neurais. Por fim, é realizado uma revisão bibliográfica de trabalhos constituídos como estado da arte.

2.1 Contexto Histórico

Para entender a motivação de resolver um problema de falta de comunicação entre a comunidade surda e a ouvinte, é necessário estudar o contexto do problema.

A língua de sinais tem sido observada como uma manifestação natural da comunicação de pessoas nascidas surdas. No entanto, ainda há uma preferência por disciplinas auditivas em escolas criadas para surdos [8].

A tradição oralista vigorou por muitos anos, forçando os alunos surdos a desenvolverem habilidades vocais por meio de treinamento intensivo, de forma a integrá-los na sociedade. No entanto, tem sido levantado o quanto isso pode atrasar o desenvolvimento psicológico do ser humano, ao ter que aprender uma linguagem que não é de toda sua.

Ao longo de toda a história registrada, surdos não tiveram participação na sociedade de forma integrada. O início de uma educação formal começou na Idade Moderna, quando membros de

famílias reais que nasciam surdos precisaram ser treinados pelos familiares aristocratas para se tornarem herdeiros.

Um abade chamado *De l'Epée*, que vivia em Paris, reconheceu a comunidade surda de classe baixa e sua língua de sinais nativa que utilizavam para se comunicar [8]. Com interesse nessa língua, que esperava ser universal, decidiu aceitar pupilos surdos para que pudesse entender mais de sua realidade. Passou a ensinar escrita e dar acesso aos conhecimentos e culturas comuns à época para esses pupilos. Seus pupilos impactaram a sociedade ao demonstrar a capacidade intelectual de pessoas da comunidade surda para o resto da população na França. A escola de *De l'Epée* foi o primeiro projeto do tipo a obter auxílio público.

Seus pupilos e professores formados em 1789 expandiram o ensino de surdos na França e Europa para 21 escolas. Em 1791 a escola inicial de *De l'Epée* se transformou no *National Institution for Deaf-Mutes* em Paris.

O impulso para alfabetização e ensino de surdos foi expandido para os Estados Unidos em 1816 com *Laurent Clerc*, um dos professores formados na escola de *De l'Epée*, e Thomas Gallaudet [8]. Em 1817 foi fundado o Asilo Americano para Surdos(do inglês *American Asylum for the Deaf*) em Hartford.

De forma semelhante ao que ocorrera na França, professores, filósofos e o público em geral ficaram impressionados pela cultura e intelecto dos surdos. O ensino de uma língua de sinais foi difundido entre diversas escolas em cidades cuja demografia significativa de surdos justificava o esforço no alcance. O Asilo de Hartford se tornou centro e berço da criação da ASL(Do inglês, *American Sign Language*).

A onda de progresso do ensino de surdos chegou a um hiato em 1869 com a morte de *Clerc*. Foi no Congresso Internacional de Educadores de Surdos de 1880, em Milão, com exclusão da votação de professores surdos, que a votação para ensino de oralismo se tornou a metodologia padrão de ensino [8] Essa decisão trouxe uma contracorrente de dois séculos na qual crianças surdas seriam forçadas à oralizar, línguas de sinais seriam ativamente combatidas e substituídas por um esforço de integrar o surdo na sociedade às custas de uma língua própria.

Os efeitos vistos então, foram drásticos para medidas de educação geral de alunos surdos. Demorava cerca de 5 a 8 anos de ensino individual extensivo para o surdo adquirir a oralização.Esse ensino reduzia o tempo que poderia ser destinado ao aprendizado de cultura, habilidades complexas e aprendizado incidental, que ocorre fora das instituições de ensino/ambiente acadêmico.

Foi somente em 1970 que psicólogos e historiadores retomaram o questionamento acerca de tamanha diferença entre os níveis de educação de crianças surdas e ouvintes. Foi proposto, então, a reintrodução do ensino de línguas de sinais, com a opção de aprendizado de oralismo caso seja do desejo do surdo.

Mesmo com todos os avanços desde então, muito do que é mostrado na mídia como língua de sinais, acaba sendo uma amálgama entre a língua de sinais e a transliteração da língua oralizada. O chamado "inglês de sinais" ou "português de sinais" ao invés da ASL e LIBRAS, respectivamente.

As verdadeiras línguas de sinais são completas da mesma forma que as línguas orais. A

diferença é fundamentalmente o meio que é expressa. LIBRAS e outras línguas de sinais utilizam mecanismo de transmissão no meio visual-espacial, enquanto as orais as fazem por meio oral-auditivo.

É importante citar ainda, que surdos-cegos utilizam apenas do tato para se comunicar, logo, a língua se mantém a mesma, mas o meio passa a ser somente espacial.

2.2 LIBRAS

Existem cerca de 300 línguas de sinais no mundo hoje [9] . As línguas de sinais, assim como línguas faladas, diferem entre si. Um sistema de tradução automática deve ser construído para um tipo específico de língua de sinais.

LIBRAS é a Língua Brasileira de Sinais, que é utilizada pela comunidade surda do Brasil. Possui gramática, sintaxe e regras bem estabelecidas. Com isso permite que seus praticantes possam expor e construir ideias, pensamentos e sensações.

É dito por vezes que é uma língua com características cinemáticas, já que podem expressar ideias no campo tridimensional do espaço junto com o tempo, do ponto de vista de quem vê pode ser análogo à absorver informações em uma peça de teatro ou filme [8]. Assim, tem um conjunto de elementos e modificadores que possibilitam a comunicação, flexão de verbos e regionalismos.

Um sinal consiste em formato da mão, localização da mão no espaço, movimento do braço e da mão e expressão facial. Nesse projeto, expressão facial não será considerada. É importante destacar a diferença entre LIBRAS e Datilologia. Datilologia é o alfabeto manual, utilizado para representar letras oralizáveis por meio sinais de mãos [10]. Visto isso, o foco deste projeto é em 10 sinais comuns de conversação em LIBRAS apenas, sem o alfabeto manual.

O consenso entre a bibliografia de LIBRAS é de que para confecção de um sinal, existem 5 parâmetros necessários [1].

1. Configuração das Mãos (CM): São os formatos que as mãos direita e esquerda assumem no momento da execução do sinal.

Figura 2.1: Configuração de mãos. À esquerda o sinal de 'educado(a)', e à direita o sinal de 'hábito' [1].

   Os sinais, apresentados na Figura 2.1, diferem apenas na configuração das mãos. Da mesma forma que existem um número limitado de fonemas em uma língua oral, também existe em cada língua de sinais um conjunto finito de configurações de mão. É importante citar que a configuração de mãos que compõe esse conjunto não é universal entre línguas de sinais e nem coincide com o alfabeto manual.

2. Ponto de Articulação (PA): Definido ao longo da literatura como a área no corpo, ou no espaço de articulação definido pelo corpo, em que ou perto da qual o sinal é articulado.

Figura 2.2: Ponto de articulação. À esquerda o sinal de 'aprender', e à direita o sinal de 'sábado' [1].

A Figura 2.2 apresenta os sinais de 'aprender' e 'sábado' que possuem CM similares, porém executados em PA diferentes do corpo.

3. Movimento (M): Corresponde ao deslocamento das mãos durante a realização do sinal.

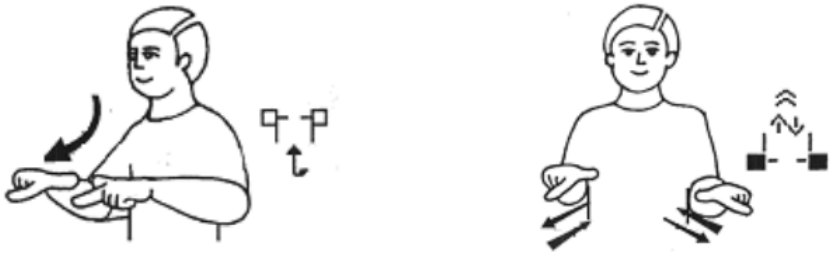


Figura 2.3: Movimento. À esquerda o sinal de 'vídeo', e à direita o sinal de 'trabalhar' [1].

Na Figura 2.3, é possível perceber que ambos sinais possuem a mesma configuração de mãos e no mesmo ponto de articulação. No primeiro sinal, o movimento é realizado para frente, enquanto que no outro é feito com as duas mãos alternadas para frente e para trás.

4. Orientação ou Direcionalidade (O/D): Esse parâmetro corresponde à direção e orientação da palma da mão. Alguns sinais apenas se diferenciam pela orientação da mão, com mesma configuração, movimento e ponto de articulação.

   No exemplo mostrado na Figura 2.4, os verbos IR e VIR, SUBIR e DESCER possuem significados diferentes apenas pela direcionalidade [11].

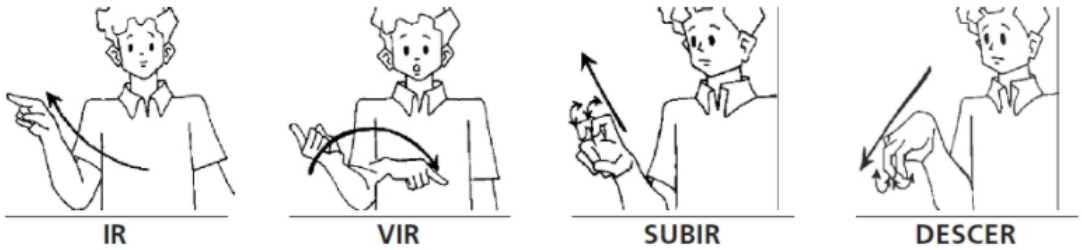


Figura 2.4: Orientação [1].

É possível observar a mudança de orientação das mãos para indicar sentidos diferentes.

5. Expressão facial e/ou Corporal (EF/C): Alguns atributos da língua são enfatizados e comunicados por meio de expressões faciais, como sinalizar emoções e responder “sim” ou “não” balançando a cabeça.

Figura 2.5: Expressão não corporal. À esquerda o sinal de ’triste’, e à direita o sinal de ’exemplo’ [1].

Pela Figura 2.5 é possível observar que a diferença principal entre os sinais é a expressão facial do indivíduo realizando-a.

Dessa forma, segue que LIBRAS possui em sua constituição de gramática e sintaxe com todos os parâmetros acima descritos Estes cinco parâmetros contêm significados, e portanto, sendo morfemas [11].

2.3 Microcontroladores

De maneira à realizar um sistema embarcado que seja capaz de capturar dados de movimentos e posições de uma mão humana, é necessário um dispositivo capaz de controle de sensores, processamento e armazenamento de dados. Além disso, é desejável que este dispositivo possua a possibilidade de utilizar um protocolo de comunicação capaz de transmitir dados para uma aplicação externa.

Microcontroladores são pequenos circuitos integrados programáveis que possuem unidade de processamento, sistema com clock próprio, memória, alimentação, interface de comunicação com outros dispositivos, como pinos GPIO (em inglês: *General Purpose Input/Output*).

São utilizados em aplicações que requerem controle e interação com sensores e outros circuitos menores, servindo como ponte de comunicação. No entanto, não costumam possuir poder de processamento elevado, implicando em pouco consumo de energia.

O interesse em usar microcontroladores em aplicações é o de embarcar o sistema, criando um sistema específico que não muda de programação durante o uso. Ao longo dos anos foram surgindo novas placas de desenvolvimento que entram na especificação de microcontroladores e microprocessadores.

2.3.1 Arduino

A plataforma mais comum no mercado, por sua facilidade de uso e aprendizado, é o Arduino, mostrado na Figura 2.6.

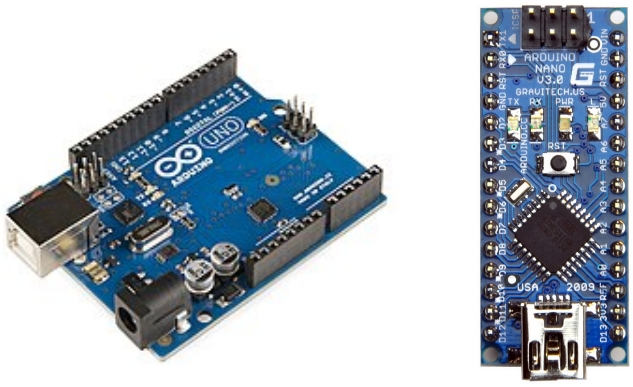


Figura 2.6: À esquerda um Arduino Uno e à direita um Arduino Nano

É uma plataforma *open source*, com ambiente de desenvolvimento e linguagem próprios. Permite a utilização por chamados *makers*, estudantes e pesquisadores, funcionando como uma alternativa de prototipagem rápida a outros sistemas como PIC [12].

Existem diversas placas que utilizam a plataforma Arduino: Arduino Uno, Arduino Mega, Arduino Nano, etc. Existem ainda, outras placas que possuem plataformas próprias, mas que são compatíveis com o ambiente de desenvolvimento Arduino, como a ESP32.

2.3.2 ESP32

Para projetos que necessitam de um controle de plataforma maior por parte dos desenvolvedores, existem placas com mais recursos disponíveis e uma delas é o ESP32.

ESP32 é um sistema de baixa potência em um chip, com módulos Wifi e Bluetooth integrados. Foi criado pelo Espressif Systems e pode possuir os chips ESP32-D0WDQ6, ESP32-D0WD, ESP32-D2WD ou ESP32-S0WD [13].

É possível observar a arquitetura de blocos na Figura 2.7, com a memória embarcada em flash se comunicando com os barramentos em hardware como: SPI, I²C, I²S, SDIO, UART, etc. Além disso é possível analisar os protocolos de comunicação wireless presentes no chip ESP32 como Wifi, Bluetooth e Rádio.

Os chips da família ESP32 possuem como atributo central o microprocessador *Tensilica Xtensa LX6*, com frequência de clock de até 240 MHz. Possui switchs de antena, filtro de passa baixa para inputs, amplificador de potência e módulos de gerenciamento de potência. É possível visualizar na Figura 2.8 uma outra placa desenvolvida com o chip ESP32 chamada de *LOLIN D32*, com módulo de carregamento de bateria *LiPo* integrado.

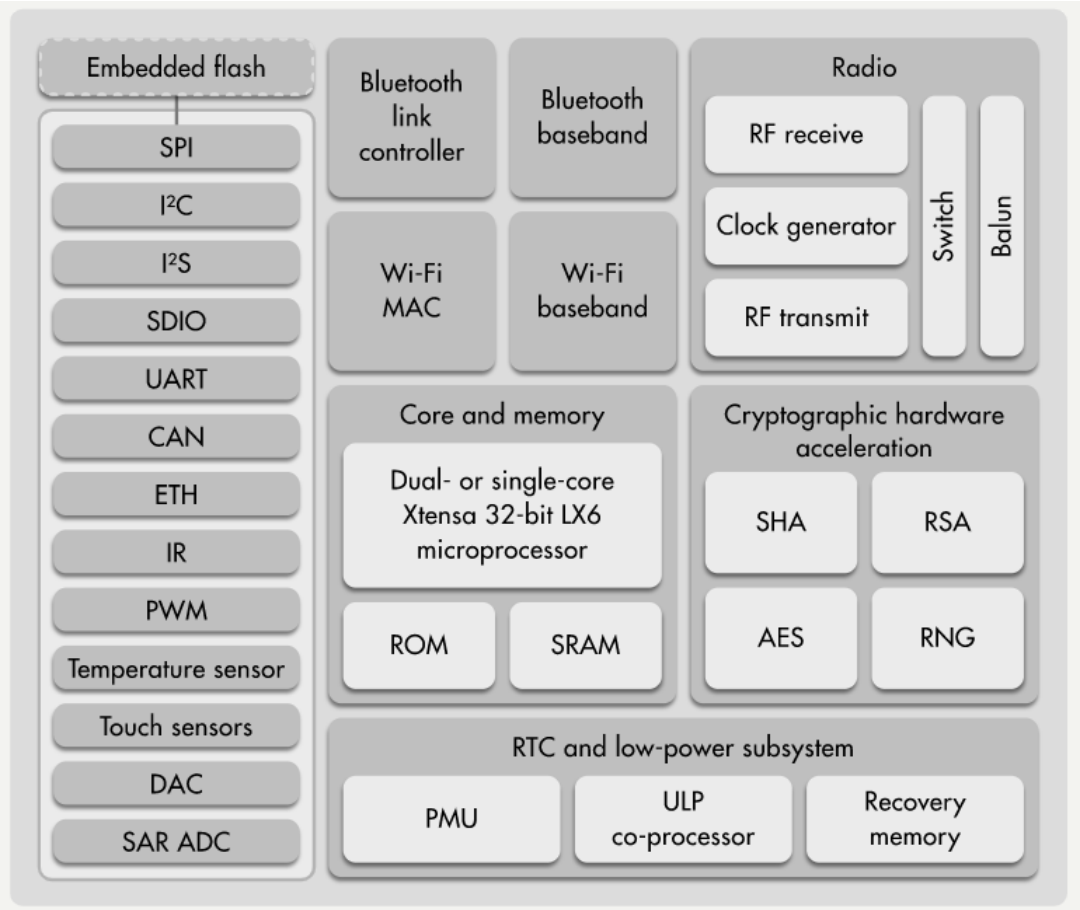


Figura 2.7: Diagrama de blocos da arquitetura da plataforma ESP32

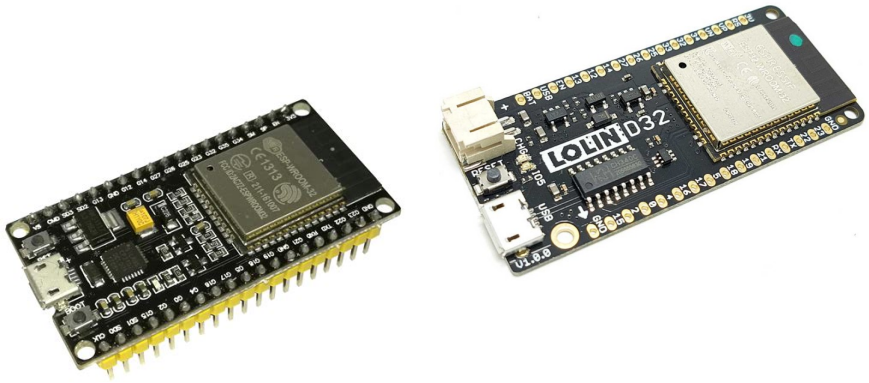


Figura 2.8: Placas Esp32 e LolinD32 que utilizam o mesmo chip ESP32-WROOM-32

É utilizado para projetos embarcados, *wearables* e IoT. Em mais detalhes, seguem alguns atributos do ESP32:

- Memória

  - 448 KiB de ROM
  - 520 KiB de RAM
  - Memória Flash Embarcada: 4 MiB
- Processador com 2 Cores
- Frequência de clock de até 240 MHz
- GPIO: Interfaceamento com I²C, UART, SPI, I²S, PWM, DMA com toque capacitivo, conversor analógico-digital e digital-analógico.
- Conexão Wireless:
  - Wifi: 802.11 b/g/n/e/i (802.11n @ 2.4 GHz até 150 Mbit/s)
  - Bluetooth: v4.2 BR/EDR e Bluetooth Low Energy (BLE)

2.4 Sensoriamento

É possível assumir duas abordagens de sensoriamento de forma a capturar dados cinemáticos de juntas humanas. Uma das abordagens é visual, com utilização de câmeras e análises óticas em laboratórios [14], enquanto a outra é com a utilização de sensores inerciais.

Nos anos mais recentes, sensores capazes de operar com baixo consumo de energia com o avanço de sistemas Micro-Eletro-Mecânicos(MEMS) permitiram soluções computacionais *wearable* [15].

Sensores *wearable* (vestível) consistidos de acelerômetros, giroscópios e sensores magnéticos estão com grande disponibilidade no mercado [16]. Esses sensores têm sido amplamente fabricados ao longo dos anos e têm demonstrado ser uma alternativa de baixo custo e consumo de energia, além de serem altamente transportáveis [14].

Sensores flexores e Unidades de Medição Inercial(IMU do inglês: *Inercial Measurement Unit*) são as duas alternativas mais comuns para rastrear movimentos de mãos e dedos [6].

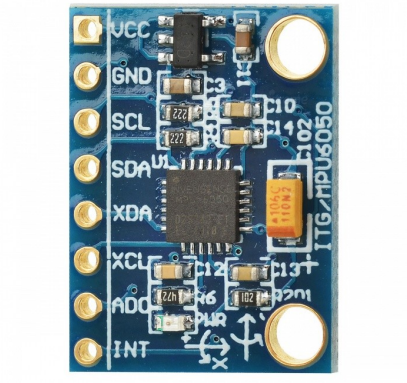


Figura 2.9: IMU - Sensor GY-521

A Figura 2.9 apresenta uma IMU comercial. Nesta IMU são presentes acelerômetros de 3 eixos, giroscópios de 3 eixos, um magnetômetro e um termômetro. Cada sensor IMU retorna aceleração em (m/s^2) e velocidade angular em (graus/segundo).

Assim, esses sensores *wearables* tem sido utilizados na medição de cinemática de juntas de forma a expandir as análises de movimentação humana para fora de um laboratório.

As vantagens de utilização de sensores inerciais sobre sistemas óticos de análise de movimento se tornam mais evidentes em aplicações como: sistemas ambulatoriais para medição de movimento em tempo real, retorno rápido para performance em esportes competitivos e agir como alarme protetor para quando o usuário estiver em risco de acidente [17].

2.4.1 Acelerômetro

Acelerômetro é um dispositivo responsável por medir a aceleração de um objeto em que está acoplado. É comumente utilizado em conjunto com giroscópios e bússolas em circuitos integrados IMU.

Acelerômetros são regidos pela segunda *Lei de Newton* em conjunto com a *Lei de Hooke*, segundo as equações

$$F = ma \tag{2.1}$$

e

$$F = kx \tag{2.2}$$

onde F é a força aplicada, m a massa, a aceleração, x o deslocamento da mola e k a constante da mola.

Dessa forma, é possível correlacionar a medida de aceleração relativo à massa conhecida do acelerômetro com o deslocamento das molas que compõe o sistema, de acordo com

$$a = \frac{kx}{m}. \tag{2.3}$$

O acelerômetro pode ser compreendido como uma massa amortecida em uma mola. Quando a massa se move, a mola pode acelerar a massa na mesma velocidade e em sentido oposto. O sistema deve ser amortecido de forma a evitar que oscilações do sistema massa-mola prejudiquem medições. Cada acelerômetro possui comportamento adaptado para frequências de acelerações compatíveis com a frequência natural do sistema. Dessa forma, possui resposta em frequência diferente para cada tipo de entrada de aceleração no sistema.

É possível criar acelerômetros utilizando dispositivos mecânicos como alavancas e realizar medições por meio da utilização de componentes piezoelétricos. Utilizando o efeito piezoelétrico, as

propriedades elétricas dos materiais permitem valer-se de uma relação de corrente para deslocamento. Com a massa de prova conhecida, estima-se a aceleração.

No entanto, a maior parte de acelerômetros no mercado são do tipo MEMS (do inglês: *Micro-Electro-Mechanical Systems*) que são fabricados com uma massa conhecida amortecida. O amortecimento pode ser devido a gases internos ao dispositivo ou a capacitores do sistema.

A fabricação mais comum de acelerômetros MEMS são capacitivas, devido à resolução e sensitividade em mV/g maior que acelerômetros realizados com efeito piezoelétrico para medidas em baixa frequência [18].

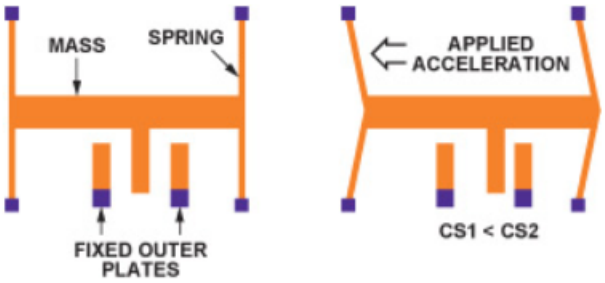


Figura 2.10: Estrutura interna de um Acelerômetro MEMS Capacitivo [2].

É possível observar na Figura 2.10 que o princípio massa-mola se mantém. Um acelerômetro MEMS possui dois atuadores, um fixo em um plano em um substrato, enquanto o outro é montado em molas que podem se mover a partir de uma aceleração. Essa aceleração muda a capacitância entre o atuadores físico e móvel.

2.4.2 Giroscópio

Giroscópio é um dispositivo que permite medir a orientação e velocidade angular de um objeto. Cada tipo de giroscópio tem um comportamento diferente de acordo com o tipo. O giroscópio tradicional possui um rotor, que gira em direção a um eixo de rotação. Devido à conservação de momento angular, o giroscópio tende à se manter ao eixo de rotação inicial. Dessa forma, qualquer força que tente mudar esse eixo pode então ser medida.

De forma similar ao acelerômetro, giroscópios também são encontrados do tipo MEMS. É possível ver na Figura 2.11 que um giroscópio MEMS possui molas, sensores de medição de força *Coriolis*, massa e um sistema inercial.

O princípio de funcionamento do giroscópio MEMS utiliza o efeito *Coriolis*, visto na Figura 2.12. O sensor se utiliza de uma massa ressonante de forma a simular o movimento de aproximação e afastamento do eixo. A massa ressonante é fabricada de forma a ressoar em apenas uma direção.

Dessa forma como vemos na Figura 2.12, quando a massa se desloca para fora do círculo de rotação em azul, ela é acelerada para a direita, gerando uma reação no sistema para a esquerda.

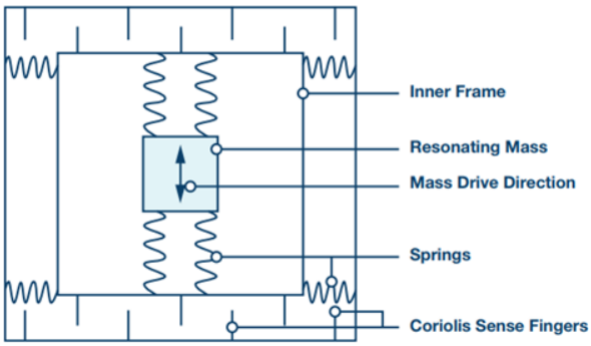


Figura 2.11: Estrutura interna de um Giroscópio MEMS [3].

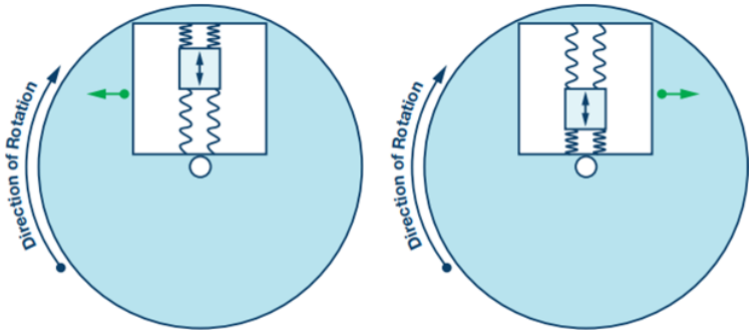


Figura 2.12: Efeito Coriolis demonstrado [3].

O oposto ocorre quando a massa se desloca em direção ao centro, acelerando o sistema para a esquerda e gerando uma reação do sistema para a direita.

Na Figura 2.11, vemos que são acoplados sistemas de mola perpendiculares ao eixo de deslocamento da massa, e os sensores Coriolis capacitivos percebem o deslocamento dessas molas perpendiculares, estimando a força gerada no sistema [3].

Assim, com a taxa de rotação aumentando, a massa vai ser deslocada pelo sistema, gerando uma medição da pseudo-força de Coriolis correspondente à mudança capacitiva dos sensores.

2.5 Barramento I²C

Existem diversos protocolos de comunicação via hardware que permitem a utilização de vários dispositivos com economia de pinos GPIO, que é o desejado para este projeto. O protocolo escolhido foi o barramento I²C (do inglês, *Inter-Integrated Circuit*).

I²C combina as melhores partes dos protocolos SPI (do inglês, *Serial Peripheral Interface*) e UART (do inglês *Universal Asynchronous Receiver-Transmitter*) , possibilitando a comunicação de vários dispositivos escravos a um único dispositivo mestre, como no SPI. Além disso é possível colocar múltiplos mestres controlando um ou vários escravos.

Assim como o protocolo UART, o I²C utiliza utiliza apenas dois fios para transmitir dados entre dispositivos, conforme apresentado na Figura 2.13.

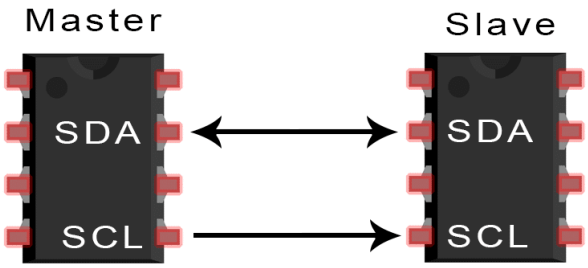


Figura 2.13: Transferência de dados no I²C [4]

O sinal SDA (*Serial Data*) consiste na linha que o mestre e o escravo possuem para receber ou enviar dados. Já o sinal SCL (*Serial Clock*) consiste na linha que carrega o sinal de clock de forma a sincronizar os dois dispositivos

I²C é um protocolo de comunicação serial, assim cada dado é transmitido bit a bit por meio da linha SDA. O barramento I²C é síncrono, assim como no SPI, logo, a saída de bits é sincronizada por meio do sinal de clock compartilhado pela linha SCL. O sinal de clock é sempre controlado pelo mestre.

2.6 Protocolo Bluetooth

Uma transmissão sem fio é desejada para transmissão de dados capturados no protótipo embarcado para uma aplicação externa. Com isso, é possível reduzir cabeamento para transmissão do dado obtido pelo microcontrolador para uma aplicação externa que extrai características destes dados. Uma forma comum de transmissão de dados sem fio para projetos embarcados é o *Bluetooth*.

Em 1996, Intel, Ericsson e Nokia planejaram criar um padrão para tecnologia de rádio de curto alcance para suportar colaboração entre diferentes produtos e indústrias. Durante a reunião foi utilizado um nome temporário remetendo ao rei Harald Bluetooth, famoso por unir a Escandinávia da mesma maneira que as três empresas líderes pretendiam unificar as indústrias de celular e de computador com links de curto alcance sem fio [19].

Em 2020 existem dois tipos de tecnologia Bluetooth, Bluetooth Low Energy e Bluetooth Clássico [20]. Ambos operam na mesma banda de frequência de 2400 à 2483,5 MHz dentro da banda ISM (do inglês *Industrial, Scientific and Medical (ISM) applications (of radio frequency energy)*) 2,4 GHz com canais diferentes.

No entanto, o Bluetooth LE, também conhecido como BLE, é mais popular pelo menor consumo de energia e de custo de operação, além de permitir a comunicação ponto a ponto do Bluetooth clássico.

2.6.1 Bluetooth Low Energy

Bluetooth Low Energy como protocolo de comunicação surgiu como resultado de pesquisa de funcionários da Nokia [21]. A principal necessidade que não era suprida por outros protocolos da época era o baixo consumo de energia e custo. Se baseando na tecnologia Bluetooth, surgiu em 2004 o novo protocolo chamado Bluetooth Low End Extension. No início a tecnologia foi chamada de Wibree e renomeado posteriormente para Bluetooth Low Energy [22].

Integrado como parte da especificação Bluetooth 4.0, o protocolo BLE tem sido preferido para aplicações IoT(do inglês: *Internet of Things*) e implementa a arquitetura de Tabela de Atributos Genéricos(GATT, do inglês *Generic Attribute Table*).

2.6.2 GATT

A partir da conexão de dois dispositivos, é necessário um protocolo de troca e acesso de dados disponibilizados. Bluetooth Low Energy utiliza o GATT(do inglês *Generic Attribute Profile*) para realização desta tarefa. O GATT define o formato dos dados expostos por um dispositivo BLE, assim como os procedimentos de acesso desses dados e dois agentes: Servidor e Cliente. A seguir, a terminologia utilizada para descrever o protocolo de acordo com conceitos de GATT [23].

- Cliente: O agente que inicia comandos e realiza requisições GATT. O protocolo pode ser implementado por um computador, celular ou um microcontrolador;
- Servidor: O agente que recebe comandos GATT, requisições e retorna respostas. É importante ressaltar que um dispositivo BLE pode ser Servidor e Cliente ao mesmo tempo, por poder acessar dados de outro dispositivo como Cliente, enquanto expõe seus próprios dados como Servidor;
- Característica: Dado a ser exposto pelo Servidor e transmitido para o Cliente. Por exemplo, a leitura de sensores;
- Serviço: A coleção de características de um mesmo tipo. É utilizado como forma de organizar informações de mesma funcionalidade;
- Perfil de Aplicação: Define o comportamento do Cliente e do Servidor, como serão seus serviços expostos e dados a serem transmitidos;
- Descritor: Informação adicional da característica. É um valor opcional, e cada característica pode ter um ou mais descritores;
- Identificadores: Todas as características, serviços e descritores são reconhecidos pela arquitetura GATT como um atributo. De forma a organizar e relacionar características com seus respectivos serviços e descritores, são utilizados valores de 16 bits ou 32 bits para representar um valor único de identificação. Esses valores são Identificadores Únicos Universais (UUID, do inglês, *Universally Unique Identifier*) dos atributos.

- MTU: *Maximum Transmission Unit*, representa o tamanho máximo do pacote de dados a ser enviado em uma única transmissão. É ideal ter um valor de MTU alto para evitar consumo desnecessário de energia na transmissão em sistemas cujo processamento por pacote é custoso. Um MTU maior traz eficiência ao sistema por permitir que cada pacote de rede carregue mais dados quando o protocolo de comunicação gera *overhead* grande no gerenciamento de pacotes [24] como: *Headers* ou *delays* por pacote;

Existem 6 tipos de operações em características:

- Requisições: Enviados do Cliente para o Servidor com espera de uma resposta, como Escrita de dados ou Leitura de dados;
- Respostas: Dados enviados do Servidor após receber uma requisição. Pode enviar dados específicos ou mensagens pré-definidas de sucesso ou falha de escrita;
- Comandos: Enviados do Cliente para o Servidor e não requer uma resposta;
- Notificações: São mensagens enviadas do Servidor para o Cliente de forma a indicar que alguma característica mudou. O Cliente deve permitir o recebimento de notificações para a característica que pode mudar. O Cliente não retorna recebimento ou não de notificação. O Cliente pode permitir Notificações ou Indicações pelo Descritor e Configurador de Características do Cliente (CCCD - *Client Characteristic Configurator Descriptor*). A opção 1 habilita as Notificações; 2, as Indicações; e 0, desabilita ambos;
- Indicação: São enviadas do Servidor para o Cliente, de forma semelhante à notificações, no entanto, é esperado um ACK de retorno do Cliente;
- Confirmação: Enviados do Cliente para o Servidor após recebimento da Indicação;

2.7 Arquitetura de Software

Para a realização de um projeto em equipe, é necessário o estabelecimento de diversos contratos de como o projeto será desenvolvido. Há a modelagem do projeto e dos requisitos e a forma como será desenvolvido. Uma das escolhas ao longo do projeto foi de como desenvolver software de forma que ambos os alunos pudessem compartilhar códigos sem dificuldade. Logo, foi seguido princípios de Orientação a Objeto e boas práticas do *SOLID* como serão descritos a seguir.

A metodologia de *Orientação a Objeto* permitiu o desenvolvimento de softwares com agilidade e com direcionamento de responsabilidades menor para cada desenvolvedor individual. No entanto, viu-se necessário o desenvolvimento de práticas para facilitar a utilização e extensão de softwares utilizando essa metodologia.

Em projetos que seguem orientação a objeto, existem 4 parâmetros principais que indicam que o sistema está deteriorando, o que o tornará obsoleto por não ser extensível [25]. Os parâmetros são: Rigidez, Fragilidade, Imobilidade e Viscosidade.

- Rigidez:

  Rigidez é a dificuldade de modificar um código mesmo que a modificação seja simples. Cada modificação gera uma cascata de modificações necessárias em módulos dependentes. O tempo para realizar tarefas simples se torna mais difícil de medir, o que pode levar setores administrativos a não permitir que engenheiros ou programadores modifiquem mais o projeto, instalando uma rigidez oficial. O que começou como uma falha de projeto, se torna uma política de administração falha;

- Fragilidade:

  Similar à rigidez, fragilidade é caracterizada pela tendência de quebra no funcionamento do software em diversas partes toda vez que é modificado. Normalmente as quebras ocorrem em lugares sem relação conceitual com a área que foi modificada. Com o tempo, softwares que tendem a quebrar a cada tentativa de correção se tornam obsoletos e perdem credibilidade;

- Imobilidade:

  Imobilidade é a incapacidade de reutilizar software de outros projetos ou até mesmo partes do mesmo projeto. É o que ocorre se um módulo desejado de um software já existir, mas para reapropriá-lo para outro contexto do projeto demande muito tempo para separar o útil do desnecessário. Dessa forma o módulo é reescrito completamente, ao invés de reutilizado;

- Viscosidade:

  Viscosidade pode ser dividida em 2 formas: Viscosidade do projeto e viscosidade do ambiente. Sempre que presentes com uma necessidade de mudança do código, engenheiros possuem diversas abordagens possíveis para realizar tal mudança. Algumas mudanças preservam o projeto, e outras deterioram o projeto como era delimitado. Quando métodos preservadores se tornam mais custosos de implementar que os métodos que degradam, a viscosidade do projeto é alta.

  Viscosidade do ambiente são fatores externos ao desenvolvimento que afetam eficiência e produtividade dos engenheiros. Com tempos de compilação longos, engenheiros podem preferir mudanças que não forcem recompilações longas, mesmo que não sejam otimizadas do ponto de vista de projeto;

Esses quatro sintomas citados são sinais de uma arquitetura inferior. Aplicações que exibem esses comportamentos sofrem de um projeto que dificulta a realização de tarefas por parte dos desenvolvedores.

Assim, foram desenvolvidos os princípios *SOLID*, que visam corrigir os problemas de softwares desenvolvidos utilizando *Orientação a Objeto* de forma impensada.

De forma a tornar projetos de software mais inteligíveis, flexíveis e sustentáveis a longo prazo, 5 princípios foram criados por Robert C. Martin [25]. Estes 5 princípios formam o mnemônico *SOLID*:

- Princípio de única responsabilidade (do inglês *Single-responsibility Principle*): Uma classe deve ter apenas uma única responsabilidade, de forma que a mudança de apenas uma especificação do software deveria modificar a especificação daquela classe;
- Princípio Aberto-Fechado (do inglês *Open-closed Principle*): Entidades de software devem ser abertas à extensão, mas fechadas à modificações;
- Princípio Substituição de Liskov (do inglês *Liskov substitution Principle*): Objetos em um programa devem ser substituíveis com instâncias dos seus subtipos sem alterar a corretividade de um programa;
- Princípio Segregação de Interface (do inglês *Interface segregation Principle*): Várias interfaces específicas de cliente são melhores que uma única interface de propósito geral;
- Princípio Inversão de Dependência (do inglês *Dependency inversion Principle*): Deve-se depender de abstrações e não dos objetos concretos. É possível observar na Figura 2.14 que o comum é utilizar camadas inferiores como a *Utility Layer* serem consumidas por componentes de camadas superiores como a camada *Policy*. Dessa forma, *Policy* necessita diretamente de resposta da *Utility* para realizar tarefas. Assim, os componentes superiores são limitados pelos componentes inferiores.

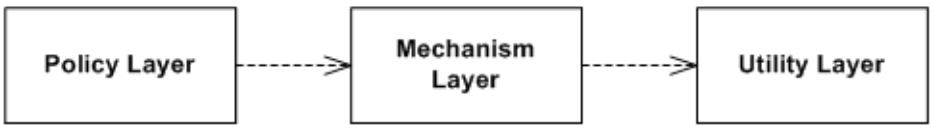


Figura 2.14: Arquitetura de camadas padrão

O que o princípio de Inversão de Dependência propõe é a arquitetura da Figura 2.15.

Dessa forma, cada camada depende unicamente de abstrações, ou interfaces para realizar suas tarefas. Agora, não existe dependência dos objetos concretizados, o que não limita o comportamento de camadas superiores, aumentando flexibilidade e sustentabilidade de código.

2.8 Frameworks

No desenvolvimento de software, a utilização de bibliotecas e *frameworks* é essencial para reduzir tempo de trabalho e aumentar eficiência.

Frameworks são um conjunto de códigos comuns entre diversos projetos de software provendo uma funcionalidade genérica. A utilização de *framework* altera a forma como a aplicação é desenvolvida e executada, contrário da utilização de bibliotecas, que apenas fornecem funcionalidades.

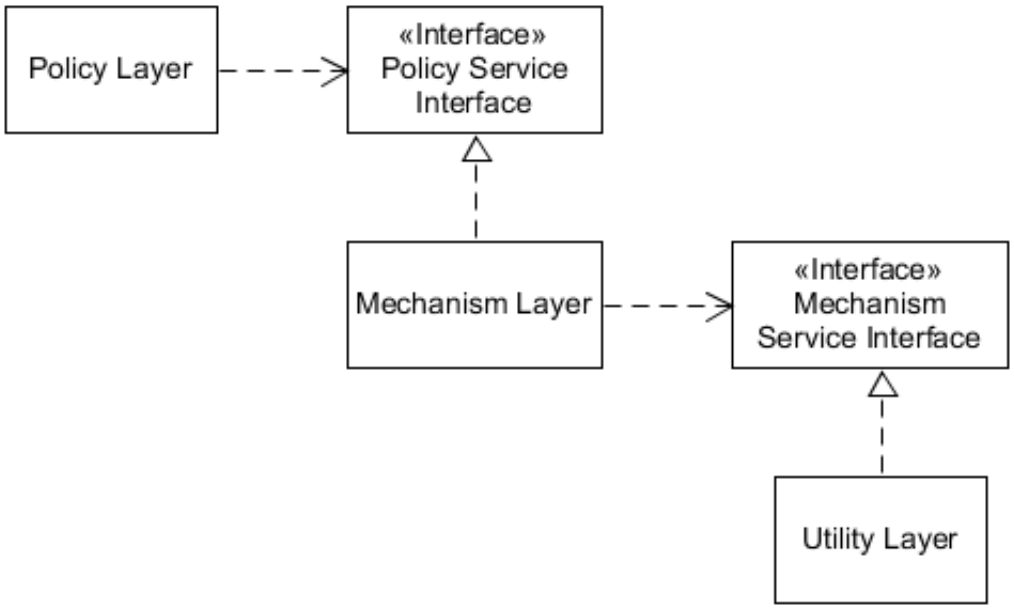


Figura 2.15: Arquitetura de camadas com inversão de dependência

2.8.1 ESP-IDF

ESP-IDF é o *framework* oficial de desenvolvimento para ESP32 e ESP32-S utilizado em Windows, Linux e macOS [26]. É de uso desejável para este projeto por permitir utilização de recursos como escrita de registradores, controle por API's do ESP nativo e otimizar a utilização de recursos da placa ESP32.

Espressif provê recursos de hardware e software para auxiliar desenvolvedores de aplicações utilizar o hardware ESP32. O *framework* de desenvolvimento ESP-IDF é projetado para desenvolvimento de projetos IoT com Wi-fi, Bluetooth, gerenciamento de energia e outras características de sistema [26].

2.8.2 Flutter

É de interesse deste projeto desenvolver uma aplicação móvel para captura de dados via BLE que seja multiplataforma. Assim, a utilização de Flutter foi escolhida com esse objetivo.

Flutter é um *framework open-source* desenvolvido pela Google [27]. É capaz de, com um único conjunto de códigos, gerar aplicações para Android, iOS e Web. O foco é permitir que desenvolvedores produzam projetos com alta performance competitiva com aplicativos construídos em ambientes nativos. Aplicativos escritos em Flutter utilizam a linguagem Dart [28].

Durante o desenvolvimento, aplicativos Flutter executam dentro de uma máquina virtual em Dart que oferece recarregamento rápido (do inglês *hot reload*), de mudanças na página sem a necessidade de recompilação.

Quando o aplicativo está pronto para ser utilizado pelo usuário final, Flutter gera uma versão de *release* que é compilada diretamente em código de máquina, seja Intel x64 ou ARM, ou em Javascript para aplicativos Web.

2.9 Aprendizado de Máquina

Adotar treinamentos de redes neurais artificiais (RNAs) para reconhecimento de atividades humanas como fala e escrita tem demonstrado sua efetividade ao extraírem *features* discriminativas de sequências de dados computados sem pré-processamento de sensores utilizados junto ao corpo [29]. Um tipo de atividade humana que será estudado ao longo deste projeto, com semelhanças do reconhecimento de fala, é a classificação de sinais de uma língua de sinais.

Existem muitas formas de resolver um problema que requer a utilização de aprendizado de máquina (do inglês *machine learning*). Uma das abordagens atuais de resolução de problemas complexos cujas saídas atuais dependem temporalmente de saídas anteriores como HAR (do inglês, *Human Activity Recognition*) é o uso de redes neurais recorrentes (do inglês *Recurrent Neural Networks*) [29].

2.9.1 Redes Neurais Recorrentes

Do inglês *Recurrent Neural Network*, RNNs são uma classe de RNAs que permite que as saídas dos passos anteriores sejam utilizadas como entradas no passo atual. Para se caracterizar como uma Rede Neural Recorrente, uma rede deve possuir a estrutura de nós como na Figura 2.16 [30], com cada estado oculto h_t sendo calculado utilizando o estado oculto h_{t-1} com a entrada x_t.

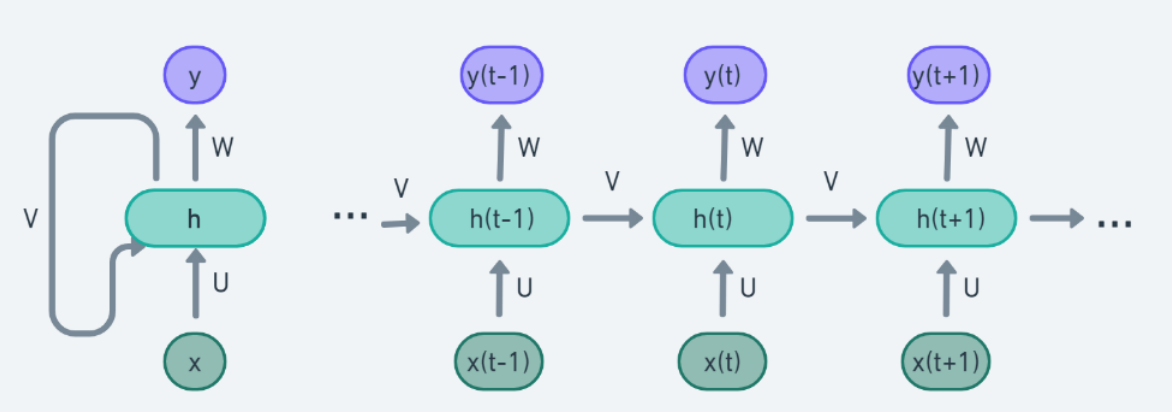


Figura 2.16: Rede Neural Recorrente

Cada saída y_t é calculada utilizando h_t e x_t, para cada passo, os sinais h_t são realimentados.

Esse comportamento difere de redes neurais tradicionais, que apenas processam o dado da entrada atual, ignorando sua dependência com estados ocultos anteriores. Assim, torna-se não adequados ao processamento de sequências temporais.

De forma a estabelecer capacidade de análise de estados anteriores, Redes Neurais Recorrentes possuem conexões internas que a permitem aprender dinâmicas temporais de dados sequenciais [29].

A Figura 2.17 demonstra um nó de uma RNN simples (SRN), com uma função de ativação F, estado oculto atual h_t, saída atual y_t, a entrada atual x_t e estado oculto anterior h_{t-1}.

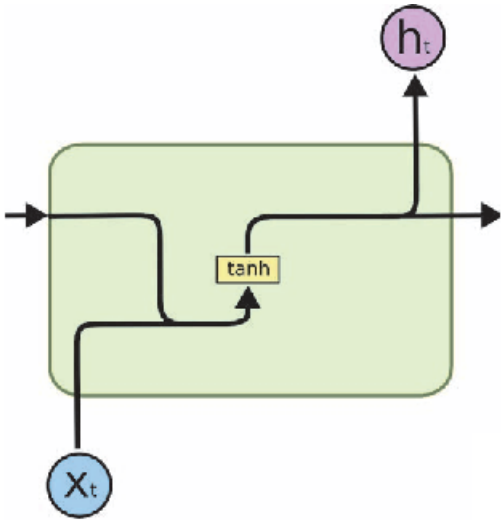


Figura 2.17: Diagrama esquemático de uma célula RNN [5].

É possível observar que as Equações

$$h_t = F(W_h h_{t-1} + U_h x_t + b_h) \tag{2.4}$$

e

$$y_t = F(W_y h_t + b_y) \tag{2.5}$$

demonstram a obtenção do estado oculto atual e da saída atual. Ambas são descritas com pesos tais como: W_h, o peso que relaciona estado oculto anterior com o atual; U_h, o peso que relaciona entrada com estado oculto atual conectados; e W_y, o peso de conexão oculta para saída. Além dos pesos, há ainda b_h e b_y, que são termos de viés para estado oculto e saída, respectivamente. Por fim, a função de ativação representada em F pode ser escolhida dependendo da propriedade desejada. Alguns exemplos de funções de ativação são: sigmoide (σ), tangente hiperbólica ($tanh$) e ReLU (do inglês *Rectified Linear Unit*).

A utilização de RNNs é comum na literatura de captura de cinemática de mãos humanas, e uma das redes recorrentes utilizadas em trabalhos recentes é a LSTM como visto em *Boon et al.* (2020) [6].

2.9.2 LSTM

As redes *Long Short-Term Memory* (LSTM) apresentaram uma solução para o problema do *Vanishing Gradient* [31]. Substituindo os nós comuns por células de memória que possuem recorrência interna e externa.

Uma célula de memória contém novos parâmetros e portas(do inglês, *gates*). Essas portas controlam quando um estado oculto anterior deve ser esquecido e quando deve ser atualizado com novas informações. A função de cada célula é mostrada na Figura 2.18.

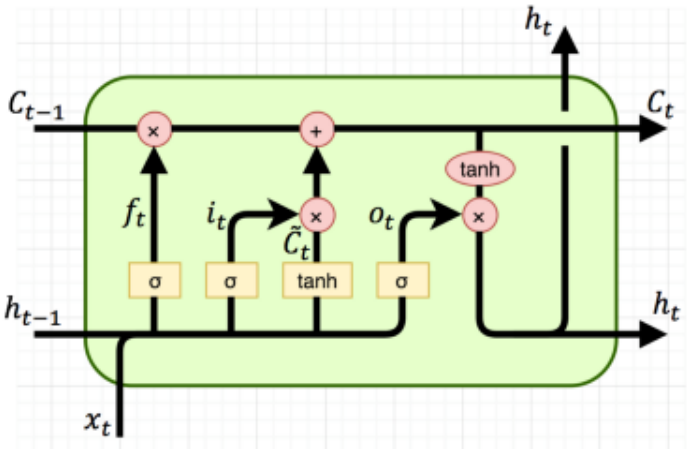


Figura 2.18: Célula LSTM [5].

Temos os parâmetros: i_t como porta de entrada, controlando o fluxo de novas informações entrando na célula; f_t porta de esquecimento, determinando quando esquecer um conteúdo relacionado ao estado interno; o_t a porta de saída, controlando a informação que sai da célula; g_t a porta de modulação da entrada, que é a porta de entrada principal; C_t o estado interno, que lida com a recorrência interna da célula; h_t o estado oculto tal como na Seção 2.9.1.

A forma como cada componente se comporta é de acordo com

$$i_t = \sigma(b_i + U_i x_t + W_i h_{t-1}) \tag{2.6}$$

$$f_t = \sigma(b_f + U_f x_t + W_f h_{t-1}) \tag{2.7}$$

$$o_t = \sigma(b_o + U_o x_t + W_o h_{t-1}) \tag{2.8}$$

$$g_t = tanh(b_g + U_g x_t + W_g h_{t-1}) \tag{2.9}$$

$$C_t = f_t c_{t-1} + g_t i_t \tag{2.10}$$

e

$$h_t = tanh(C_t)o_t \tag{2.11}$$

de modo a realizar o treinamento da rede. Os parâmetros a serem aprendidos pela rede são b, U e W de cada porta. Com isso, é possível analisar a solução para o problema do gradiente. Para tal análise, é necessário observar o gradiente de uma LSTM. Como na LSTM, C_t(estado interno oculto) é uma função de f_t (a porta de esquecimento), i_t (porta de entrada) e g_t (porta de modulação).

A rede LSTM aprende a decidir quando uma informação anterior não deve mais ser propagada adiante, possuindo maior liberdade de decidir quanta memória irá possuir para o aprendizado atual [32].

Outra forma de rede recorrente utilizada em trabalhos de classificação de dados sequenciais é a GRU, de maneira a realizar uma comparação com a LSTM [33].

2.9.3 GRU

GRU (do inglês, *Gated recurrent unit*), da mesma maneira que LSTM, possui portas de entrada que modulam o fluxo de informação dentro da sua unidade. No entanto, não possui memórias de célula separadas.

GRU possui apenas dois *gates*, um *gate* de *reset* e um de *update*, como visto na Figura 2.19. Como não há um *gate* de saída, GRUs utilizam o estado oculto para transferir informações.

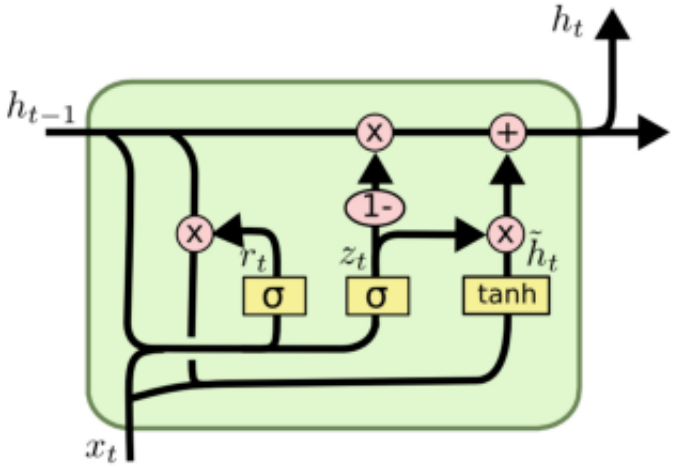


Figura 2.19: Célula GRU [5].

A forma de ativação de h_t de uma GRU em um tempo t é uma interpolação entre a ativação h_{t-1} e g_t como segue em

$$h_t = (1 - z_t)h_{t-1} + z_t g_t \tag{2.12}$$

em que z_t é a porta de atualização (do inglês, *update gate*) que decide o quanto uma unidade atualiza sua própria ativação, ou conteúdo. O portão de atualização é computado por

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{2.13}$$

e é similar à porta de esquecimento e portão de entrada de uma célula LSTM, pois decide qual informação manter e qual apagar.

O portão de modulação g_t é computado de forma semelhante da LSTM,

$$g_t = tanh(W_g x_t + U_z(r_t \odot h_{t-1}) \text{ [34]}) \tag{2.14}$$

com r_t a porta de reset e $\odot$ é multiplicação de elemento a elemento (do inglês, *Element-wise multiplication*). Quando r_t tem valor próximo de 0, a porta de reset age como se a sequência visse a primeira aparição de uma sequência. A utilidade do *gate* de *reset* é de decidir o quanto da informação anterior manter. É computado da forma

$$r_t = \sigma(W_r x_t + U_r h_{t-1}). \tag{2.15}$$

GRU tem menos operações quando comparadas com uma LSTM e por isso tem sido considerada mais eficiente [34].

2.10 Estado da Arte

Abordagens para reconhecimento e mapeamento de cinemática de juntas humanas são de dois tipos atualmente: Visão Computacional e Sensores *Wearable*. Como cada abordagem possui vantagens e desvantagens, escolheu-se alguns trabalhos de estado da arte para cada método.

2.10.1 Visão Computacional

A análise com base em Visão Computacional utiliza uma câmera RGB e, muitas vezes, um sensor de profundidade para captura de dados. Assim, são aplicados algoritmos de visão computacional nos dados coletados para analisar os sinais das mãos e expressão facial e corporal [35].

Appenrodt et al. (2010) [35], propôs um sistema com diversos tipos de câmeras de forma a compará-las quanto à performance em segmentação. As imagens de uma câmera eram utilizadas como entrada de dados para análise. Foram comparadas câmeras de cor única, câmeras stereo e câmera termal, de forma a apontar vantagens e desvantagens. Os sinais utilizados foram o alfabeto manual (A-Z) e números(0-9). O sistema foi alimentado com um banco de dados de 30 vídeos

por sinal, utilizando 20 sequências para treinamento e 10 para testar o algoritmo. Obteve-se uma acurácia média de 98% para dados estáticos e 95,7% para sinais contínuos utilizando Modelos Ocultos de Markov (do inglês *Hidden Markov Models* - HMM).

Em *Köpüklü et al.* (2019) [36], foi proposto um sistema de reconhecimento em tempo real de sinais por meio de sequências de vídeos. O desafio consistiu em permitir que o sistema percebesse quando um sinal termina e outro começa, que sinais realizados devem ser reconhecidos somente uma única vez, além de que o sistema deveria ser projetado levando em conta custo de memória e energia. Assim, foi realizado uma estrutura de Rede Neural Convolucional (do inglês *Convolutional Neural Network* - CNN) utilizando Protocolo de Janelas Deslizantes.

Com esse protocolo, o receptor sempre envia uma mensagem ACK de recebimento da mensagem, e o transmissor verifica se recebe o ACK dentro de um período de tempo, senão reenvia a mensagem. Foi implementado um detector de sinais utilizando CNN simples e um classificador de sinais detectados utilizando CNN profunda. A utilização da distância de Levenshtein como métrica de avaliação permitiu avaliar falsas classificações, detecções múltiplas e não detecções. Por fim, a arquitetura do projeto foi avaliada contra dois *Data Sets* disponíveis: EgoGesture e NVIDIA Dynamic Hand Gesture, que também requerem detecção temporal e classificação de sinais performados. O modelo utilizado para classificador, ResNeXt-101, obteve acurácia *offline* de $94,04\%$ e 83,82% para modalidade de profundidade nos *benchmarks* da EgoGesture e da NVIDIA. Para tempo real, obtiveram acurácia de 91,04% e 77,39% respectivamente.

2.10.2 Sensores Wearable

A abordagem mais comum em estudos de reconhecimento de língua de sinais tem sido a de utilização de visão computacional, como citado acima. No entanto, essa abordagem é limitada pelo ângulo de visão dos equipamentos de captura, condições de iluminação, além da utilização de *Machine Learning* de forma mais elaborada. Isso implica em uma colaboração entre especialistas da área e alto custo de hardware.

Após o desenvolvimento de sensores MEMs, o custo de medir a cinemática humana utilizando sensores inerciais reduziu bastante. Desta forma, torna-se viável sua utilização em massa para esse tipo de aplicação, além de diminuir consideravelmente a quantidade de dados a ser processada por algoritmos de reconhecimento.

Nessa abordagem, é utilizada uma luva com sensores capazes de medir os parâmetros de LIBRAS, com acelerômetros e giroscópios. Estes sensores conseguem medir tanto a configuração quanto a orientação das mãos. Por outro lado o uso de sensores flexores (do inglês *Flex Sensors* permitem a medição apenas a configuração das mãos [37].

No entanto, esta abordagem é limitada pelo hardware específico utilizado para construir a base de dados de reconhecimento de sinais. Ainda há outra limitação, a baixa portabilidade. Visto que a proposta da maioria dos projetos nessa área é integrar surdos na comunidade ouvinte, é um tanto incômodo utilizar diariamente uma luva com componentes eletrônicos para poder se comunicar.

Patil et al. (2014) [38], desenvolveram uma luva única com 5 sensores flex em cada dedo, de

forma a mapear a flexão completa de uma dobra (dedo fechado), dobra parcial e dedo aberto. No entanto, línguas de sinais complexas utilizam, além da configuração de mãos, orientação, que não é possível medir apenas com sensores flexores. Então, *Das et al.* (2016) [39], além dos 5 sensores de flexão, utilizaram um giroscópio no centro das costas da mão de forma a também medir sua orientação. Dessa forma, a taxa de reconhecimento média foi de 86,67% para 20 sinais dinâmicos.

Mummadi et al. (2018) [40], utilizaram apenas 5 sensores IMU, um em cada ponta do dedo, de forma a adquirir *pitch*, *raw* e *yaw* de cada dedo para reconhecer 24 letras ASL estáticas. Foi utilizado um classificador baseado em Florestas Randômicas, reportando uma acurácia final de 92,95%.

Carvalho, A (2019) [41] propôs um sistema de luva para a captura de dados de Datilologia. O trabalho confunde o conceito de um sinal da língua LIBRAS e de configurações de mãos do Alfabeto Manual. Os valores eram todas as posições de mão do alfabeto manual (A-Z) em português com dados estáticos. Cinco sensores inerciais eram acoplados a luva e o microcontrolador se comunicava via Bluetooth com um *smartphone*. Os classificadores utilizados foram: i)Perceptron multicamadas (MLP); ii) K vizinhos mais próximos (KNN); e iii) Redes defunções de base radial (RBFN). Obteve desempenho na RBFN de 99,84% de acurácia no conjunto de dados de teste, 99,69% no KNN e 99,93% no MLP.

A proposta do projeto descrito neste documento em relação ao *Carvalho, A* (2019) [41] é de realizar um sistema de captura de dados não estáticos de maneira a englobar sinais em LIBRAS que, contrário das posições de mão de Datilologia, necessitam de sequências temporais para serem interpretados. Além disso, é desejado um protótipo mais portátil, com alimentação incluída e de pequeno porte.

Boon et al. (2020) [6], propuseram um modelo RNN com camadas LSTM configurado para entregar a melhor performance classificando 27 palavras da ASL.

O banco de dados foi criado com 12 pessoas do campus da universidade, para coletar os 27 sinais. Como nenhuma dessas pessoas fazia parte da comunidade de surdos, todas receberam vídeos dos sinais que seriam realizados com 3 meses de antecedência de forma a praticarem.

A cada sujeito foi solicitado que vestisse a luva na mão direita e realizasse os sinais. Cada sinal era realizado num tempo de 10 a 15 segundos, com vídeos auxiliares para visualização do usuário. Com isso foi construído um banco de 38.451 amostras.

A luva possui 6 IMUs, uma em cada dedo, além de uma no dorso da mão como pode ser visto na Figura 2.20. O sensor IMU foi customizado com uma placa de fabricação própria de forma a reduzir o tamanho. Dessa forma o tamanho foi reduzido em 77%. O acelerômetro retorna a ACC (do inglês *Acceleration*) em m/s² a uma taxa de amostragem de 100 Hz e o giroscópio retorna o AGR (do inglês *Angular Rate*) em graus/s também a uma taxa de amostragem de 100 Hz.

Então, foi aplicado uma fusão de sensores, com a calibração do acelerômetro, giroscópio e magnetômetro em ângulo Euler (ELA, 100 Hz) ou quatérnions (QTR, 100 Hz).

Dessa forma, foi analisado como cada característica (do inglês *feature*), ACC (aceleração), AGR (velocidade angular), ELA (ângulo de Euler) em x, y e z e QTR (Quatérnions em a,b,c,d)

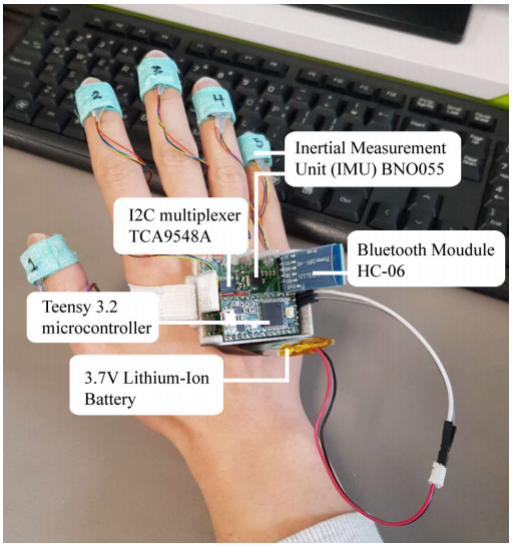
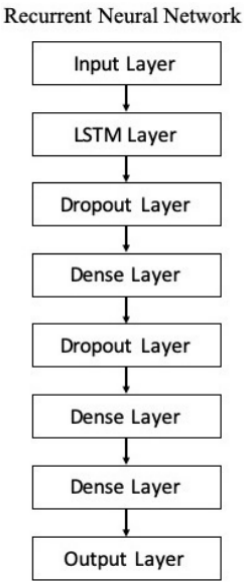


Figura 2.20: Proposta do artigo de 2020 [6].

afetaria a performance do modelo de classificação. Segue a divisão de características:

- C1: QTR
- C2: ELA
- C3: AGR
- C4: ACC
- C5: QTR + ELA + AGR + ACC
- C6: QTR + ELA + AGR
- C7: QTR + ELA + ACC
- C8: QTR + AGR + ACC
- C9: ELA + AGR + ACC
- C10: QTR + ELA
- C11: QTR + ACC
- C12: QTR + ACC
- C13: ELA + AGR
- C14: ELA + ACC
- C15: AGR + ACC



(a) Classificador

Category	Features	AR (%)
C_1	QTR	99.65
C_2	ELA	99.70
C_3	AGR	99.56
C_4	ACC	99.66
C_5	QTR + ELA + AGR + ACC	99.85
C_6	QTR + ELA + AGR	99.84
C_7	QTR + ELA + ACC	99.84
C_8	QTR + AGR + ACC	99.82
C_9	ELA + AGR + ACC	99.82
C_{10}	QTR + ELA	99.83
C_{11}	QTR + AGR	99.83
C_{12}	QTR + ACC	99.82
C_{13}	ELA + AGR	99.78
C_{14}	ELA + ACC	99.82
C_{15}	AGR + ACC	99.79
	Average	99.67

(b) Resultados

Figura 2.21: Proposta de classificador e resultados do artigo [6]

Com as categorias de 1 a 4 consistindo apenas de um tipo de dado IMU; Categoria 5 contendo todos os dados; Categorias de 6 a 9 contém todas menos 1 dado de IMU; e por fim de 10 a 15 possui apenas 2 dados de IMU. O interesse foi em caracterizar os grupos mais representativos de dados. O classificador proposto está mostrado na Figura 2.21(a), consistindo de uma camada LSTM seguida de camadas de *Dropout* (camada de descarte) e *Dense* (camada completamente conectada).

Utilizando os dados IMU com *10-fold*, os autores relatam os seguintes resultados mostrados na Figura 2.21(b). Com acurácia média acima de 99% na melhor rede pode-se perceber que os fatores QTR, ELA, AGR e ACC da fusão sensorial representam bem os parâmetros: Orientação, Configuração de Mão, Movimento e Ponto de Articulação.

Em relação ao trabalho de *Boon et al.* (2020) [6], o trabalho neste documento propõe a utilização de captura de dados via aplicativo móvel multiplataforma para LIBRAS, ao invés de ASL. Além disso, serão utilizados apenas 5 sensores, um em cada dedo, ao invés de 6.

Nesse capítulo, foi descrito a fundamentação teórica necessária para compreensão do projeto desenvolvido.Discutiu-se trabalhos semelhantes da área e como estes propõem melhora de resultados.

A seguir haverá uma descrição da metodologia empregada e decisões de projeto para manufatura e desenvolvimento do hardware, assim como as arquiteturas de software e de dados envolvidos para realizar os protocolos de comunicação.

Capítulo 3

Metodologia

Este capítulo tem a intenção de explicar as metodologias utilizadas no desenvolvimento do projeto.

O projeto a ser desenvolvido é o de uma luva de captura de dados cinemáticos do tipo de sinais de LIBRAS, a transferência de dados para um aplicativo móvel multiplataforma e a classificação destes dados utilizando redes neurais. É de interesse de estudo também, como a taxa de amostragem dos sensores da luva interferem na acurácia da rede. Portanto, além da luva e do aplicativo móvel que captura dados, foi desenvolvido um módulo de divisão de frequência que cria dados sub amostrados.

Como pode ser visto na Figura 3.1, o projeto consiste em: uma luva que possui uma placa de desenvolvimento com a responsabilidade de ler os sensores IMU usando I²C, condicionar os dados, e enviar por BLE para um aplicativo móvel; Um aplicativo móvel, que por sua vez, recebe os dados, escreve em arquivo que é levado para um computador; No computador há tratamento dos dados obtidos e seu armazenamento em um banco de dados; Por fim, há o treinamento de rede rede neural recorrente usando Keras quando completado o banco de dados.

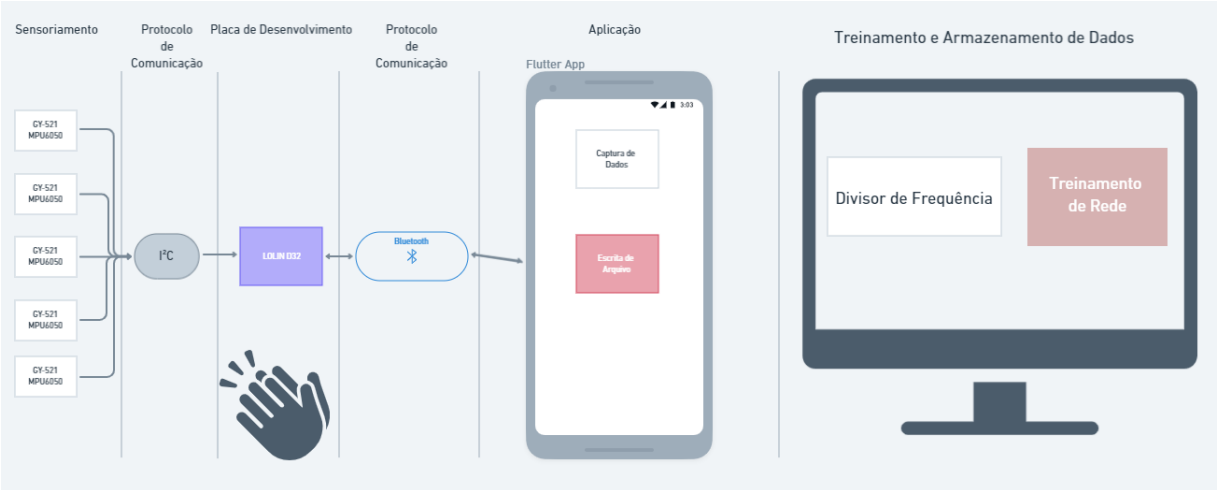


Figura 3.1: Visão geral do projeto

É desejado que seja possível fazer a captura de dados via interface com usuário, dessa forma, é inserido um botão multifuncional no projeto da luva. Esse botão é conectado em um GPIO da placa de desenvolvimento, tornando possível em software a definição de funções diversas para a leitura de botão apertado. Nas próximas seções serão apresentados fluxogramas com ícones de "Botão Iniciar Leitura"e "Botão Próxima Leitura"que representam essas funções criadas.

Para desenvolver o projeto, a metodologia utilizada seguiu duas frentes: Gerenciamento de Projeto e Arquitetura de Software. Assim, o gerenciamento foi realizado utilizando Kanban [42] simples e gerenciamento de versão de código, para isto, utilizou-se as ferramentas: GitHub [43] e Trello [44]. Como o trabalho foi realizado em dupla e a maior parte dele foi realizada de forma remota devido à pandemia de Coronavirus, tornou-se necessário reuniões diárias via *Zoom* [45].

A outra frente, Arquitetura de Software, foi atuada decidindo como realizar o trabalho no ambiente do microcontrolador e no aplicativo. Dessa forma, a primeira etapa foi o mapeamento dos requisitos do projeto, sendo eles módulo de transferência de dados, de aquisição e de treinamento.

O projeto de Hardware evoluiu ao longo do desenvolvimento, inicialmente utilizando um Arduino Nano com módulos Bluetooth externos e uma luva de couro, para em seguida favorecer a utilização de um ESP32 com módulo Bluetooth e WiFi integrados e uma luva mais flexível. Assim, o projeto foi dividido nas 4 etapas a seguir:

- Etapa 1 - Luva e Microcontrolador: Repositório *Glove Reader* [46].

  É a etapa inicial do projeto, envolvendo a fabricação da luva e desenvolvimento do software embarcado inicialmente no *Arduino*, posteriormente no ESP32 e por fim com LOLIN D32.

  Foram realizados testes e leituras de sensores com microcontroladores de maneira a identificar o melhor microcontrolador para a aplicação. Foram implementadas a leitura de sensores MPU6050 no *Arduino* e no ESP32, assim como transferência por *Bluetooth*.

  O objetivo é permitir a captura e envio de dados de cada IMU conectada ao ESP32 para uma aplicação externa.

- Etapa 2 - Aplicativo Mobile: Repositório Flutter App [47].

  Desenvolvimento da aplicação externa de captura de dados. Aplicativo em Flutter com a responsabilidade de obter dados via *Bluetooth Low Energy* com o ESP32 e depois LOLIN D32.

  Há a criação de protocolo de comunicação entre microcontrolador e aplicativo mobile, onde o primeiro byte é utilizado para indicar o estado da comunicação.Além disso criou-se o código necessário para escrita de arquivos obtidos pela comunicação.

- Etapa 3 - Divisão de Frequência e recuperação dos dados: Repositório Pré-Processamento [48].

  Tratamento dos dados obtidos dos arquivos. Análise de ruído, gráficos de todos os eixos de cada acelerômetro e giroscópio presentes nos sensores.

  A divisão de frequência de dados foi realizada de forma a gerar dados com frequências de amostragem distintas para simular diferentes meios de comunicação dos dados. Dessa forma

será possível obter dados de leituras de sensores com frequências de amostragem compatíveis com os canais Wi-Fi, Serial, e *Bluetooth* Clássico.

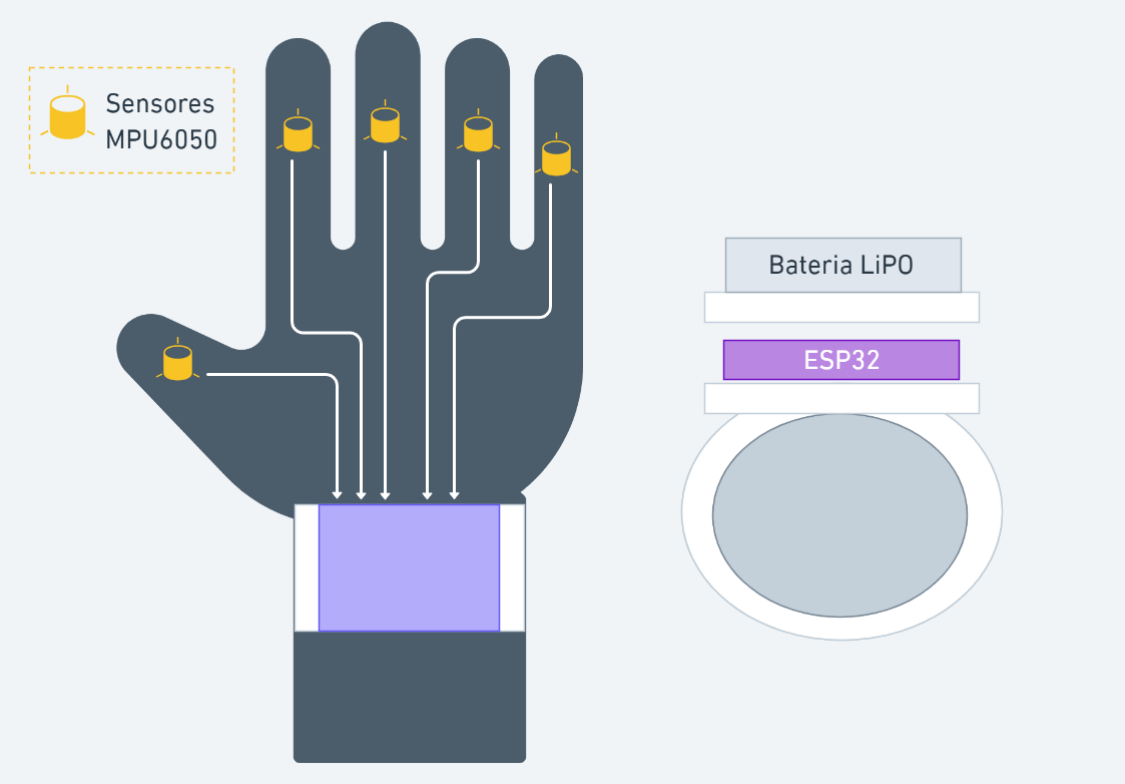
Essa etapa tem o propósito de realizar pré-processamento dos dados coletados pela aplicação Flutter, construir a estrutura de dados e realizar a divisão de frequência.

- Etapa 4 - Classificadores: Repositório Rede Neural [49].

  Treinamento de redes neurais com dados intactos e com dados após serem tratados pelo divisor de frequência. O último repositório do projeto ficou responsável por esse treinamento da rede e armazenamento dos pesos dos melhores resultados.

3.1 Projeto da Luva

A luva possui 5 sensores MPU6050, um módulo LOLIN D32 e sua bateria LiPo. A Figura 3.2 apresenta uma desenho esquemático da estrutura proposta para a luva.



(a) (b)

Figura 3.2: Ilustração final da luva da mão direita

A Figura 3.2(a) mostra a posição dos sensores que ficam localizados sobre as falanges distais dos dedos indicador, médio, anelar e mínimo e na falange proximal do polegar [50].

A Figura 3.2(b) mostra um corte transversal da luva indicando a disposição do módulo ESP32 e da bateria no pulso, empilhados um sobre o outro. Esta disposição não gera peso sobre o dorso

da mão aumentando o conforto durante a execução dos gestos.

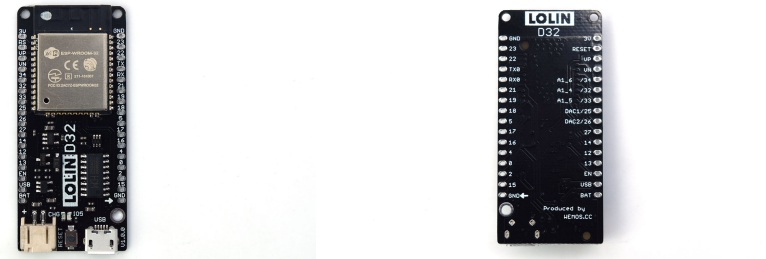
3.2 Microcontrolador

Inicialmente decidiu-se utilizar um *Arduino* Nano em conjunto com um módulo *Bluetooth* HC-05. De forma a reduzir o tamanho do projeto como um todo, optou-se por um microcontrolador que possuísse módulos de comunicação integrados. Após testes de comunicação do *Arduino* Nano com intuito de comparar com os de outro microcontrolador, foi escolhida a placa LOLIN D32, que utiliza um ESP32.

A decisão da placa levou em conta vários aspectos, os mais importantes foram os seguintes:

- A placa periférica deve ser capaz de regular a tensão de uma bateria LiPo de uma célula (3.7v) para a faixa de alimentação do microcontrolador e dos sensores.
- Possuir módulo *Bluetooth* integrado.
- Preço baixo.
- Baixo consumo de energia.
- Possuir um número de GPIOs suficiente para interfacear com 5 sensores MPU6050.

Escolheu-se então a placa LOLIN D32 da Wemos mostrada na Figura 3.3. Esta placa possui todos os requisitos mencionados anteriormente sendo baseada no chip ESP-WROOM-32 [51]:



(a) Vista superior (b) Vista inferior

Figura 3.3: LOLIN D32

A placa LOLIN D32 possui dimensões de 57mm×25,4mm com peso de 6,1g.

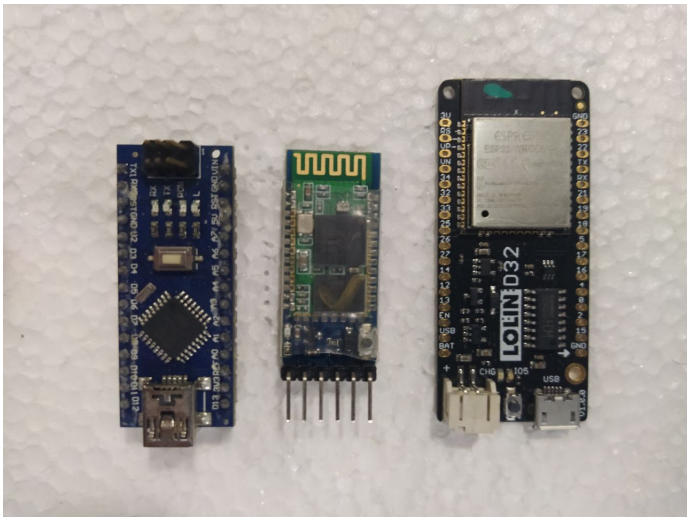


Figura 3.4: Arduino Nano, Módulo HC-05 e LOLIN D32, respectivamente

Peso leve e dimensões significantemente menores que um *Arduino* Nano em conjunto com um módulo *Bluetooth* como vistos na Figura 3.4 também se adequaram mais para o objetivo de reduzir o tamanho do protótipo. É possível observar, que além de tamanho agregado da placa, seria requerido roteamento de fios de forma a utilizar o módulo HC-05 com Arduino.

3.3 Alimentação

O projeto possui algumas particularidades quanto a bateria. De maneira que o projeto seja portátil, leve e atender às exigências do microcontrolador escolhido e os sensores, a bateria deve ser a menor possível capaz de alimentar o projeto. Assim, a bateria deve dificultar minimamente os movimentos das mãos.

A luva será alimentada por bateria, permitindo assim que ela seja totalmente móvel e portátil. Para isso, o tipo mais comum de baterias para aplicações embarcadas deste tipo são as baterias LiPo. Deve ser levado em conta a faixa de alimentação do microcontrolador e dos sensores, pois dependendo desta faixa, a tensão elétrica da bateria poderá precisar ser regulada para uma tensão menor dentro da faixa de alimentação destes componentes.

Fatores cruciais para a bateria a ser usada:

- **Tamanho**: A circunferência média de um pulso humano feminino é de 13,7cm e masculino de 16,2cm [52]. Para ser posicionada no dorso do pulso de um ser humano adulto, a bateria deve possuir no máximo 1/3 do tamanho de um pulso humano. Avaliando pelo tamanho feminino, que é menor, a bateria deve possuir não mais que 45,3 mm de largura.
- **Faixa de tensão**: Faixa de tensão utilizada pelo sistema com microcontrolador e sensores. No caso do LOLIN D32 com MPU6050, é necessária uma faixa de tensão de 3,7V a 3,3V.
- **Processo de recarregamento**: Bateria de fácil recarregamento, via módulo de carrega-

mento embutido no microcontrolador escolhido (LOLIN D32), ou módulo de carregamento próprio com indicativo de carga concluída.

- **Capacidade de carga**: Capacidade de manter a aplicação durante a captura de sinais por pelo menos 30 minutos.

Pela documentação da Wemos, LOLIN D32 é compatível com baterias Lipo 3,7V [53]. O processo de recarregar a bateria é mais simples quando utilizado com um LOLIN D32, pois, ao conectar um cabo USB, é possível carregar a bateria por meio do microcontrolador.

Assim, considerando os requerimentos descritos anteriormente, a bateria escolhida é do modelo LiPo 902540 [54] , que possui as seguintes características:

- **Capacidade:** 800 mAh
- **Tensão:** 3,7 V
- **Tamanho:** 42mm × 25mm × 9,8mm
- **Peso:** 19g
- **Fabricante:** Limskey

Figura 3.5: Bateria LiPo 902540

A Figura 3.5 mostra uma foto da bateria escolhida para esta aplicação.

3.4 Sensoriamento

Neste trabalho escolhemos a abordagem de utilização exclusivamente de sensores inerciais, visto a capacidade de rastrear o movimento de cada dedo no espaço ao longo do tempo e o baixo custo de implementação. Cada sensor deve ser capaz de retornar os valores de velocidade angular e de aceleração nas três dimensões, permitindo assim uma análise da cinemática mais precisa de cada dedo, estes dados serão úteis para estimação do sinal feito pela mão naquela janela de tempo.

O sensor escolhido foi o GY-521, com módulo IMU modelo MPU6050, mostrado na Figura 3.6.

Figura 3.6: Sensor GY-521

Com esse sensor é possível obter informações de aceleração e giroscópio de cada eixo. Os fatores que influenciaram na escolha do GY-521 com o chip MPU6050 são:

- **Custo baixo**: Menor que 20 reais. No caso de uma compra internacional (China), o preço fica em torno de 4 reais;
- **Simplicidade de uso**: Há várias bibliotecas na internet que servem como base para nossa aplicação, além do mesmo permitir a leitura de dados com o protocolo I²C, a tornando mais simples;
- **Disponibilidade no mercado nacional de eletrônica**: O componente é facilmente encontrado no mercado nacional, permitindo uma prototipagem fácil e rápida.

3.5 Software Embarcado

Na primeira etapa, o projeto no LOLIN D32 tomou a forma da Figura 3.7. Há duas maneiras de trabalhar com o LOLIN D32:

- Usando a biblioteca oficial de desenvolvimento ESP-IDF, com as ferramentas nativas de gravação e Debug (GDB, do inglês, *GNU Debugger* e gravador da própria fabricante escrito em *Python*) [55];
- Usando a IDE do *Arduino* e os códigos desenvolvidos pela comunidade para facilitar o uso do LOLIN D32, tornando-o um dispositivo similar ao *Arduino*;

Decidiu-se usar a ESP-IF pois não possui limitações de desempenho e nem depende da comunidade que desenvolve essas soluções.

Dessa forma, há duas bibliotecas externas às APIs do ESP-IDF, que são I²CBus [56] e MPU6050 [57], utilizadas para leitura dos sensores por meio do protocolo I²C.

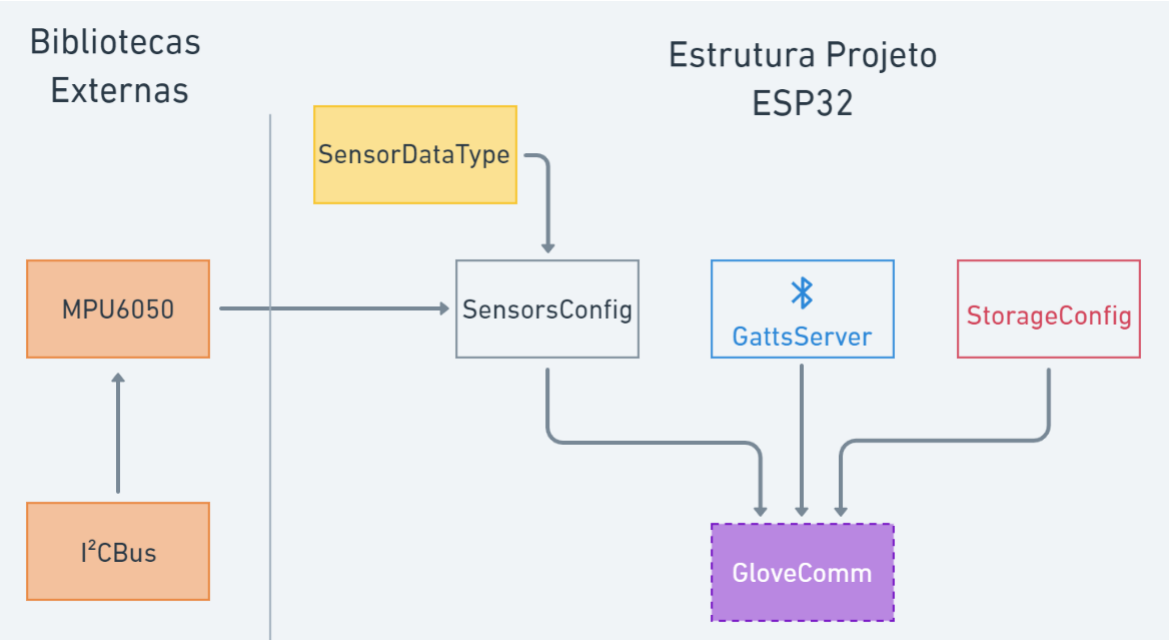


Figura 3.7: Diagrama de blocos da estrutura principal do projeto no microcontrolador. Bloco principal é GloveComm.

Dessa forma, foram construídos três módulos internos utilizando a metodologia SOLID. Cada módulo tem uma funcionalidade única e independente do outro, permitindo alterações individuais sem efeitos colaterais.

- SensorsConfig: É o módulo de configuração de sensores, responsável por iniciar o barramento I²C com cada sensor individualmente e inicializar o módulo de cada sensor com filtro passa-baixas e escala de medida.
- GattsServer: Inicializa o servidor *Bluetooth Low Energy* utilizando GATT;
- StorageConfig: Armazena arquivos localmente utilizando o sistema de arquivos do LOLIN D32 Spiffs;

Definimos o *SensorDataType* como o tipo de dado que declara *HandReading*, mostrado em detalhes na Figura 3.8. Assim, *Handreading* representa um bloco de leitura de sensores IMU com um *timestamp*, englobando todas as leituras de um número N de IMUs e o tempo medido no microcontrolador na leitura da IMU N/2. Dessa forma o *timestamp* representaria a medida média das leituras dos N sensores. No nosso caso, são 5 sensores, e N/2 = 2,5, foi decidido por medir o *timestamp* na leitura do sensor número 3. Cada dado do tipo IMU possui 3 leituras de acelerômetro e 3 de giroscópio, cada uma ocupando um número de 8b

Como visto na Figura 3.7, o código principal no LOLIN D32 é o GloveComm, que importa todos os outros módulos: SensorsConfig, GattsServer e StorageConfig.

O Algoritmo 1 apresenta a sequência de execução do programa GloveComm no LOLIN D32.

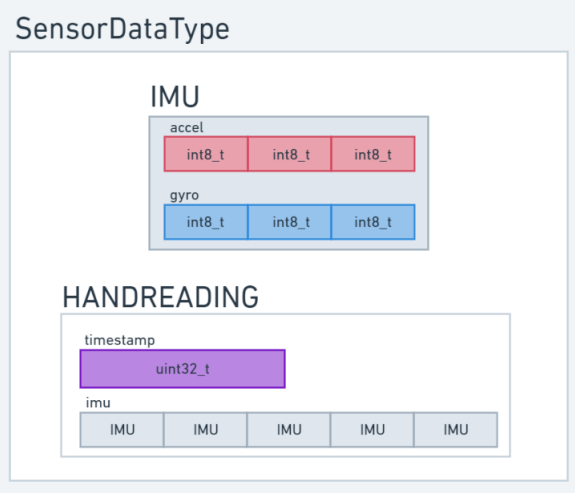


Figura 3.8: Tipos de dados dos Sensores

Algoritmo 1 Execução no LOLIN D32

1: Inicializa o barramento I²C
2: Inicializa Todos os Sensores MPU6050
3: Inicializa todos os GPIO necessários
4: **while** Botão Iniciar não Pressionado **do**
5: Verifique estado do Botão
6: **end while**
7: Inicia variável cnt.
8: **for** contagem $cnt < BUFFER/2$ **do**
9: Capture Leitura dos Sensores em memória RAM.
10: **end for**
11: **while** Botão Continuar Leitura não Pressionado **do**
12: Verifique estado do Botão
13: **end while**
14: **for** contagem $cnt < BUFFER$ **do**
15: Capture Leitura dos Sensores em memória RAM.
16: **end for**
17: Relaciona memória RAM dos dados com memória utilizada para envio por BLE
18: Inicia servidor de GATTS do BLE.

O algoritmo utiliza as bibliotecas I²CBus [56] e MPU6050 [57] nos primeiros 2 passos, para inicializar o barramento e preparar o LOLIN para receber leituras dos sensores.

Em seguida são inicializados todos os GPIO necessários para fazer chaveamento entre os 5 sensores para capturar a leitura um por vez.

Então, o programa aguarda a ação do usuário por meio de um botão que indica o início da leitura dos sensores.

É iniciado uma variável de contagem, que irá servir de separador de sinais. Para capturar sinais, é necessário armazenar um conjunto de dados *Handreading* em sequência que representem o sinal. Como um dado Handreading possui 34 bytes em memória, tornou-se necessário calcular o tamanho máximo de armazenamento desse tipo de dado em memória RAM, ou memória flash do chip ESP32. Esse tamanho máximo permitido pela aplicação descrita no Algoritmo 1 foi chamado de BUFFER.

O valor do BUFFER obtido foi de 1902 posições de *Handreading* (de 34 bytes cada) à uma taxa de obtenção de dados de 317Hz, que é uma frequência explicada adiante na seção de resultados. Com isso, éramos capazes de realizar $1902/317 = 6$ segundos de leitura.

Dessa forma, no passo 8, são realizadas meia capacidade máxima de leitura, que seriam 3 segundos, tempo suficiente para realizar sinais de uma mão comuns. Então, o programa entra em espera da atuação do usuário para realizar a segunda leitura.

Após a segunda leitura, é inicializado um servidor *Bluetooth Low Energy* para enviar os dados coletados para qualquer cliente que se conecte a ele.

A Figura 3.9 apresenta o fluxograma de execução dos estados gerais do software embarcado.

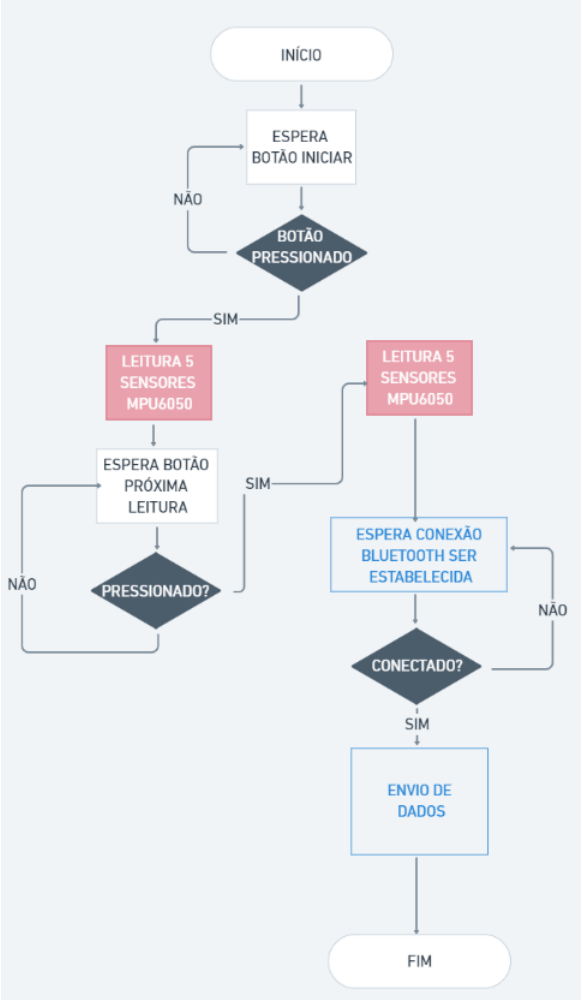


Figura 3.9: Fluxograma do software no LOLIN D32

O fluxograma representa visualmente o Algoritmo 1 comentado anteriormente. Captura de sensores, espera pela indicação de desejo de leitura pelo usuário e início de um servidor *bluetooth* para envio dos dados adquiridos.

3.6 Leitura de Dados

O protocolo para a comunicação entre os sensores e o microcontrolador deve permitir a conexão de vários dispositivos ao mesmo barramento, limitando assim a quantidade de fios presentes na luva.

De forma a obter leituras de todos os sensores MPU6050 foi utilizado o barramento I²C. O Algoritmo 2 apresenta os passos para a inicialização dos sensores por meio de I²C.

Algoritmo 2 Inicialização dos 5 sensores MPU6050

1: Inicialize o barramento I²C, com portas selecionadas
2: Defina o endereço para o SPI observar o dado, nesse caso é $AD0$ quando seu valor é LOW.
3: Defina taxa de amostragem 1000Hz, o máximo possível permitido pela API do MPU6050 utilizada.
4: Defina escala do acelerômetro de 8.192LSB/g ou +/- 4G.
5: Defina escala do giroscópio de 500 graus por segundo ou 65,5 LSB/(°/s).
6: Envie comandos de inicialização a todos os sensores
7: Inicialize 5 GPIOs responsáveis por alterar o valor de $AD0$ de cada MPU6050.

Assim, após o Algoritmo 2 ser executado, o programa está pronto para ler dados dos sensores.

Como o microcontrolador só fornece duas posições de memória 0x68 e 0x69 para este barramento, foi necessário desenvolver uma lógica de leitura dos sensores. Essa lógica é apresentada no Algoritmo 3.

Algoritmo 3 Leitura dos 5 sensores MPU6050

1: **for** cada sensor $imu_i \in IMU$ **do**
2: Configura valor LOW de $AD0$ para o sensor imu_i(resultando no endereço 0x68) e um valor $HIGH$ para os demais sensores (resultando no endereço 0x69)
3: Leia os dados dos registradores do sensor imu_i no endereço 0x68 definido no passo anterior.
4: **end for**

A partir da leitura de dados, o programa os armazena em memória flash, permitindo acesso por outros módulos do *GloveComm*.

3.7 Tratamento de Dados

O formato de cada pacote de leitura de dados de todos os sensores foi agrupado conforme apresentado na Figura 3.10.

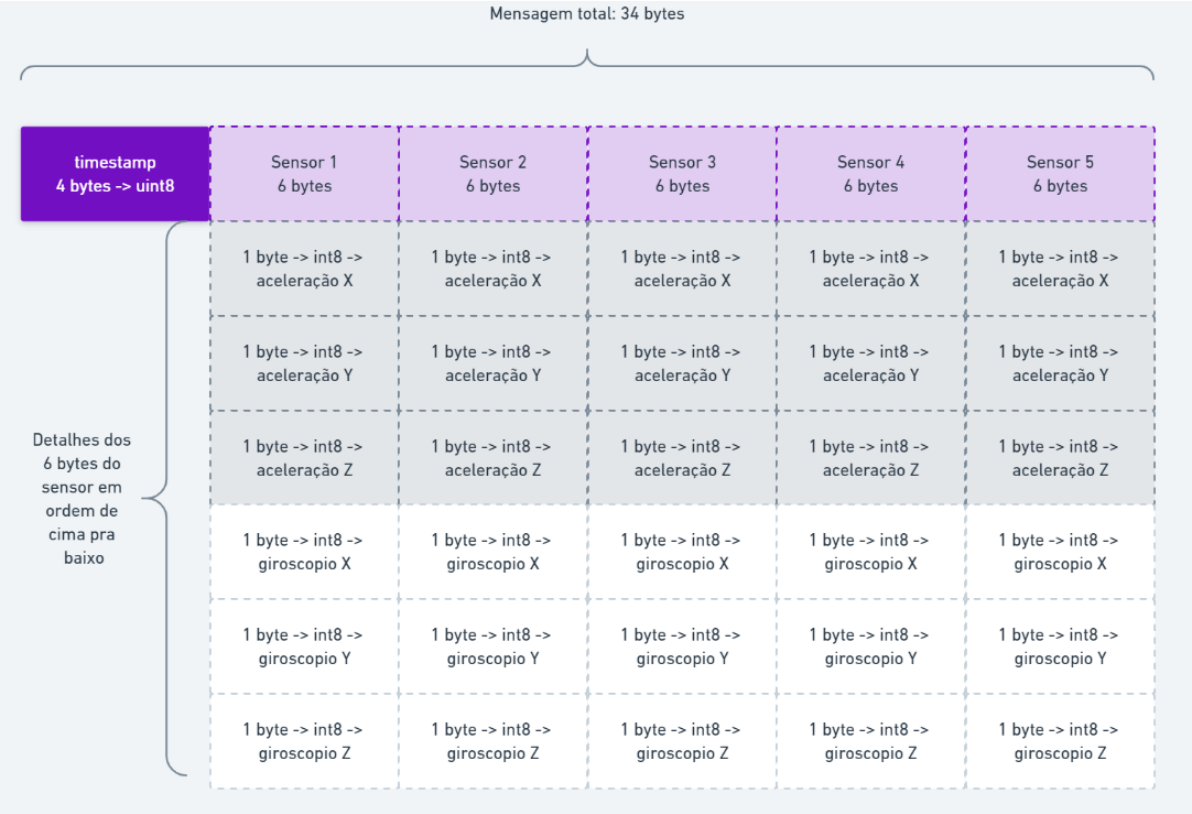


Figura 3.10: Tipo de estrutura de dado *HandReading*

Os valores dos acelerômetros e dos giroscópios são representados originalmente em 12 bits, portanto, para armazená-los em 8 bits, foi necessário realizar a operação de deslocamento (do inglês *shift*) para a direita em 4 bits. Dessa forma os 8 bits mais significativos são armazenados de acordo com o formato apresentado na Figura 3.10. A seguir foi construído um pacote de leituras de IMU incluindo um marcador de tempo (do inglês *timestamp*) de 4 bytes, para auxiliar na captura dos dados na frequência de amostragem desejada, e assim poder estudar o quanto ela afeta a acurácia da rede neural. Deste modo cada amostra da estrutura de dados *HandReading* possui 34 bytes.

Como citado na Seção 3.5, foi construído um *BUFFER* de 1902 posições de *HandReading*, amostrados a 317Hz, permitindo aquisição de 6 segundos de dados.

Cada sinal então consistirá basicamente de uma lista da estrutura de dados *HandReading* ilustrada na Figura 3.10, sendo cada elemento do *buffer* um quadro (do inglês *frame*) que representa a postura manual capturada pela luva em um determinado instante de tempo. Assim que o usuário concluir o movimento do sinal, essa lista de leituras é enviada por BLE para o celular, nesse momento esses dados passam a ser enxergados como sequência de bytes.

A sequência de bytes recebida pelo celular é escrita em arquivos .csv, que são levados para um notebook e decodificados em uma estrutura de dados equivalente a anterior, porém usando a estrutura lista da linguagem *Python* [58] . Com isso, é possível reconstruir a estrutura *Handreading* mapeada no chip ESP32 no ambiente de desenvolvimento *Python*.

Tendo os dados equivalentes em *Python*, pode-se usar os mesmos para o divisor de frequência, e posteriormente para o treinamento da rede neural recorrente. A rede recebe 30 dados de entrada, os 3 eixos de acelerômetro e giroscópio de cada sensor dos 5 dedos. Cada byte é normalizado, isto é, dividido por 255, de modo que a rede neural recebe os dados em ponto flutuante no intervalo de 0 a 1.

3.8 Protocolos de Comunicação

O protocolo responsável pela comunicação sem fio deve ser capaz de transmitir os dados dos sensores coletados pelo microcontrolador para o dispositivo móvel onde está a parte de interpretação dos sinais. É necessário que a comunicação seja sem fio para não limitar o movimento da mão do usuário, deixando o uso seja mais prático e confortável.

Além disso, terá que ser rápido o suficiente para poder coletar a maior quantidade de dados permitindo assim um volume maior de informação para a interpretação e inferência de um sinal para uma determinada janela de tempo.

3.8.1 Protocolo Bluetooth Low Energy

O protocolo BLE necessita que os dados sejam salvos em memória flash, ou seja, utiliza NVS(do inglês *Non-Volatile Storage*) no ESP-IDF.

Um protocolo simples foi implementado para captura de dados para treinamento entre LOLIN D32 e o aplicativo em Flutter, bastando informar, no primeiro byte, a natureza da sequência de dados a ser enviada. O Algoritmo 4 apresenta os passos para a inicialização e configuração do servidor GATTS no BLE.

Algoritmo 4 Servidor GATTS BLE

```
Inicia partição NVS por meio da API ble do esp-idf
if Erro de inicialização de NVS then
    Inicia partição NVS
end if
Checar erro da partição NVS
Liberar memória pré-alocada de módulos de Bluetooth inutilizados
Iniciar Controller
if Erro de inicialização de Controller then
    Para a Execução do programa
end if
Habilitar Controller
if Erro de habilitação de Controller then
    Para a Execução do programa
end if
Iniciar bluedroid
if Erro de inicialização de bluedroid then
    Para a Execução do programa
end if
Configura função de callback do GATTs
if Erro de configuração GATTs then
    Para a Execução do programa
end if
Configura função de callback do GAP
if Erro de configuração GAP then
    Para a Execução do programa
end if
Registra perfil de aplicação a ser utilizada, com lista de serviços e características.
if Erro de registro then
    Para a Execução do programa
end if
Configura MTU para 517 bytes locais
if Erro de configuração de MTU then
    Para a Execução do programa
end if
```

Após todas as configurações realizadas no Algoritmo 4, o servidor BLE é executado, permitindo visualização e conexão de outros dispositivos além de escutar quaisquer requisições feitas por clientes conectados.

3.8.2 Comunicação entre LOLIN D32 e Aplicativo

O ESP32 envia pacotes de acordo com o primeiro byte da sequência. Se o byte tiver valor de PRIMEIRO_ESTADO_ITERACAO, que é 0, inicia o envio de dados para o aplicativo. Caso seja 1, valor de DADOS_SENDO_ENVIADOS, continua enviando dados até o final do tamanho armazenado em memória RAM.

3.9 Aplicação Mobile

Na segunda etapa do projeto, utilizamos a plataforma Flutter para desenvolver um aplicativo que pudesse ser gerado tanto para Android quanto iOS. A Figura 3.11 ilustra a funcionalidade do aplicativo como fluxograma.

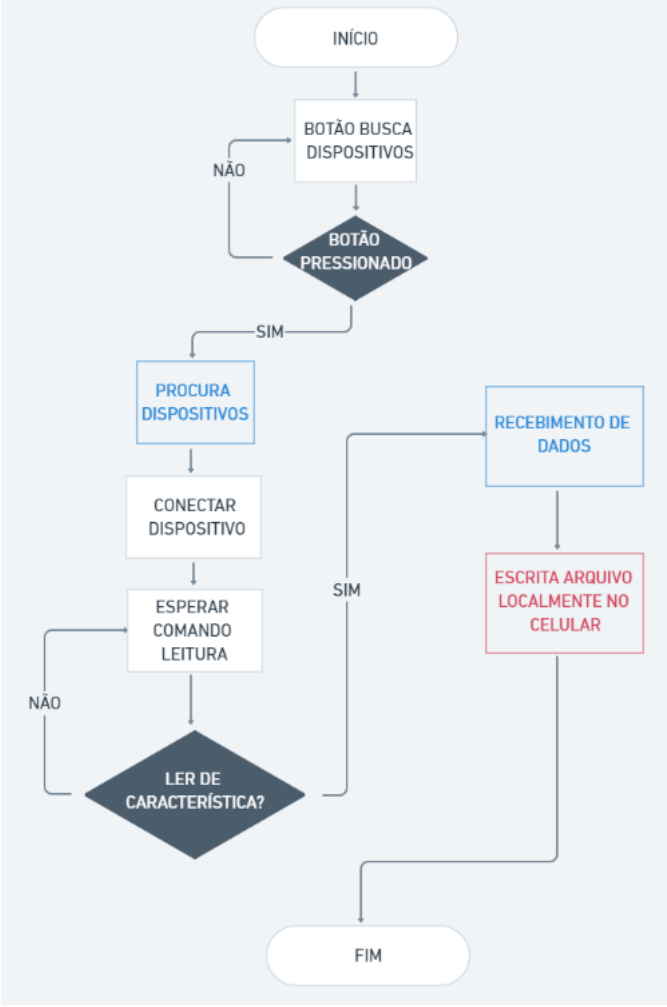


Figura 3.11: Fluxograma do projeto no Aplicativo Flutter

O aplicativo desenvolvido permite a conexão com o microcontrolador via *Bluetooth Low Energy*, realização de requisições de leitura de dados e armazenamento de dados localmente. A aplicação serve como cliente conectando à um dispositivo que agirá como Servidor.

Os elementos funcionais em Flutter são *Widgets*. As telas do aplicativo eram *Widgets* que englobavam os comportamentos específicos de cada tela. As telas foram separadas por requisitos tais quais:

- **Tela de descoberta:** Tela responsável por mostrar todos os dispositivos *Bluetooth* que estejam em modo de publicar dados e receberem requisições de conexões.
- **Tela de Conexão:** Tela com detalhes da conexão atual com um dispositivo específico, após esta conexão ser estabelecida.
- **Tela de Serviços e Características:** Detalhes de todos os serviços e características de cada serviço disponibilizados com o dispositivo agindo como Servidor. Para cada característica visível na tela, há possíveis interações, leitura de dados, escrita de dados e ajuste de MTU.

  Com a requisição da leitura de dados, é realizado um ACK simples para captura de todos os dados do chip ESP32 e logo em seguida é realizado uma rotina de armazenar os dados em arquivo .csv na memória interna do celular.

Assim, a aplicação utiliza dos *Widgets* para permitir a visualização de todos os requisitos citados, além da interação com o usuário ser agradável.

3.10 Armazenamento

Dados recebidos do LOLIN D32 são armazenados na memória local do celular. O armazenamento é realizado em uma pasta chamada *Glove-Reader* no formato .csv. Assim, é criado um banco de dados com repetições de cada sinal a partir dos arquivos em .csv. Pelo Algoritmo 1 e o tamanho do BUFFER escolhido anteriormente, é possível obter dois sinais, cada um com cerca de 3 segundos de duração.

3.11 Divisor de Frequência

A terceira etapa do projeto engloba o pré e pós processamento de dados capturados pelo LOLIN D32, enviados para a Aplicação desenvolvida na segunda etapa e guardados em arquivos.

A partir dos dados em .csv, é possível reconstruir as estruturas *HandReading* originais com remoção dos *timestamps*, dessa forma são construídos objetos *IMU* para frequência máxima de dados.

De forma a agregar na análise do projeto, é possível sub-amostrar os objetos *IMU* em diferentes frequências de amostragem.

O exemplo do processo de subamostragem pode ser visto na Figura 3.12. Conhecendo a frequência inicial de captura, medida pelos *timestamps* de início e fim de um sinal, é possível

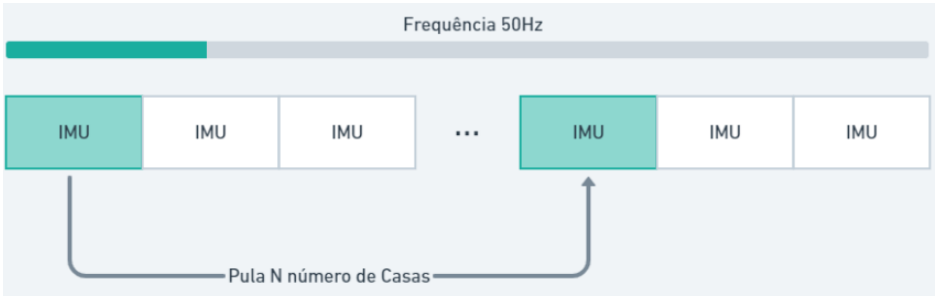


Figura 3.12: Exemplificação do algoritmo de subamostragem

percorrer a lista de objetos decimando de N amostras, isto é, pulando de um número N em N de casas para obter dados a uma frequência específica, submúltipla da frequência de amostragem original.

Os dados resultantes do módulo divisor de frequência não possuem um tratamento por um filtro passa-baixas para evitar efeito de *aliasing*. A rede então receberá dados sub-amostrados de forma a simular os atrasos dos protocolos de comunicação incapazes de manter uma taxa de transmissão de dados para dispositivos externos igual a taxa de captura de leituras dos sensores por I^2C.

3.12 Classificadores

Todo o projeto de classificadores foi realizado em ambiente simulado em Nuvem.

3.12.1 Processamento em Nuvem

O software *Google Colab* [59] permite a utilização de recursos da nuvem da *Google* de forma a treinar as redes neurais sem o uso dos recursos computacionais (taxação) do computador local. Deste modo, as redes podem ser treinadas em paralelo sem acréscimo de tempo de desenvolvimento.

3.12.2 K-Fold

Tradicionalmente, projetos em *Machine Learning* dividiam dados de entrada entre teste e de treinamento [60] . No entanto, essa metodologia não era confiável, pois os dados de teste nem sempre possuem as mesmas características dos dados de treinamento e isso afeta a medida da acurácia do modelo. Assim, foi escolhido a metodologia de Validação Cruzada (do inglês *Cross Validation*), pois divide os dados em diversos grupos, ao invés de apenas dois [60].

Assim, os dados dentro de uma classe de sinais são embaralhados de forma a reduzir qualquer ordenação e divididos em 10 *folds*. Um dos 10 *folds* é escolhido para teste, enquanto os outros são reservados para treinamento.

Esse processo é repetido para todos os 10 *folds*, calculando para cada iteração, com um dos

folds separados para esse teste, a acurácia da rede neural. Cada acurácia é armazenada e no fim é calculado a acurácia média da rede.

3.12.3 Redes Neurais Utilizadas

As redes utilizadas foram de dois tipos, LSTM e GRU. Essas topologias são utilizadas para problemas com cunho temporal ou sequencial. Estas redes conseguem manter estados anteriores e processar dados com entradas atuais e antigas. GRU é uma topologia proposta para casos com bancos de dados reduzidos, demonstrando maior acurácia no geral para esse tipo de problema [33]. Assim, temos:

- LSTM: Foi utilizada uma LSTM profunda e uma simples, de forma a comparar a acurácia entre diferentes tipos de topologia;
- GRU: Foram utilizadas duas GRU, uma simples e outra profunda. Assim, da mesma forma que a comparação em LSTM, é possível identificar vantagens e desvantagens de utilização de mais camadas;

Dessa forma é possível comparar qual das duas abordagens de estrutura de rede rende melhor acurácia para um problema com banco de dados reduzido.

Para o treinamento da rede, foram carregados os arquivos do banco de dados. De cada arquivo os valores de todos os sensores de todos os dedos foram concatenados em listas de inteiros. Cada elemento desta lista teve o valor normalizado por uma divisão por 255 obtendo resultados de 0 a 1 em todos os casos. E então, o valor era salvo em *float* de 16 bits, obtendo assim listas *numpy* de *float* de 16 bits.

Finalmente, a rede com desempenho superior foi escolhida para ser treinada com dados de frequências sub-amostradas de dados. Assim, é possível medir o quanto a queda de frequência de amostragem afeta o desempenho da rede.

Neste capítulo foram vistos o projeto da luva, envolvendo microcontrolador, alimentação, sensoriamento e software embarcado. Foram discutidos as decisões tomadas para o software permitir a leitura de sensores e armazenar dados de forma específica. A partir dos dados armazenados, foi necessário o desenvolvimento de um protocolo de comunicação para a transmissão dos dados para um ambiente de pré processamento. Para envio de pacotes de dados capturados na frequência máxima que cinco MPU6050 lidos por I²C permitiam, utilizou-se o *Bluetooth Low Energy*.

A partir de então, foi discutido o processo de desenvolvimento de um aplicativo multiplataforma, e como seria a comunicação entre ele e o LOLIN D32. Com os dados enviados para o aplicativo e armazenados em memória interna do celular, houve um pré-processamento para dividir frequência dos dados, de forma a permitir uma análise de acurácia de rede treinada para dados obtidos em diferentes frequências. Por fim, houve a discussão de quais topologias de rede neural,

de que forma dividir dados para treinamento e que o ambiente de treino utilizado seria na *Google Colab* na Nuvem.

Assim, o próximo capítulo tratará de todos os testes realizados de transmissão de dados, protótipos em hardware, aplicação mobile, sinais capturados e análise de ruído de sensores e de acurácia das redes.

Capítulo 4

Resultados Obtidos

Este capítulo tem a intenção de demonstrar e explicar os resultados obtidos a partir das metodologias mencionadas no capítulo anterior

Os resultados que serão abordados nesse capítulo são o fruto do trabalho seguindo a metodologia proposta, assim, analisaremos as taxas de amostragem obtidas de acordo com cada tecnologia utilizada, a seleção dos componentes, demonstração do protótipo e o consumo de energia do mesmo em diferentes estados, o aplicativo móvel desenvolvido, o banco de dados, as acurácias obtidas com diferentes tipos de redes neurais assim como o efeito da frequência tanto no tempo de treinamento quanto na acurácia do resultado

4.1 Taxa de Amostragem

Foram realizados diversos testes com o objetivo de obter uma taxa de amostragem alta o suficiente que permitisse a tradução em tempo real dos dados obtidos em um determinado microcontrolador considerando a comunicação entre o mesmo e o dispositivo que irá processar os dados.

Foram realizados diversos testes com os microcontroladores *Arduino* e ESP32 a fim de mensurar a máxima frequência de amostragem atingível por cada sistema para a aquisição dos sinais dos 5 sensores. Ainda foi medida a máxima frequência atingível considerando o tempo do canal de comunicação a ser utilizado com o dispositivo móvel.

4.1.1 Arduino

Um dos primeiros testes foi feito com o microcontrolador Arduino Nano. Com o objetivo de testar a taxa de transferência entre a leitura de sensores e o envio da mensagem, os testes foram divididos entre testes de comunicação sem fio via *Bluetooth* e testes de leitura dos dados dos sensores.

Um teste realizado foi o da taxa de transferência entre cinco sensores MPU6050 e um Arduino Nano a fim de verificar a viabilidade do projeto. Se frequência das leituras for maior que 100Hz,

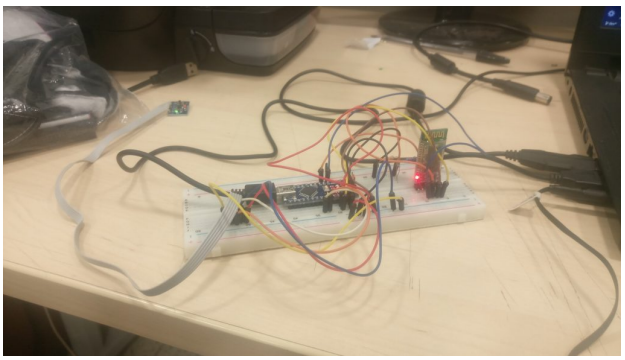


Figura 4.1: Teste feito com Arduino, um sensor MPU6050 e um Módulo Bluetooth HC-05.

significando mais de 100 leituras de cada sensor por segundo, então o sistema permite uma captura fiel de dados de sinais manuais.

Foi feita uma função que realiza a troca entre sensores MPU6050 (ou GY521) de forma a mudar o sensor sendo lido 5 vezes obtendo 6 leituras, 3 leituras dos Acelerômetros e 3 leituras dos Giroscópios.

Esse processo é realizado 100 vezes. Antes do processo iniciar, o tempo inicial é setado (t_{init}), e após a execução do processo o tempo final (t_{final}) é medido.

É calculada a diferença $t_{final} - t_{init}$. Essa diferença é dividida por 100, de forma a estimar o tempo médio entre cada leitura dos 5 sensores.

O resultado da média de 100 medidas de 5 *IMU*s trocando entre as leituras por chaveamento e sem envio *Bluetooth* é de 7,816ms que corresponde a uma frequência de 142Hz.

Esse resultado mostra que a taxa de amostragem está acima de 100Hz, porém a margem para o tempo de comunicação por *Buetooth* com o dispositivo móvel será muito apertada. No entanto uma solução alternativa foi procurada com o ESP32 de forma a não só executar com taxa de transferência maior, mas tornar o projeto mais compacto sem um módulo *Bluetooth* externo como seria necessário com um *Arduino*.

4.1.2 ESP32

Foram realizados os testes no ESP32 para verificar as hipóteses de maior taxa de transferência e deixar o projeto mais compacto , conforme mostrado na Figura 4.2. O ESP32 é um chip microcontrolador com módulos Wi-Fi e *Bluetooth* embutidos.

- **ESP32 e 5 sensores** O primeiro teste realizado com o ESP32 visou testar a hipótese de que a transferência de dados de um sensor MPU6050 para o ESP32 seria mais rápida que a transferência entre um MPU6050 e um *Arduino*. O teste envolveu o uso de bibliotecas de leitura de tempo do *framework* esp-idf e da biblioteca de sistema operacional de tempo real embutida FreeRTOS.

  Definimos um taxa de amostragem de 500Hz no ESP32, isto é, metade da taxa de amostra-

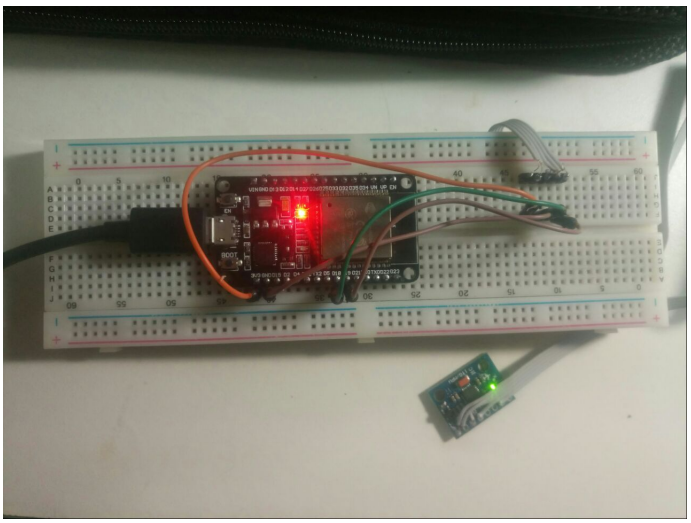


Figura 4.2: Testes realizados com um ESP32 e um Sensor MPU6050.

gem máxima. O resultado do teste foi de 2,710 ms, o que resulta numa taxa de amostragem dos 5 sensores de aproximadamente 370 Hz, que é maior que 100 Hz e mais rápida que a do *Arduino* Nano de 142 Hz.

- **ESP32 e 5 sensores com timestamp**

  Foi definido a taxa de amostragem no ESP32 com 500Hz, com leitura de *timestamp* entre a leitura do segundo e o terceiro sensor, de forma a popular os dados na estrutura *Handreading* mencionada anteriormente. Dessa forma, cada pacote lido possuía 34 bytes, contrário do teste anterior que possuía apenas 30 bytes. O resultado do teste foi de 3,155 ms, o que resulta numa taxa de amostragem dos 5 sensores de 317Hz.

  A Tabela 4.1 apresenta um resumo dos dados obtidos nos 3 experimentos.

Tabela 4.1: Comparação da taxa de captura de dados de 5 sensores

Microcontrolador	**Taxa de Amostragem**
Arduino Nano	142Hz
ESP32 (sem *timestamp*)	370Hz
ESP32 (com *timestamp*)	317Hz

Podemos concluir que o uso do microcontrolador ESP32 é cerca de $2,6$ vezes mais rápido que o uso de um Arduino para dados sem *timestamp*. Finalmente, com a utilização de *timestamp*, a taxa de amostragem tem uma queda de 14,32%.

4.2 Microcontrolador LOLIN D32

De forma a desenvolver uma plataforma mais reativa, que coletasse dados dos sensores e ao mesmo tempo enviasse os dados por BLE para o celular, escolhemos outro microcontrolador que não possui apenas um único núcleo de processamento (do inglês *Single-Core*). Isso faria com

que os tempos de espera necessários nas operações, seja de envio ou leitura de sensores, sejam paralelizados, reduzindo o tempo total de espera.

Assim, o *LOLIN D32* se mostrou adequado por possuir dois núcleos para a execução dos códigos. Além de possuir uma frequência de *clock* mais alta em cada core do que o *Arduino*.

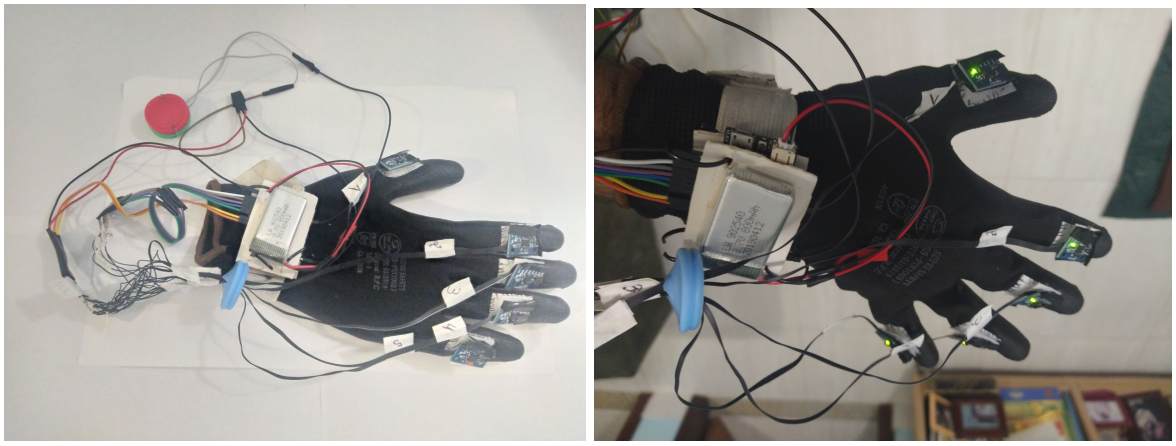
Tabela 4.2: Capacidade de processamento dos microcontroladores

	Quantidade de núcleos	**Frequência de *clock***
LOLIN D32	2	240 MHz
Arduino	1	16 MHz

Vale ressaltar que o *LOLIN D32* utiliza o mesmo chip que o ESP32, logo as operações de leitura de sensores e condicionamento de dados podem ser executadas com mais recursos e mais rapidamente, como apresentado na seção anterior. Dados da frequência de *clock* comparada entre *LOLIN D32* e um Arduino podem ser visualizados na Tabela 4.2.

4.3 Protótipo da Luva

A luva foi construída de acordo com a proposta da Figura 3.2, com o microcontrolador e a bateria acoplados no pulso conforme mostrado na Figura 4.3.



(a) Luva Completa (b) Unidade de controle

Figura 4.3: Protótipo final da Luva de aquisição de dados

A Figura 4.3(a) mostra o protótipo construído, a Figura 4.3(b) a unidade de controle baseada no Microcontrolador LOLIN-D32 PRO.

4.3.1 Sensoriamento

Cada sensor MPU6050 necessita de 5 fios, 2 de alimentação, 2 de dados e um de endereçamento quando lido por um barramento I²C. Os fios de alimentação 3.3V e o GND de todos os sensores foram roteados para mesmas portas de 3.3V e GND do LOLIN-D32 PRO . Os fios de dados

(SDA) e de clock (SCL) do protocolo I²C de cada sensor também foram roteados em conjunto (compartilhando o barramento I²C entre todos os sensores) fazendo com que a leitura de cada sensor dependa unicamente do endereço *AD*0 de cada sensor.

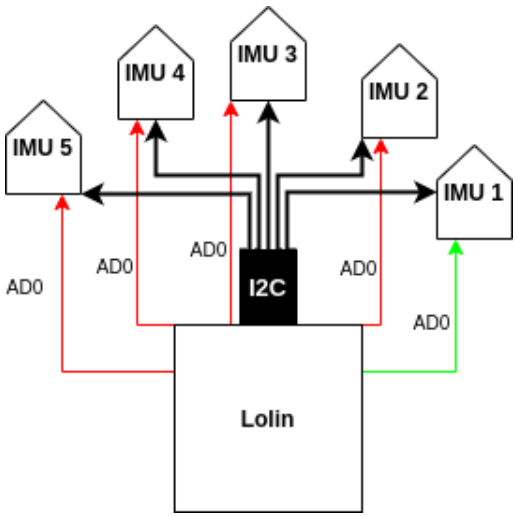


Figura 4.4: Leitura de um sensor por vez utilizando o AD0

Cada fio de *AD*0 dos sensores foi ligado a uma entrada GPIO específica do LOLIN-D32 PRO. Quando a leitura de um sensor determinado é desejada, basta escrever *LOW* no *AD*0 do mesmo e *HIGH* em todos os *AD*0 dos outros sensores. Dessa forma o endereço do sensor escolhido com *LOW* no *AD*0 seria de 0x68, enquanto que os outros ficam em 0x69.

Na configuração do protocolo I²C utilizando *Esp-idf*, selecionamos endereço de leitura de sensor como 0x68 e assim a leitura é realizada. Na Figura 4.4 pode ser visualizado o momento onde o LOLIN-D32 PRO deseja ler o sensor IMU 1, acionando o respectivo *AD*0 e desligando os outros, tornando possível a leitura desse sensor individualmente mesmo que barramento I²C esteja sendo compartilhado por todos.

4.3.2 Estrutura da Luva

Além da transmissão dos sensores e do roteamento de fios, o protótipo necessitou de um botão externo como mostrado na Figura 4.5. O botão age para indicar ações desejadas pelo usuário como iniciar a leitura de sensores.

A luva utilizada foi uma luva de construção leve, com uma fina camada de borracha nos dedos de forma a manter maleabilidade e conforto. O tecido é respirável no dorso da luva, facilitando uso contínuo e prolongado.

O peso do protótipo foi concentrado na região do pulso, como visto na Figura 4.3(b) de forma a facilitar a realização dos movimentos da mão. O roteamento de fios de cada sensor para o microcontrolador atravessa um separador de fios em azul na Figura 4.5. Pelo roteamento manter os fios pouco flexionados mantendo liberdade de movimento, além de apoiar o peso deles no pulso. A sensação de utilizar a luva é a de que não há fios ligados em cada dedo.

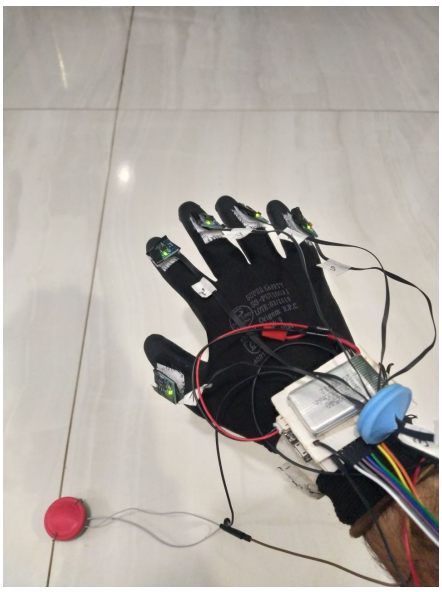


Figura 4.5: Protótipo colocado na mão de um usuário.

(a) Protótipo antigo com sensores costurados direto na luva

(b) Protótipo novo com sensores fixados por meio de velcro

Figura 4.6: Método de fixação dos sensores

As conexões com os fios de cada sensor foram soldadas e isoladas umas das outras. Após o isolamento de cada sensor individual, estes foram presos ao tecido da luva por meio de velcro. A região isolada por fita isolante do sensor foi colada com velcro no tecido costurado na luva como visto na Figura 4.6(b). Com isso, observou-se que os sensores permaneciam mais firmes do que se tivessem sido costurados diretamente na luva como feitos no protótipo anterior da Figura 4.6(a).

4.4 Aplicativo Mobile

As telas do aplicativo são responsáveis pela obtenção de conexões possíveis e conectar dispositivo ao celular. A partir de então é possível ver as características de cada serviço do dispositivo

conectado. Finalmente é possível realizar operações de Escrita e Leitura como requisição GATT.

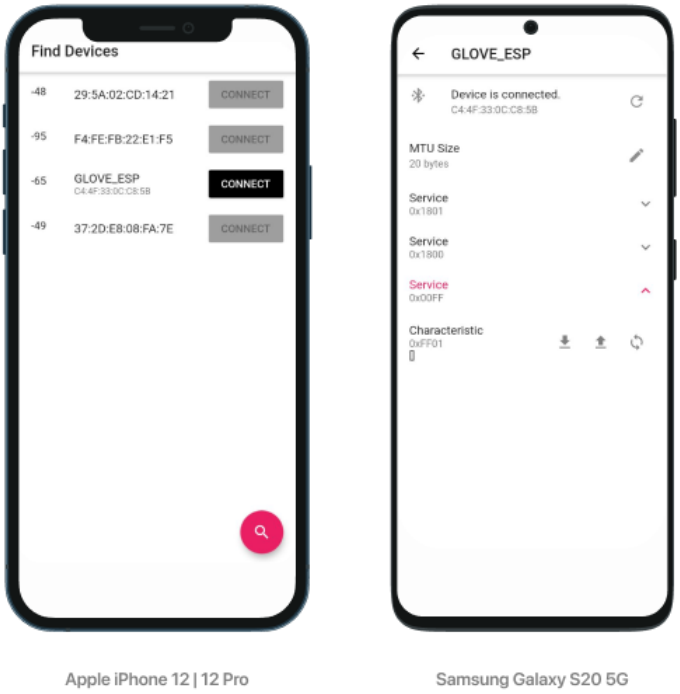


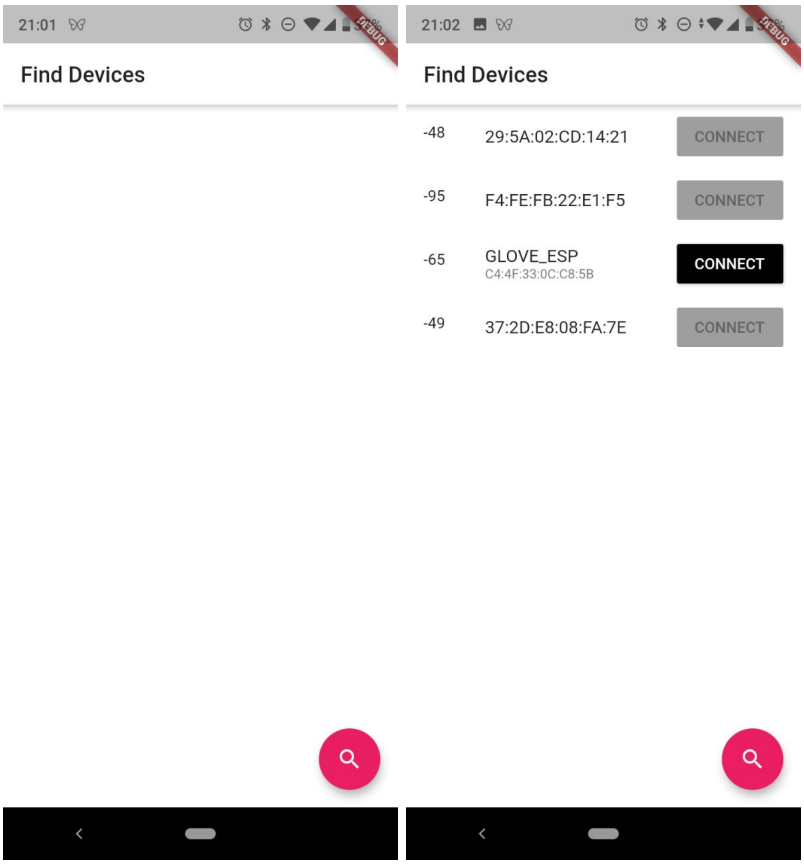
Figura 4.7: App visto em emulador de *Iphone* e *Android*

O *Flutter* torna possível gerar o mesmo projeto em diversas plataformas, conforme descrito no Capítulo 2 deste trabalho. Com poucas modificações é possível simular o software em iOS e Android, como pode ser visto na Figura 4.7.

As telas a seguir, obtidas a partir da simulação da versão para iOS, possuem uma *flag* do *Flutter* de DEBUG no canto superior direito, pois o aplicativo foi testado em versão de desenvolvimento.

1. Tela de descoberta:

   A tela de descoberta, apresentada na Figura 4.8, é responsável pela listagem de dispositivos com status *Bluetooth* como descobríveis.



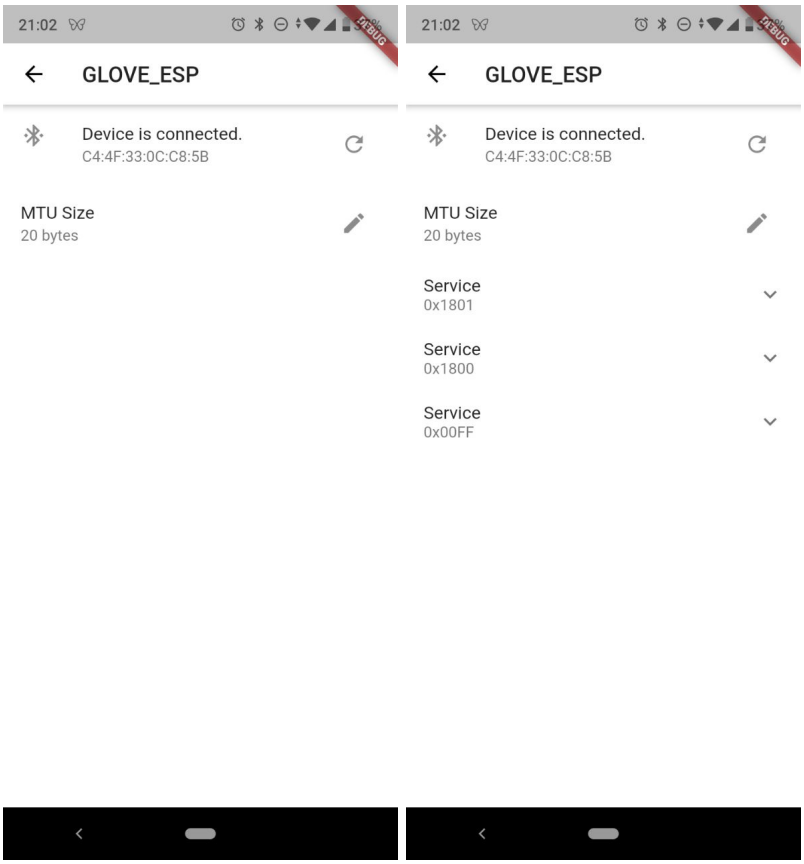
(a) Procurando dispositivos (b) Lista de dispositivos encontrados

Figura 4.8: Tela de Descoberta antes e após achar o ESP32.

A partir dessa lista de dispositivos disponíveis, é possível conectar com o dispositivo desejado, no nosso caso o dispositivo *GLOVE_ESP*.

2. Detalhes da conexão:

   Após a conexão ser estabelecida é possível ver, Figura 4.9, qual o MTU (*Maximum Transmission Unit*) daquele dispositivo, e requisitar um aumento de MTU.



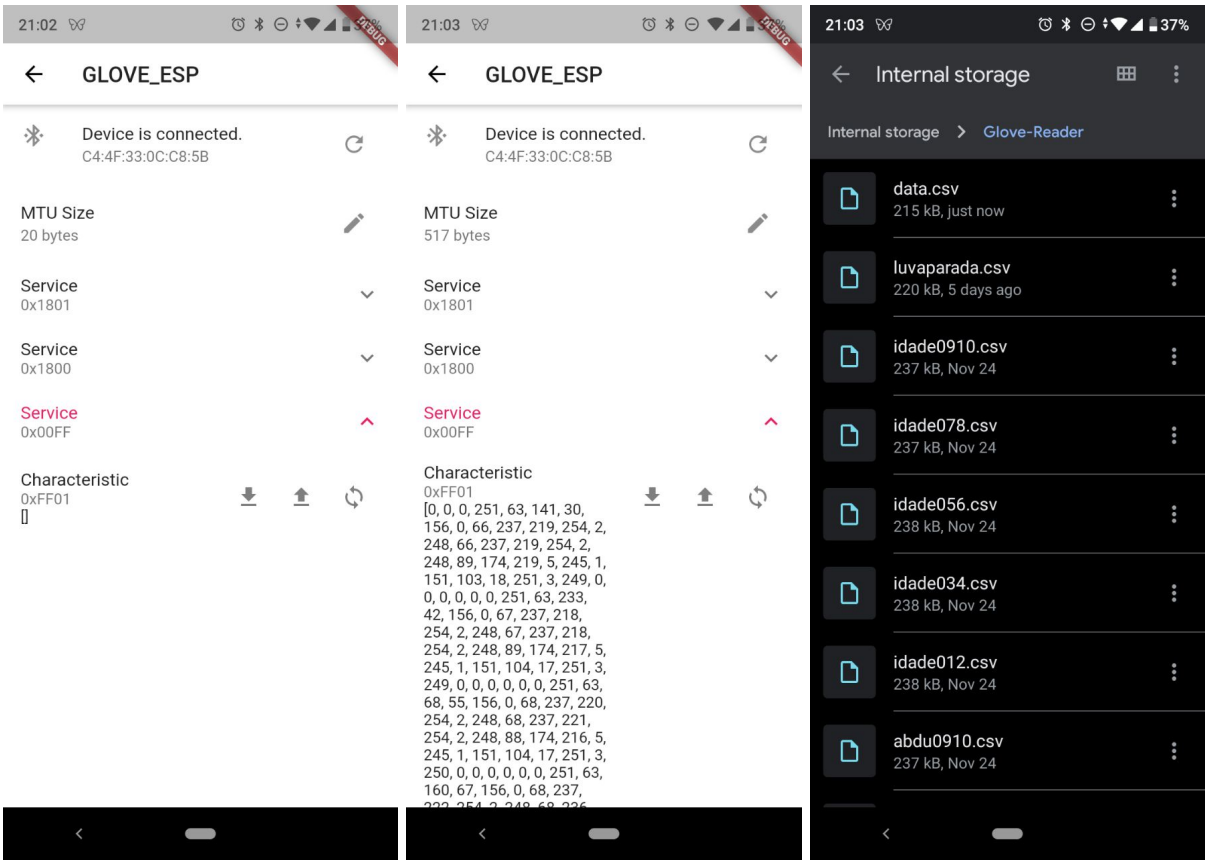
(a) Detalhes da conexão (b) Serviços disponíveis

Figura 4.9: Tela de detalhes de conexão e Serviços GATT da mesma.

Como mostrado na Figura 4.9(b), o dispositivo *GLOVE_ESP* contém 3 serviços, dois do próprio protocolo *BLE* e o último é o serviço utilizado para a troca de dados de leitura dos sensores.

3. Características do Serviço GATT e recebimento de dados:

   A Figura 4.10 ilustra as funcionalidades relacionadas a característica e recebimento de dados. Na primeira tela, Figura 4.10(a), o serviço é selecionado e a única característica existente no mesmo para troca de dados. Na próxima tela, Figura 4.10(b), o usuário a faz a requisição para o LOLIN-D32 PRO clicando na seta direcionada para cima (sentido *Status Bar*) e demonstra o vetor de dados recebidos pela característica. Já na tela mostrada na Figura 4.10(c), podem ser visualizados os arquivos .csv gerados a partir dos dados recebidos pela aplicação mobile.



(a) Listar características (b) Trocar dados (c) Escrever os dados para arquivo

Figura 4.10: Lista de Características de um Serviço GATT e Captura de dados.

É importante mencionar que o próprio protocolo de leitura de dados via BLE que implementamos faz a expansão do tamanho do MTU automaticamente para 517 bytes, que é o valor máximo suportado pelo LOLIN-D32 PRO, conforme pode ser visto na Figura 4.10(b).

Os dados obtidos são então empacotados em um arquivo `data.csv` no armazenamento local do dispositivo móvel na pasta `Glove-Reader` como pode ser visto na Figura 4.10(c). Estes arquivos são a matéria prima da nossa base de dados refinada, os mesmos depois são levados para o computador por qualquer meio de comunicação, como USB, Wi-Fi, *Cloud* etc. Após isso os dados são tratados e decodificados para tornar possível o uso dos mesmos no treinamento, como descrito anteriormente na metodologia.

Vale mencionar que o aplicativo terá dois modos de operação, um para a captura de dados e armazenamento e outro para o uso em tempo real. O aplicativo apresentado está no modo de captura de dados. Os dois modos do aplicativo podem compartilhar muitas ferramentas comuns, como as telas de BLE e o código de serviço de trocas de dados além do gerenciamento de permissões. No segundo modo de operação do aplicativo, é necessário importar a estrutura e os pesos da rede neural já treinada para a interpretação em tempo real dos dados gerados pela luva.

4.5 Consumo de Energia

Para medição do consumo de energia do sistema embarcado na luva, um amperímetro foi posicionado de forma a medir a corrente entre a tensão positiva da bateria com a entrada de bateria do ESP32. Dessa forma em cada estado possível da luva temos a seguir as medições realizadas.

- **Luva parada:** Estado da luva vestida parada sem início da rotina de leitura de sensores - 27,0 mA
- **Leitura de Sensores:** Dois estados da leitura foram medidos:
  - Luva parada lendo sensores - 66,3 mA
  - Luva parada aguardando início da leitura pelo botão pressionado pelo usuário - 72,5mA
- **Luva Luva movimentado lendo sensores:** Leitura de sensores com mudanças bruscas de movimentos - 68,6mA
- **Luva enviando por BLE:** Inicialização do servidor BLE e espera de conexão - 81,3mA

Com esses valores, é possível estimar a autonomia da luva. A bateria possui uma capacidade de carga de 800mAh, e a aplicação requer maior consumo de energia nos estados de leitura e inicialização de servidor BLE. Assumindo que a aplicação passa 40% do tempo em leitura, 20% do tempo em espera e 40% enviando dados, a corrente ponderada de consumo seria

$$40\% \times (66{,}3 + 81{,}3) + 20\% \times (72{,}5) = 73{,}54mA. \tag{4.1}$$

Assim, a autonomia estimada é de $800mAh/73{,}54mA = 10{,}87$ horas de uso contínuo da aplicação.

Dessa forma, é satisfeito o requisito de projeto de autonomia de 30 minutos de protótipo ligado capturando e enviando sinais.

4.6 Banco de Dados de Sinais da LIBRAS

Foram escolhidos 10 sinais que utilizem do parâmetro movimentação de LIBRAS . Todos os sinais podem ser visualizados no vídeo postado no *Youtube* através do seguinte link[1] . Devido à limitação de hardware para uma luva apenas, os sinais da lista à seguir são completamente realizáveis com apenas uma mão.

1. O sinal de “Oi”:

   O sinal de cumprimento “Oi” em LIBRAS é realizado conforme apresentado na Figura 4.11.

[1] `https://youtu.be/PEk5PcdaXjU`

Figura 4.11: Sinal “Oi” - link para o vídeo.

Este sinal se constitui na junção das posturas manuais das letras O e I e movimento da esquerda para a direita, e utiliza todos os parâmetros de Libras, a saber Ponto de Articulação, Movimento, Orientação, Configuração de Mãos e Expressão Facial.

2. O sinal de “Tudo bem?”:

   A pergunta “Tudo bem?” é realizado em LIBRAS conforme mostrado na Figura 4.12.

Figura 4.12: Sinal representando “Tudo Bem?” - link para o vídeo.

Este sinal constitui na junção dos sinais “tudo” e “bem”. Neste caso a expressão facial evidencia ser uma pergunta.

3. O sinal de “Qual o seu nome?”:

   A pergunta “Qual o seu nome?” é realizada em LIBRAS conforme apresentado na Figura 4.13.

Figura 4.13: Sinal para “Qual o seu nome?” - link para o vídeo.

O sinal é realizado pelo movimento da esquerda para a direita de dois dedos seguido do balanço desses dedos três vezes acompanhado pela expressão facial de pergunta.

4. O sinal de “Meu nome é...”:

   A declaração “Meu nome é...” é performada em LIBRAS como apresentado na Figura 4.14.

Figura 4.14: Sinal representando “Meu nome é. . . ” - link para o vídeo.

O sinal “Meu nome é. . . ” é constituído do mesmo movimento de "Qual o seu nome?", mas com a palma da mão voltada para o praticante, indicando que se trata de si. Note também a ausência da expressão facial de pergunta.

5. O sinal “Lucas”:

   Uma das formas utilizadas para responder a pergunta “Qual o seu nome?” é a soletração do nome utilizando o alfabeto manual [61] . No entanto é comum que pessoas se identifiquem com sinais próprios para agilizar a comunicação. Assim, para simplificar as respostas, dois sinais foram criados, “Lucas” e “Abdullah” para representar os nomes dos autores deste trabalho.

Figura 4.15: Sinal representando o nome “Lucas” - link para o vídeo.

O sinal da Figura 4.15 é “Lucas”, realizado apontando-se com o dedo indicador três vezes no canto da boca.

6. O sinal “Abdullah”:

   O sinal para o nome próprio “Abdullah” criado para este trabalho é apresentado na Figura 4.16

Figura 4.16: Sinal “Abdullah” - link para o vídeo .

Este sinal é realizado com um movimento de pinça com os dedos indicador e polegar três vezes próximo ao rosto.

7. O sinal de “Qual sua idade?”:

   A pergunta “Qual sua idade?” é representada em LIBRAS conforme apresentado na Figura 4.17.

Figura 4.17: Sinal “Qual a sua idade?” - link para o vídeo .

   O sinal desta pergunta é performada com a Configuração da Mão ’hangloose’ posicionada próximo ao ombro seguida de três movimentos de baixo para cima em conjunto com a expressão facial de pergunta.

8. O sinal de “23”:

   O sinal do número 23 é incluído no Banco de Sinais deste trabalho com o objetivo de servir de resposta à pergunta “Qual sua idade?”. Este sinal é mostrado na Figura 4.18.

Figura 4.18: Sinal representando o número “23” - link para o vídeo .

   A representação de números em LIBRAS é feita utilizando as posturas da mão de cada número [61]. Neste caso, o número 23 é representado pela postura manual do número 2 seguida da postura do 3.

9. O sinal de “Prazer em te conhecer”:

   A frase “Prazer em te conhecer” é sinalizada conforme mostrado na Figura 4.19.

Figura 4.19: Sinal de “Prazer te conhecer” - link para o vídeo.

Esta frase é composta pelo sinal de “prazer” seguido de “conhecer” finalizando com o sinal de “você”.

10. O sinal de “Tchau”:

    O sinal de “tchau” em LIBRAS é mostrado na Figura 4.20.

Figura 4.20: Sinal representando “Tchau” - link para o vídeo.

O “Tchau” em LIBRAS se utiliza do gesto comum de “tchau”, isto é, o abano com a mão aberta .

11. Silêncio:

    Geralmente em LIBRAS, quando se termina uma frase e esperamos a resposta do interlocutor toma-se a postura de silêncio. Neste trabalho a postura de silêncio é apresentada na Figura 4.21.

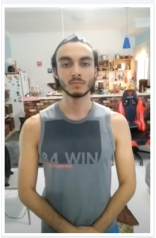


Figura 4.21: Sinal de “Silêncio” - link para o vídeo.

É utilizada a posição de mão neutra em repouso para indicar que nenhum sinal está sendo realizado. Usamos este sinal a fim de facilitar o treinamento das redes neurais indicando de maneira clara quando nenhum sinal está sendo gesticulado.

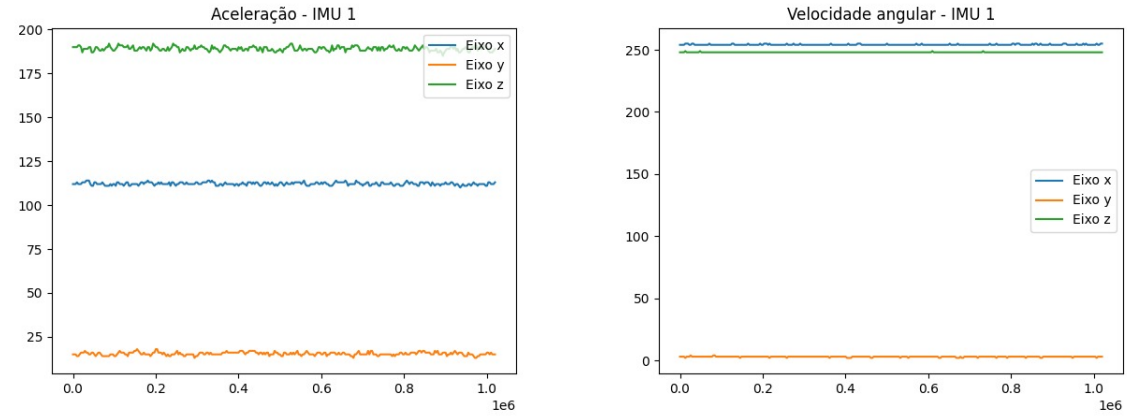
Banco de Dados usado neste trabalho foi construído com 20 repetições para cada um dos 10 sinais escolhidos, além do silêncio, adquiridos a uma frequência de amostragem de 300Hz. Assim, o banco de dados possuiu apenas 220 sequências. Logo, as propostas de redes neurais tiveram que se adaptar à essa limitação. Este Banco de Dados de Sinais encontra-se disponibilizado para fins de pesquisa neste link.

Cada sinal está gravado em um arquivo .csv de acordo com a estrutura especificada anteriormente na Seção 3.10.

4.7 Ruído na leitura de sensores

Analisar o ruído de sensores para a aplicação é fundamental para permitir uma estimativa de quanto os sensores estão com leitura confiável em suas condições de operação comuns. Caso seja identificado que o sensor de algum dedo está com uma variância alta na leitura de ruido, com a luva parada, será um indicador relevante para possível troca ou manutenção do sensor pois o mesmo provavelmente está problemático.

Para analisarmos o ruído dos sensores da luva, deixamos a luva em repouso sem nenhuma interferência externa, enquanto isso coletamos as leituras da IMU do dedo polegar.



(a) Sinal dos acelerômetros com a luva em repouso

(b) Sinal dos giroscópios com a luva em repouso

Figura 4.22: Dados da IMU do dedo 1 (polegar) com a luva em repouso

Foram utilizadas as Equações 4.2 e 4.3 para obter os resultados apresentados nas Tabelas de 4.3 até 4.7

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{4.2}$$

$$\sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu_x)^2 \tag{4.3}$$

Tabela 4.3: Media e variância das acelerações e velocidades angulares no dedo 1 (polegar)

	a_x	a_y	a_z	ω_x	ω_y	ω_z
μ	112.189	15.391	189.116	254.151	2.921	248.012
σ^2	0.618	0.7971	1.467	0.1282	0.0912	0.0124

Tabela 4.4: Media e variância das acelerações e velocidades angulares no dedo 2 (indicador)

	a_x	a_y	a_z	ω_x	ω_y	ω_z
μ	112.220	15.425	189.066	254.145	2.917	248.019
σ^2	0.636	0.740	1.464	0.1236	0.0813	0.0185

Tabela 4.5: Media e variância das acelerações e velocidades angulares no dedo 3 (médio)

	a_x	a_y	a_z	ω_x	ω_y	ω_z
μ	118.914	172.302	231.864	5.031	244.305	53.902
σ^2	0.7757	0.6821	1.280	0.0556	0.2242	1.930

Tabela 4.6: Media e variância das acelerações e velocidades angulares no dedo 4 (anelar)

	a_x	a_y	a_z	ω_x	ω_y	ω_z
μ	157.261	115.523	14.899	252.422	1.776	249.296
σ^2	0.7901	0.6574	1.172	0.2432	0.1858	0.2079

Tabela 4.7: Media e variância das acelerações e velocidades angulares no dedo 5 (mínimo)

	a_x	a_y	a_z	ω_x	ω_y	ω_z
μ	100.671	161.876	243.022	253.936	246.022	90.899
σ^2	0.7920	0.6547	1.3108	0.0589	0.0466	0.7131

É possível observar que as variâncias dos sinais estão próximas de 1, o que indica um comportamento pouco ruidoso das medições do sistema. Dos 5 sensores visualizados, o sensor de velocidade angular no eixo z do dedo 3 da Tabela 4.5 foi o mais ruidoso.

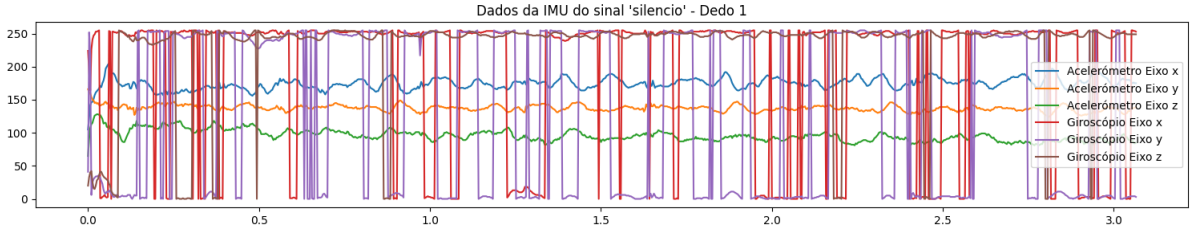
No entanto, é possível afirmar que mesmo após a decisão de armazenar os valores de aceleração e velocidade angular em variáveis de 8 bits, os dados ainda são representativos.

4.8 Gráficos obtidos para sinais

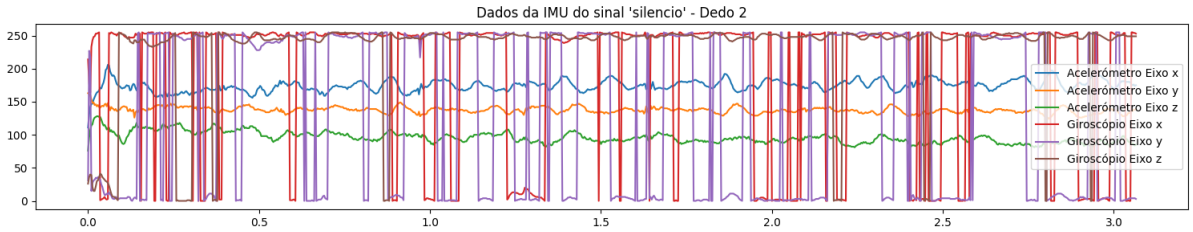
De forma a visualizar como cada sinal era representado nas 5 IMUs, dois movimentos foram escolhidos, o representativo de Silêncio, e o sinal de Oi em LIBRAS. Dessa forma, é perceptível visualmente a diferença dos dois quando comparado o mesmo dedo, ou mesma IMU.

4.8.1 Sinal “Silêncio”

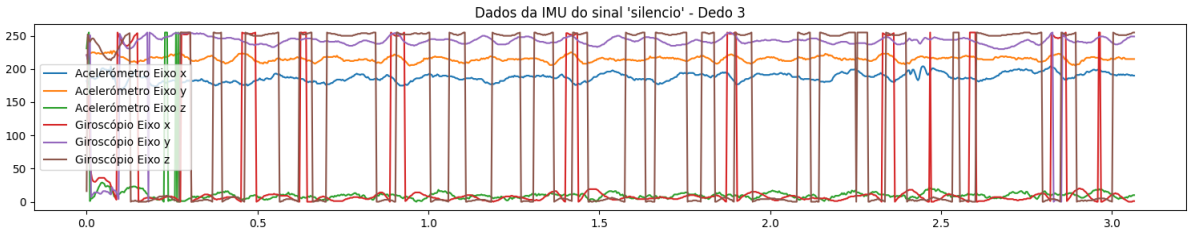
A seguir temos o pseudos-sinal considerado como “Silêncio”.



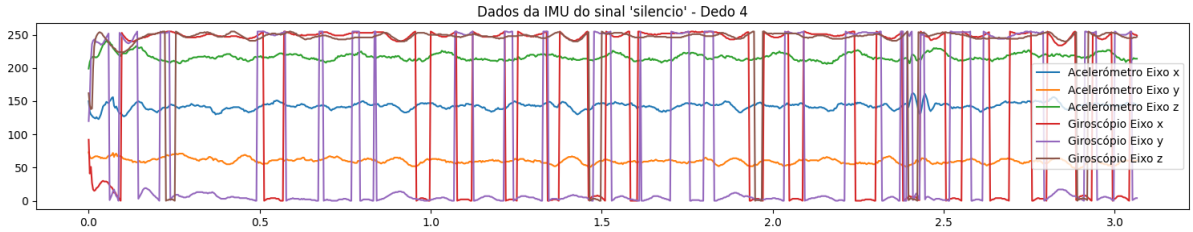
(a) Dados dos sensores do dedo 1 do Silêncio



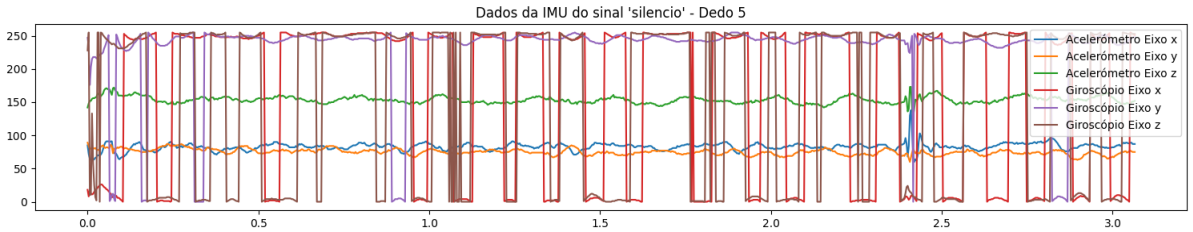
(b) Dados dos sensores do dedo 2 do Silêncio



(c) Dados dos sensores do dedo 3 do Silêncio



(d) Dados dos sensores do dedo 4 do Silêncio

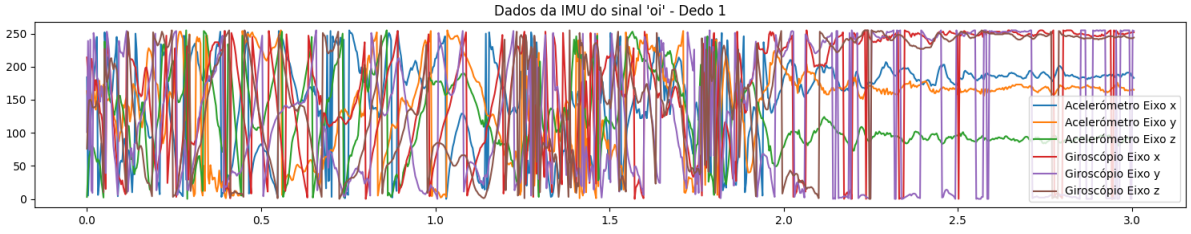


(e) Dados dos sensores do dedo 5 do Silêncio

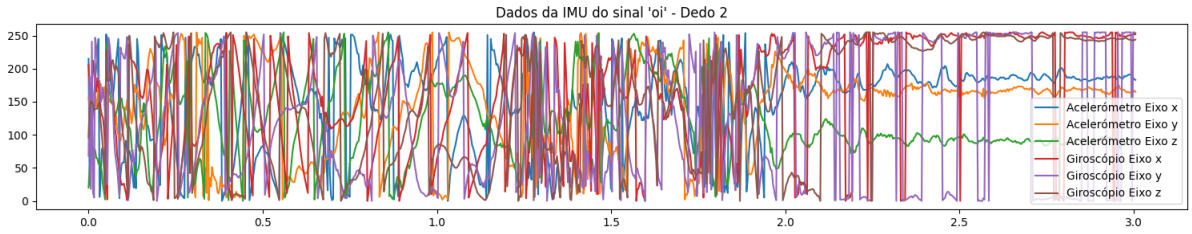
Figura 4.23: Representação do sinal “Silêncio”

4.8.2 Sinal “Oi”

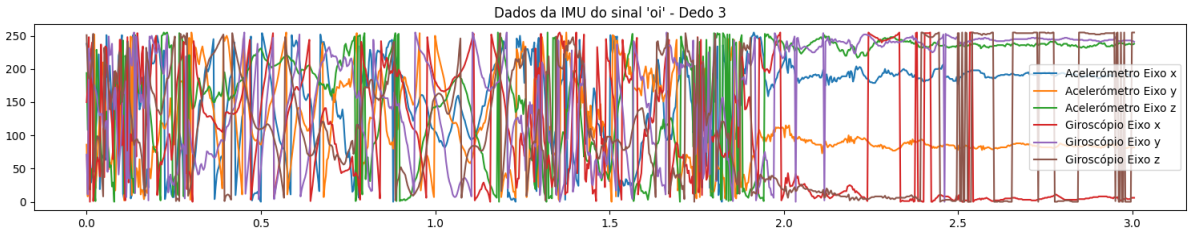
Sinal “Oi” representado nas 5 IMUS com máxima frequência de amostragem de 300Hz.



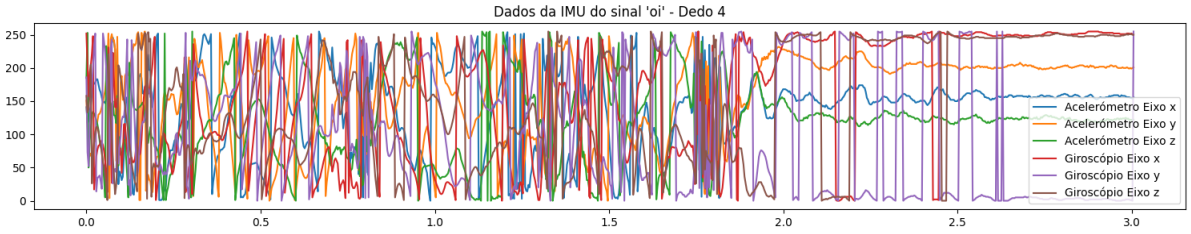
(a) Dados dos sensores do dedo 1 do sinal Oi



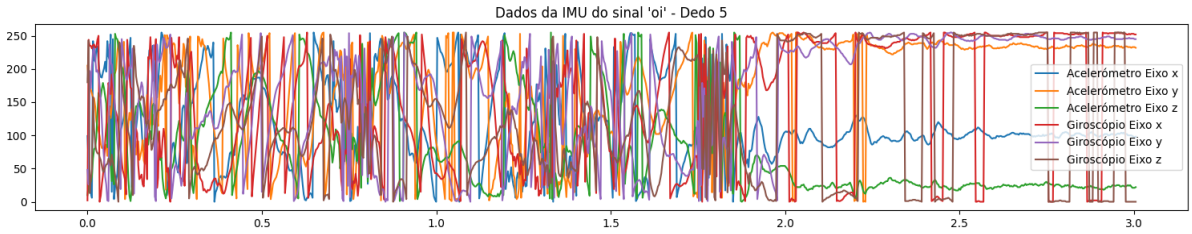
(b) Dados dos sensores do dedo 2 do sinal Oi



(c) Dados dos sensores do dedo 3 do sinal Oi



(d) Dados dos sensores do dedo 4 do sinal Oi

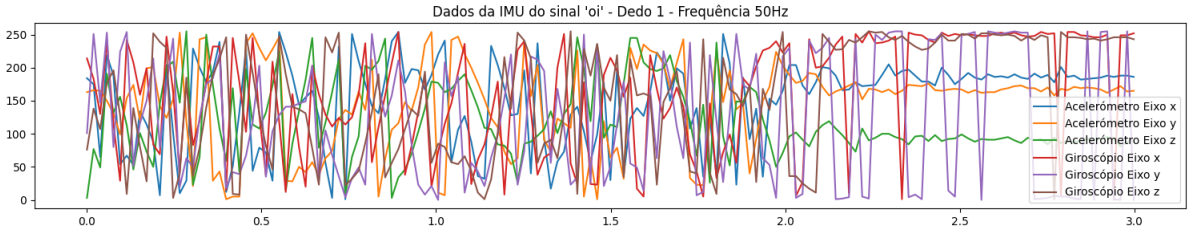


(e) Dados dos sensores do dedo 5 do sinal Oi

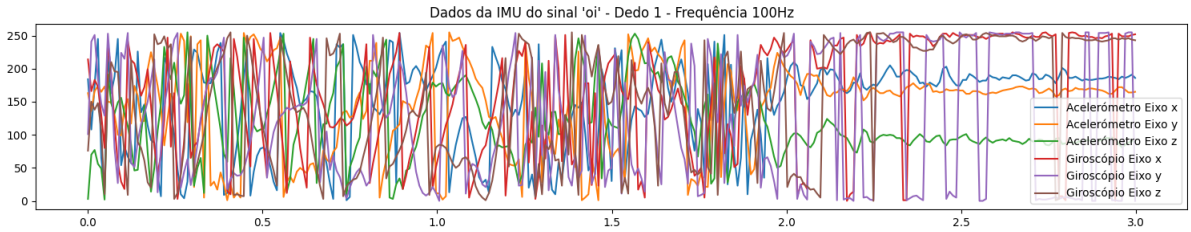
Figura 4.24: Sinal “Oi”

4.8.3 Sinal “Oi” para outras frequências

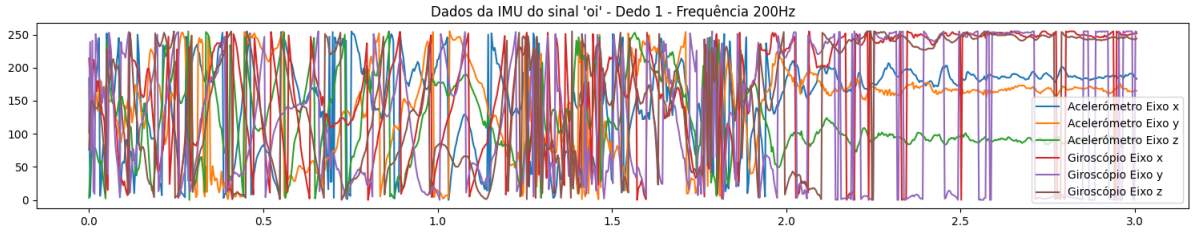
De forma a perceber a diferença entre os dados de frequências de amostragem inferiores, os gráficos do dedo 1 durante o sinal de “Oi” foi gerado para 3 diferentes frequências: 50Hz, 100Hz e 200Hz.



(a) Na frequência 50Hz



(b) Na frequência 100Hz



(c) Na frequência 200Hz

Figura 4.25: Leituras dos sensores para o sinal oi em diferentes frequências

Dessa forma é perceptível nas figuras 4.25(a), 4.25(b) e 4.25(c) que o número de picos dos sinais e representação dos gráficos aumenta proporcionalmente com a frequência de amostragem.

4.9 Resultados da Rede

As redes neurais deste trabalho foram treinadas com o algoritmo de*Backpropagation*. Parâmetros utilizados nos treinamentos de todos os resultados desta seção encontram-se na Tabela 4.8.

Tabela 4.8: Parâmetros de treinamento

Parâmetro de treinamento	**Valor**
Algoritmo de otimização	Adam
K de K-Fold	10
Número máximo de épocas	100
Condição de parada	Δ acurácia de validação $<$ 0,001 por 3 épocas
Porcentagem de validação	30%
Tamanho do lote (*Batch Size*)	32

Com o objetivo de medir o desempenho das nossas redes, calculamos a precisão, *recall*, *F1-score* e a acurácia de acordo com

$$Precisão = \frac{Tp}{Tp + Fp}, \tag{4.4}$$

$$Recall = \frac{Tp}{Tp + Fn}, \tag{4.5}$$

$$F1\_Score = 2\frac{(precisão \cdot recall)}{(precisão + recall)} = \frac{Tp}{Tp + \frac{1}{2} \cdot (Fp + Fn)} \tag{4.6}$$

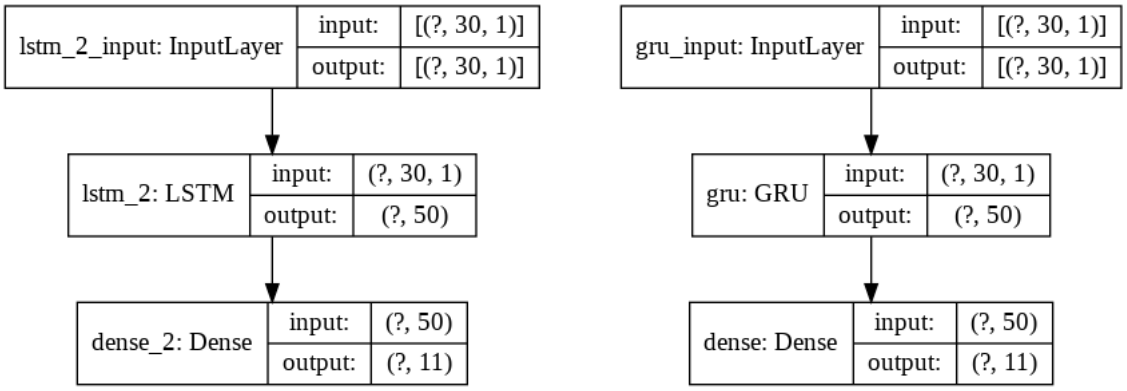
e

$$Acurácia = \frac{Tp + Tn}{Tp + Tn + Fp + Fn} \tag{4.7}$$

onde Tp é o número de verdadeiros positivos encontrados pela rede neural, Tn é o número de verdadeiros negativos, Fp são os falsos positivos e Fn os falsos negativos.

4.9.1 Estrutura da Rede

As estruturas utilizadas foram LSTM e GRU por ter como propósito resolverem problemas de origem temporal e sequencial. A rede GRU é proposta como uma alternativa à LSTM para casos com bancos de dados menores. Nos propusemos a testar se para nossa aplicação específica esta característica iria se apresentar. Assim, seguem as estruturas das redes neurais utilizadas nas Figuras 4.26 e 4.27.

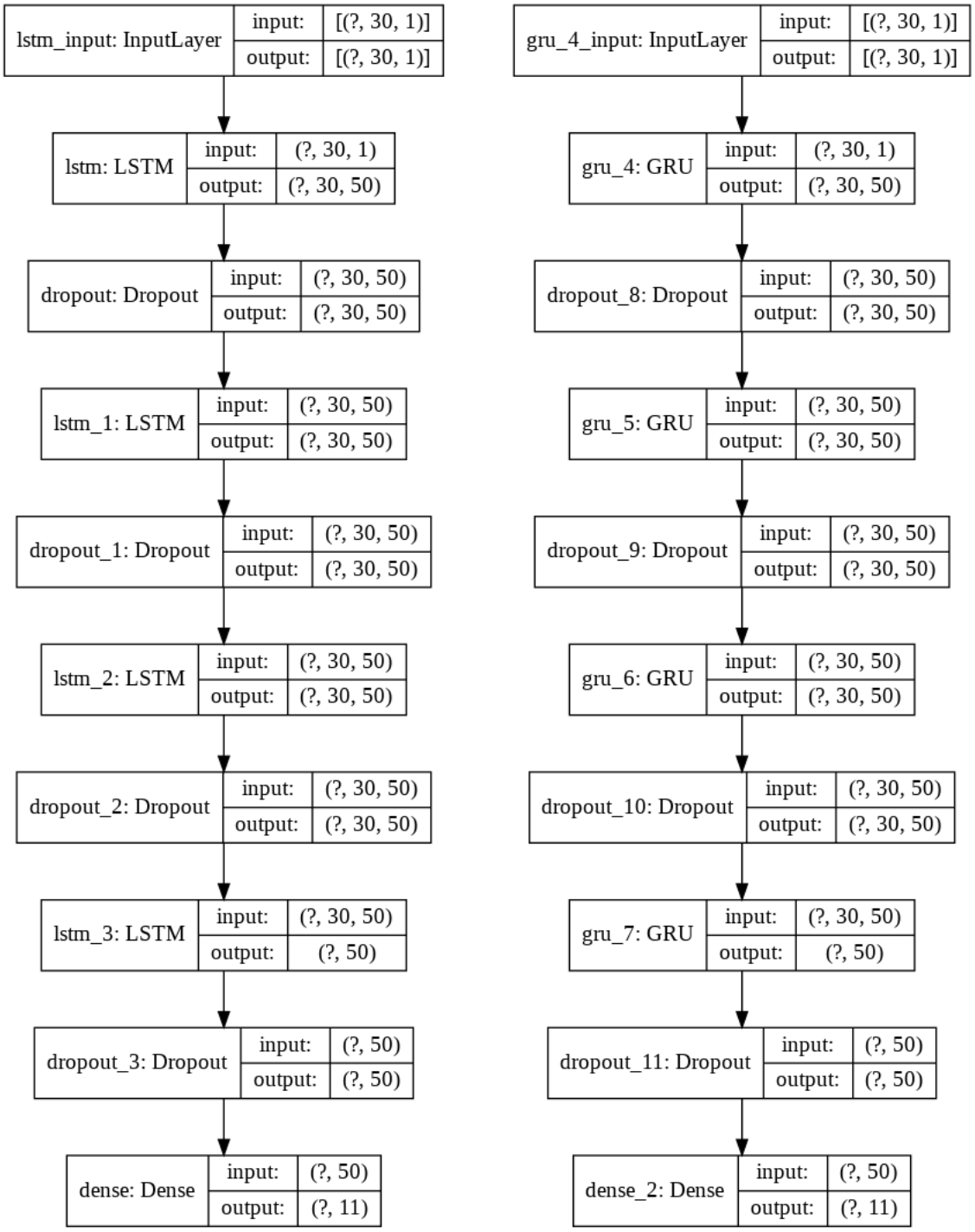


(a) LSTM Rasa

(b) GRU Rasa

Figura 4.26: Estruturas LSTM rasa e GRU rasa

A Figura 4.26 apresenta as estruturas das redes neurais rasas LSTM e GRU utilizadas. Ambas as redes são compostas de 30 nós de entrada, 50 nós na camada escondida recorrente e 11 nós de saída.



(a) LSTM Profunda

(b) GRU Profunda

Figura 4.27: Estruturas LSTM profunda e GRU profunda

A Figura 4.27 apresenta as estruturas das redes neurais profundas LSTM e GRU utilizadas. Ambas as redes são compostas de 30 nós de entrada, seguidas de 4 sequências de camadas LSTM ou GRU seguida de uma camada de *Dropout*, terminando com a camada de saída de 11. As camadas *Dropout* usam um fator de descarte de 20%.

4.9.2 Análise de Desempenho

De forma a perceber a diferença de performance para cada rede, foram calculados a precisão, *recall* e *F1-Score*. As Tabelas 4.9 a 4.14 apresentam os resultados obtidos por sinal, para cada tipo de rede com a frequência de amostragem variando de 300Hz, 150Hz e 50Hz.

Tabela 4.9: Desempenho por sinal - rede LSTM profunda, frequência 300Hz

Sinal	**Precisão**	**Recall**	**F1-score**
Idade é 23	62%	79%	70%
Lucas	81%	54%	65%
Oi	62%	67%	65%
Silêncio	100%	100%	100%
Tchau	34%	41%	37%
Tudo Bem	65%	73%	69%
Qual a sua idade?	75%	64%	69%
Prazer em te conhecer	51%	49%	50%
Meu nome é	57%	47%	52%
Abdullah	58%	62%	60%
Qual é seu nome ?	62%	58%	60%

Tabela 4.10: Desempenho por sinal - rede GRU profunda, frequência 300Hz

Sinal	**Precisão**	**Recall**	**F1-score**
Idade é 23	58%	76%	66%
Lucas	78%	52%	63%
Oi	64%	65%	64%
Silêncio	100%	100%	100%
Tchau	38%	37%	38%
Tudo Bem	57%	76%	65%
Qual a sua idade?	78%	64%	71%
Prazer em te conhecer	52%	46%	49%
Meu nome é	50%	46%	48%
Abdullah	66%	55%	60%
Qual é seu nome ?	49%	60%	54%

Tabela 4.11: Desempenho por sinal - rede GRU rasa, frequência 300Hz

Sinal	Precisão	Recall	F1-score
Idade é 23	54%	39%	45%
Lucas	76%	52%	62%
Oi	64%	49%	56%
Silêncio	100%	100%	100%
Tchau	22%	32%	26%
Tudo Bem	59%	50%	54%
Qual a sua idade?	90%	48%	63%
Prazer em te conhecer	74%	37%	49%
Meu nome é	20%	60%	30%
Abdullah	89%	42%	57%
Qual é seu nome ?	40%	41%	41%

Tabela 4.12: Desempenho por sinal - rede LSTM rasa, frequência 300Hz

Sinal	Precisão	Recall	F1-score
Idade é 23	86%	47%	61%
Lucas	44%	51%	47%
Oi	35%	42%	38%
Silêncio	99%	99%	99%
Tchau	72%	55%	63%
Tudo Bem	34%	38%	36%
Qual a sua idade?	27%	45%	33%
Prazer em te conhecer	19%	21%	20%
Meu nome é	70%	42%	52%
Abdullah	54%	49%	52%
Qual é seu nome ?	40%	39%	40%

Tabela 4.13: Desempenho por sinal - rede LSTM profundas, frequência 150Hz

Sinal	Precisão	Recall	F1-score
Idade é 23	19%	33%	24%
Lucas	44%	56%	49%
Oi	90%	48%	63%
Silêncio	56%	57%	57%
Tchau	72%	56%	63%
Tudo Bem	62%	38%	47%
Qual a sua idade?	53%	45%	49%
Prazer em te conhecer	35%	46%	40%
Meu nome é	100%	100%	100%
Abdullah	85%	44%	58%
Qual é seu nome ?	33%	43%	37%

Tabela 4.14: Desempenho por sinal - rede LSTM profundas, frequência 50Hz

Sinal	Precisão	Recall	F1-score
Idade é 23	79%	48%	60%
Lucas	68%	49%	57%
Oi	41%	43%	42%
Silêncio	100%	100%	100%
Tchau	33%	53%	40%
Tudo Bem	39%	43%	41%
Qual a sua idade?	29%	30%	30%
Prazer em te conhecer	69%	47%	56%
Meu nome é	52%	59%	55%
Abdullah	90%	53%	67%
Qual é seu nome ?	31%	44%	36%

Percebe-se que o sinal que as redes mais tiveram sucesso em aprender foi o sinal de “silêncio”. Este sinal apresentou o resultado mais alto de precisão na maioria das redes treinadas e testadas. Este resultado é esperado dada a baixa complexidade do comportamento dos sensores durante a execução desse pseudo-sinal.

A Tabela 4.15 apresenta as acurácias médias ao longo do K-Fold, com os sinais amostrados a 300Hz, para os quatro tipos de redes estudados.

Tabela 4.15: Acurácias médias para cada tipo de rede

Tipo de Rede e frequência	**Acurácia**
LSTM Rasa 300Hz	46,4%
GRU Rasa 300Hz	49,9%
GRU Profunda 300Hz	61,6%
LSTM Profunda 300Hz	61,8%

Vale ressaltar que para redes simples, a rede GRU desempenhou melhor a tarefa com um *dataset* pequeno como visto na Tabela 4.15. Notou-se ainda que as redes profundas obtiveram resultados mais satisfatórios do que as redes rasas, e entre as redes GRU e LSTM profundas a diferença foi de 0.2%. Então, para o restante das análises, impacto da frequência e matriz de confusão, usou-se a rede LSTM profunda.

A Tabela 4.16 apresenta as acurácias médias ao longo do K-Fold, para as redes LSTM profundas com a variação da frequência de amostragem.

Tabela 4.16: Acurácias médias para cada frequência de amostragem

Tipo de Rede e frequência	**Acurácia**
LSTM Profunda 300Hz	61,8%
LSTM Profunda 150Hz	50,4%
LSTM Profunda 50Hz	47,8%

Pode ser observado na Tabela 4.16, que quanto maior for a frequência de amostragem dos dados de entrada para um determinado sinal, maior é a acurácia da rede.

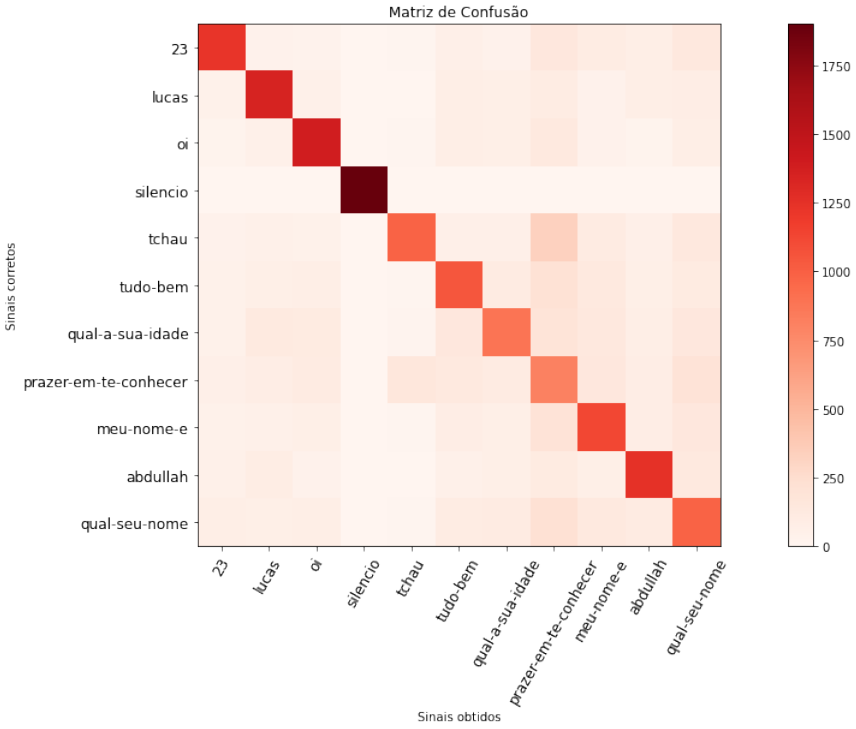


Figura 4.28: Matriz de confusão nos dados de teste para LSTM profunda e frequência de 300Hz

Por fim, é visível pela Matriz de Confusão na Figura 4.28 que há poucas regiões de falsos positivos ou falsos negativos entre os diferentes tipos de sinais. Os dois sinais que possuem a maior dependência e taxa de erro são os sinais “prazer em te conhecer” e “tchau”, isto é justificável pois os dois sinais possuem movimentos e configurações de mãos parecidos.

4.10 Custo do Projeto

Com o objetivo de verificar o custo do equipamento desenvolvido é necessário relatar o custo de cada componente. Projetar um equipamento de baixo custo é um dos requisitos deste trabalho.

Tabela 4.17: Custo do Projeto

Item	Quantidade	Preço Unitário(R$)	Preço Total (R$)
Par de luvas de construção com revestimento de borracha fino	1,0	20,00	20,00
Módulo IMU MPU-6050	5,0	16,54	82,70
Módulo LOLIN-D32PRO	1,0	52,90	52,90
Baterias LiPo LW902540	5,0	13,90	69,50
Carregador de Baterias LiPo 1 Célula	1,0	Gratuito, com as baterias	0,00
Valor total	-	-	169,50

Assim, o preço total de materiais para a fabricação da luva de R$169,50. O custo de desenvolvimento do projeto, inclui ainda o software desenvolvido e as horas de trabalho desenvolvendo o protótipo. Caso fossem inclusos, poderia-se estimar o valor de software desenvolvido de R$200 e o custo de horas trabalhadas de dois engenheiros por 2 anos de pesquisa e desenvolvimento equivalentes à cerca de R$8.000 por mês por 18 meses de trabalho, totalizando R$144.000 reais de custo estimado de desenvolvimento deste projeto.

Um preço de venda de um protótipo capaz de capturar movimentos de cinemática humana sugerido seria de R$500, um valor competitivo quando comparado com outros produtos similares no mercado.

- HI5 VR GLOVE BUSINESS EDITION [62]: Durante a elaboração deste documento, esse modelo de luva, utilizado por desenvolvedores de jogos em Realidade Virtual, custava $999 USD. Características:

  1. Sensor 9-DOF IMU para cada luva (Histerese Ultra-baixa); 7 IMUs por luva.
  2. Acurácia: Roll/pitch (dinâmica): 1,0° (RMS); Acurácia de Cabeçalhos: 2,0° (RMS); Resolução: 0,1.
  3. Vida de Bateria: >3 horas contínuas de trabalho em carga cheia charge (bateria 1AA 2100mAh); Alerta de bateria baixa.
  4. Tensão de alimentação: 1,1-1,5 VDC com uma AA bateria para cada Luva; 5,0 VDC para o *dongle* USB 2.0.
  5. Latência <5ms (de standby para movimento , dentro de condições RF limpas).
  6. Frequência de saída de dados: 180Hz.

7. Comunicação: Wireless (entre luva e receptor); 2,4GHz rádio frequência (protocolo privado).
8. Performance RF: Área de trabalho RF: 5m×5m; Suporte à multi usuários (até 6) no mesmo campo aberto (troca de canal automática para evitar interfaceamento RF).
9. Vibração: 1 vibrador programável em cada pulso da luva.
10. Material têxtil: Antibactericida, elástico têxtil respirável.
11. Peso: 105g por Luva.
12. Compatibilidade com Ambientes de Desenvolvimento: Unity SDK e Unreal SDK.

- Luvas da NANSENSE [63]: Com preço inicial de $2.399 USD por mão, as luvas possuem características diferentes que dependem da forma que será utilizada. Seguem as características comuns:

1. Transferência de dados: Latência: 30ms (Wireless) | 10ms (Cabeado)
   Protocolo: 2,4GHz | 5GHz | Wired
   Alcance Wireless: 45m (Dentro de casa) | 90m (Campo aberto)
   Frequência interna de Hardware: 1000Hz
   Frequência de software: 240fps

2. Especificação de Sensores: Acurácia Estacionária: 0° (Sensor Standby)
   Acurácia Movimento Lento: 0.5° (Roll/Pitch) | 1,0° (Yaw)
   Acurácia Movimento Rápido: 0,7° (Roll/Pitch) | 1,4° (Yaw)
   Giroscópio: ±2000°/s
   Acelerômetro: ±16g
   Alcance Magnético: ±1250µT (Roll/Pitch) | ±2500µT (Yaw)
   Resolução Magnética: 0,25µT
   Consumo de Energia:: 16mAh (R2) | 10mAh (R2m)
   Dimensões: 12×30×5mm (R2) | 11×25×5mm (R2m)

3. Hub de Processamento: Processador: Quad-Core 64bit ARM
   Número máximo de Sensores: ×60 (Total)
   Logger: 128GB
   Portas: ×2 USB-A
   Requisito de Tensão: 5V/2A
   Gerador de Timecode: Buit-In | LTC Sync Compatible
   Dimensões: 100×75×25mm
   Peso: 210g

4. Bateria: Output: 5V / 2A
   Capacidade: 13000mAh
   Suite de Vida de Bateria: 14h (Projeto Indie) | 12h (Projeto Pro) | 10h (Projeto

BioMed)
Vida da Bateria Bundle: 8h (Projeto Indie) | 7h (Projeto Pro) | 6h (Projeto BioMed)
Peso: 255g

Assim, o projeto está em um valor aceitável para um produto de captura de dados como este. Além de que todos os materiais utilizados na fabricação do produto são facilmente acessíveis, podendo ser encontrados em lojas de eletrônica e varejistas.

Capítulo 5

Conclusões

Este trabalho propôs o desenvolvimento de um sistema que consiste de uma luva portátil com bateria e sensores embarcados e um Aplicativo de *SmartPhone* multiplataforma (iOS e *Android*) para a coleta dos dados e classificação de sinais em LIBRAS.

Este sistema busca traduzir sinais em movimento usando redes neurais recorrentes, tornando possível a tradução de sinais em LIBRAS. A maioria dos sinais utilizados no dia dia utilizam parâmetro de movimento para representar uma ideia completa, seja uma frase ou palavra inteira.

Além disso, o sistema é agnóstico ao meio de comunicação, pois poderia ser utilizado com BLE, *Bluetooth* Clássico, Wi-Fi, USB Serial, etc. A transmissão pode ser implementada da forma mais conveniente e de acordo com a necessidade.

O sistema proposto mostrou resultados de qualidade média, assim podendo evoluir e ter uma acurácia melhor com o aumento de volume de dados, pois como foi mencionado anteriormente, o volume de dados que foi utilizado era limitado e pouco diverso, por ter sido de uma única pessoa e somente 20 repetições de cada sinal.

Um fator crucial que afetou este trabalho foi a pandemia. Devido ao impedimento de acesso aos laboratórios e equipamentos da Universidade e de capturar dados de sinais mais confiáveis de usuários surdos, houve um custo maior de equipamentos e dados pouco diversificados. Além disso, um dos alunos participantes do projeto ficou impossibilitado de retornar ao Brasil durante meses desde março, limitando seu acesso ao protótipo físico, o que implicou em uma divisão de tarefas de Software e Hardware mais rígidas entre os dois alunos.

5.1 Trabalhos Futuros

Como sugestão de continuação e melhoria deste trabalho propomos.

- **Cabeamento**: Utilização de cabos com entradas padronizadas para os sensores (como Micro-USB, com 5 pinos);
- **Placa de Circuito Impresso**: Desenvolvimento de placa de circuito integrado para facilitar o roteamento de fios e diminuir desgaste por flexão dos mesmos;
- **Aprimoramento de Serviços de Nuvem**: Utilização de arquitetura de projeto que permita utilização de treinamento em nuvem e serviço de requisição dos dados de nuvem;
- **Banco de Dados**: Expansão do banco com dados de múltiplos usuários, mais sinais e mais amostras de cada sinal;
- **Redes Neurais**: Experimentar outras estruturas de rede neural com o intuito de obter acurácias mais altas;
- **Aplicativo de Tradução em Tempo Real**: Exportar os pesos da rede neural e importar eles no aplicativo de *Smartphone* já feito;
- **Duas Luvas**: Integração de outra luva no sistema, de forma a capturar sinais que necessitem de ambas as mãos;
- **Pré Processamento**: Estudo de inclusão de fases de pré-processamento para criar dados em quatérnions e ângulos Euler de forma a perceber mudança na performance das redes;
- **Visão Computacional**: Utilização de sensores óticos para englobar sinais que requerem face e expressões corporais;

REFERÊNCIAS BIBLIOGRÁFICAS

[1] CAPOVILLA, F. C.; RAPHAEL, W. D. *Dicionário enciclopédico ilustrado trilíngue da língua de sinais brasileira. v.1/2.* [S.l.]: Edusp, 2001.

[2] OREILLY, R.; KHENKIN, A.; HARNEY, K. Sonic nirvana: Using mems accelerometers as acoustic pickups in musical instruments. v. 26, 03 2009. Disponível em: <https://www.researchgate.net/publication/228412951_Sonic_nirvana_Using_mems_accelerometers_as_acoustic_pickups_in_musical_instruments>.

[3] WATSON, J. Mems gyroscope provides precision inertial sensing in harsh, high temperature environments. ANALOG DEVICES, 12 2020. Disponível em: <https://www.analog.com/media/en/technical-documentation/tech-articles/MEMS-Gyroscope-Provides-Precision-Inertial-Sensing-in-Harsh-High-Temps.pdf>.

[4] CAMPBELL, S. Basics of the i2c communication protocol. Circuit Basics, Fevereiro 2016. Disponível em: <https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/>.

[5] RATHOR, S. Simple rnn vs gru vs lstm :- difference lies in more flexible control. 2018. Disponível em: <https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57>.

[6] LEE, B. G.; CHONG, T.-W.; CHUNG, W.-Y. Sensor fusion of motion-based sign language interpretation with deep learning. Sensors, 2020.

[7] CENSO de 2010 de Deficientes Auditivos em Território nacional. IBGE, 2010. Disponível em: <https://www.ibge.gov.br/apps/snig/v1/?loc=0&cat=-1,-2,-3,128&ind=4643>.

[8] SACKS, O. *Vendo Vozes: Uma viagem ao mundo dos surdos.* São Paulo: Companhia de Bolso, 2010.

[9] LIST of Sign Languages. Wikipedia, 2020. Disponível em: <https://en.wikipedia.org/wiki/List_of_sign_languages>.

[10] GESSER, A. "libras? que língua é essa?": Crenças e preconceitos em torno da língua de sinais e da realidade surda. São Paulo: Parábola, 2009.

[11] FELIPE, T. A. *Libras em Contexto - Curso Básico.* [S.l.]: WalPrint Gráfica e Editora, 2007.

[12] CHIP Hall of Fame: Microchip Technology PIC 16C84 Microcontroller. IEEE SPECTRUM, June 2017. Disponível em: <https://spectrum.ieee.org/tech-history/silicon-revolution/chip-hall-of-fame-microchip-technology-pic-16c84-microcontroller>.

[13] ESPRESSIF. *ESP32 Main Page*. Espressif Systems, Novembro 2020. Disponível em: <http://esp32.net/>.

[14] COLEY, B. et al. *Stair climbing detection during daily physical activity using a miniature gyroscope*. [S.l.]: Gait Posture, 2005. 287-294 p.

[15] SANA, U. et al. A comprehensive survey of wireless body area networks. J. Med. Syst., v. 36, p. 1065–1094, 2012.

[16] COOPER, G. et al. Inertial sensor-based knee flexion/extension angle estimation. J. Biomech, p. 2678–1685, 2009.

[17] FONG, D. T.-P.; CHAN, Y.-Y. The use of wearable inertial motion sensors in human lower limb biomechanics studies: A systematic review. Sensors, 2010.

[18] A., B. et al. Evaluation of mems capacitive accelerometers. IEEE Design & Test of Computers, v. 16 (4), p. 48–56, Dezembro 1999. Disponível em: <https://ieeexplore.ieee.org/document/808209>.

[19] BLUETOOTH. *Bluetooth Origin*. Bluetooth, Novembro 2020. Disponível em: <https://www.bluetooth.com/about-us/bluetooth-origin/>.

[20] BLUETOOTH. *Bluetooth Core Specification Version 5.2 Feature Overview*. Bluetooth, January 2020. Disponível em: <https://www.bluetooth.com/wp-content/uploads/2020/01/Bluetooth_5.2_Feature_Overview.pdf>.

[21] REYNOLDS., M. *Wibree becomes ULP Bluetooth*. ElectronicsWeekly.com, Junho 2007. Disponível em: <https://web.archive.org/web/20080907170808/http://www.electronicsweekly.com/Articles/2007/06/12/41582/wibree-becomes-ulp-bluetooth.htm>.

[22] WIBREE forum merges with Bluetooth SIG. Nokia, Junho 2007. Disponível em: <https://web.archive.org/web/20070616082658/http://www.wibree.com/press/Wibree_pressrelease_final_1206.pdf>.

[23] AFANEH, M. *Intro to BLUETOOTH LOW ENERGY*. [S.l.]: NovelBits, 2018.

[24] MURRAY, D. et al. Large mtus and internet performance. *IEEE 13th International Conference on High Performance Switching and Routing*, 2012. Disponível em: <https://researchrepository.murdoch.edu.au/id/eprint/9920/>.

[25] MARTIN, R. C. *Design Principles and Design Patterns*. 2000.

[26] ESPRESSIF - Read the Docs Template Documentation - Release v4.0.1. Espressif, Maio 2020.

[27] FLUTTER architectural overview. Flutter, Dezembro 2020. Disponível em: <https://flutter.dev/docs/resources/architectural-overview>.

[28] DART. *Dart Programming Language Documentation*. Dart, 2020. Disponível em: <https://dart.dev/guides>.

[29] MURAD, A.; PYUN, J.-Y. Deep recurrent neural networks for human activity recognition. *Sensors*, v. 17, p. 2556, 11 2017.

[30] ZHOU, V. An introduction to recurrent neural networks for beginners. 2019. Disponível em: <https://victorzhou.com/blog/intro-to-rnns/>.

[31] PASCANU, R.; MIKOLOV, T.; BENGIO, Y. On the difficulty of training recurrent neural networks. *30th International Conference on Machine Learning, ICML 2013*, 11 2012.

[32] WEBER, N. Why lstms stop your gradients from vanishing: A view from the backwards pass. 11 2017. Disponível em: <https://weberna.github.io/blog/2017/11/15/LSTM-Vanishing-Gradients.html#fn:3>.

[33] CHUNG, J. et al. Empirical evaluation of gated recurrent neuronal networks on sequence modeling, neuronal and evolutionary computing. 2014.

[34] J., C. et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. 12 2014. Disponível em: <https://arxiv.org/pdf/1412.3555v1.pdf>.

[35] J., A. et al. Data gathering for gesture recognition systems based on single color-, stereo color- and thermal cameras. *Int. J. Signal Process. Image Process. Pattern Recognit.*, v. 3, 2010.

[36] KOPUKLU, O. et al. Real-time hand gesture detection and classification using convolutional neural networks. *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, IEEE, May 2019. Disponível em: <http://dx.doi.org/10.1109/FG.2019.8756576>.

[37] LANGFORD, G. B. *Flexible potentiometer*. Google Patents, 1990. Disponível em: <https://patents.google.com/patent/US5583476A/en>.

[38] PATIL, K.; PENDHARKAR, G.; GAIKWAD, G. American sign language detection. *Int. J. Scien. R. Pub.*, v. 4, 2014.

[39] A., D. et al. Smart glove for sign language communications. *In Proceedings of the 2016 International Conference on Accessibility to Digital World (ICADW)*, Guwahati, India, December 2016.

[40] MUMMADI, C. et al. Real-time and embedded detection of hand gestures with an imu-based glove. *Informatics*, 2018.

[41] CARVALHO, C. Um sistema portátil de tradução de posturas do alfabeto manual de libras em voz utilizando luva instrumentalizada com sensores imu. Universidade de Brasília, 2019.

[42] ANDERSON, D. *Kanban Essentials*. Kanban University, Novembro 2020. Disponível em: <https://resources.kanban.university/shop/free-downloads/>.

[43] GITHUB Website. Disponível em: <https://github.com/>.

[44] TRELLO Website. Disponível em: <https://trello.com/>.

[45] ZOOM Website. Disponível em: <https://zoom.us/>.

[46] ZAITER, A.; SCHIAVINI, L. *Sign Language By Glove*. 2020. Disponível em: <https://github.com/abdullah-zaiter/Sign-Language-By-Glove>.

[47] ZAITER, A.; SCHIAVINI, L. *Glove Reader Flutter*. 2020. Disponível em: <https://github.com/abdullah-zaiter/Glove-Reader-Flutter>.

[48] ZAITER, A.; SCHIAVINI, L. *Glove Data Handler*. 2020. Disponível em: <https://github.com/abdullah-zaiter/Glove-Data-Handler>.

[49] ZAITER, A.; SCHIAVINI, L. *Recurrent NN Sign language*. 2020. Disponível em: <https://github.com/abdullah-zaiter/Recurrent-NN-Sign-Language>.

[50] MãO. Wikipedia. Disponível em: <https://pt.wikipedia.org/wiki/M~ao>.

[51] SYSTEMS, E. Esp32 wroom series overview. Jan 2018. Disponível em: <https://www.espressif.com/en/products/hardware/esp-wroom-32/overview>.

[52] 3 ANTHROPOMETRY AND BIOMECHANICS. National Aeronautics and Space Administration (NASA). Disponível em: <https://msis.jsc.nasa.gov/sections/section03.htm>.

[53] WEMOS D32 Pro Documentation. Wemos, 2019. Disponível em: <https://www.wemos.cc/en/latest/d32/d32_pro.html>.

[54] 3.7 v 800mah 25c lipo bateria 902540. AliExpress. Disponível em: <https://pt.aliexpress.com/item/32921774701.html?spm=a2g0o.productlist.0.0.39962aca5oUiju&algo_pvid=81b99d6d-c171-49d3-a1e6-a246fb5d0a99&algo_expid=81b99d6d-c171-49d3-a1e6-a246fb5d0a99-2&btsid=0bb0600116076577864247481e5e72&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_>.

[55] ESPRESSIF. *esp-idf*. GitHub, Mai 2019. Disponível em: <https://github.com/espressif/esp-idf>.

[56] RABELLO, N. *I2Cbus*. GitHub. Disponível em: <https://github.com/natanaeljr/esp32-I2Cbus>.

[57] RABELLO, N. *I2Cbus*. GitHub. Disponível em: <https://github.com/natanaeljr/esp32-MPU-driver>.

[58] PYTHON. Python Software Foundation. Disponível em: <https://www.python.org/>.

[59] GOOGLE Colaboratory. Google. Disponível em: <https://colab.research.google.com/>.

[60] GUNJAL, S. *K Fold Cross Validation*. Quality Tech Tutorials, 2020. Disponível em: <https://satishgunjal.com/kfold/>.

[61] CRISTIANO, A. *Alfabeto Manual de LIBRAS*. 07 2020. Disponível em: <https://www.libras.com.br/alfabeto-manual>.

[62] VR, H. *Glove HI5 VR BUSINESS*. 2020. Disponível em: <https://www.hi5vrglove.com/store/hi5glove>.

[63] NANSENSE. *Gloves NANSENSE*. 2020. Disponível em: <https://www.nansense.com/gloves/>.