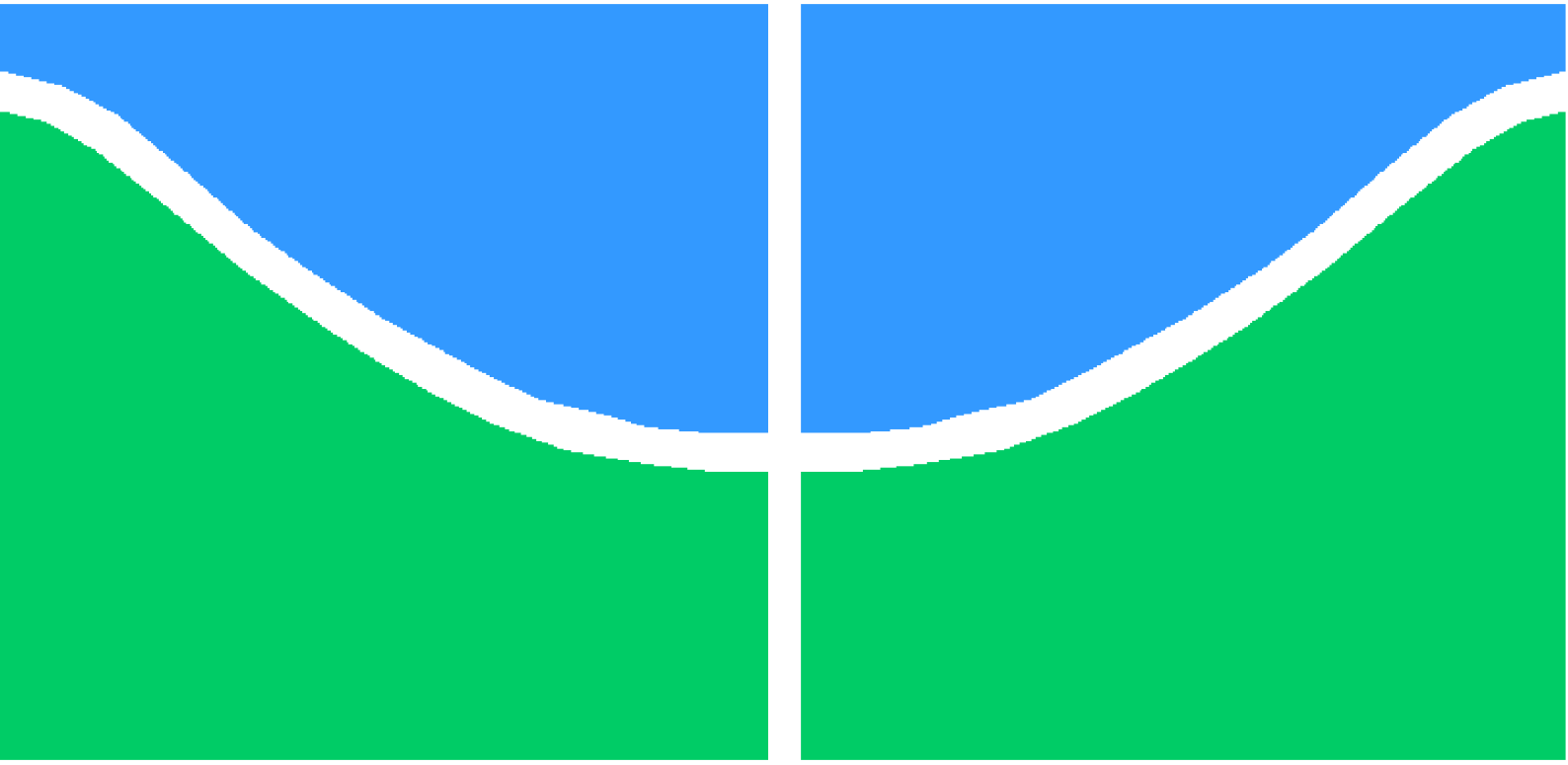


Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia Eletrônica

Luva Inteligente para traduzir LIBRAS em língua portuguesa usando método de Inteligência Artificial

Autor: Ítalo Rodrigo Moreira Borges

Orientadora: Profª. Dra. Suélia de S. R. F. Rosa

Coorientador: Me. Ronei Delfino da Fonseca Campos

Brasília, DF

2020

Ítalo Rodrigo Moreira Borges

Luva Inteligente para traduzir LIBRAS em língua portuguesa usando método de Inteligência Artificial

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA

Orientadora: Profª. Dra. Suélia de S. R. F. Rosa
Coorientador: Me. Ronei Delfino da Fonseca Campos

Brasília, DF
2020

Ítalo Rodrigo Moreira Borges

Luva Inteligente para traduzir LIBRAS em língua portuguesa usando método de Inteligência Artificial/ Ítalo Rodrigo Moreira Borges. – Brasília, DF, 2020 - 163 p. : il. (algumas color.) ; 30 cm.

Orientadora: Profª. Dra. Suélia de S. R. F. Rosa
Coorientador: Me. Ronei Delfino da Fonseca Campos

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2020.

1. LIBRAS. 2. Redes Neural Artificial. I. Profª. Dra. Suélia de S. R. F. Rosa. II. Me. Ronei Delfino da Fonseca Campos III. Universidade de Brasília. IV. Faculdade UnB Gama. V. Luva Inteligente para traduzir LIBRAS em língua portuguesa usando método de Inteligência Artificial

CDU 02:141:005.6

Ítalo Rodrigo Moreira Borges

Luva Inteligente para traduzir LIBRAS em língua portuguesa usando método de Inteligência Artificial

Monografia submetida ao curso de graduação em (Engenharia Eletrônica) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia Eletrônica).

Trabalho aprovado por:

Profª. Dra. Suélia de S. R. F. Rosa
(FGA - UnB)
Orientadora

Me. Ronei Delfino da Fonseca Campos
(PPMEC - FT)
Coorientador

Eng. Luciana Alves Fernandes
(FGA - UnB)
Examinadora Interna

Eng. Murilo Venturin
(FGA - UnB)
Examinador Interno

Me. Melissa Silva Monteiro
(IB - UnB)
Examinadora Externa

Brasília, DF, 27 de junho de 2020

.

A dedicatória deste trabalho vai para todas para todas as pessoas independente de idade que tem diversas limitações na realização dos sonhos, algumas até desistem por conta das dificuldades. A minha jornada até chegar nesse trabalhado teve muitas limitações, no entanto, tive muita ajuda e suporte de pessoas e profissionais que tornaram as dificuldade suportáveis. Então, persistam no sonho de vocês independente das limitações, porque são nas tentativas que vêm o êxito.

Agradecimentos

Agradeço primeiramente a Deus por abençoar-me em todas a fases desta grande jornada que foi a minha graduação em Engenharia Eletrônica. Ao meu pai que sempre me incentivou, ajudou, propôs reflexão para meu autocrescimento, à minha mãe que sempre esteve presente, incentivando a enfrentar as dificuldades de cabeça erguida e me ajudando nesse processo, ao meu irmão por sempre estar presente e ajudando-me da sua maneira na minha jornada. Uma profunda gratidão pela Profª. Dra. Suélia Rodrigues por ser sempre genial em suas orientações, engrandecendo de forma única este trabalho, trazendo reflexões, dicas, modos de aplicar, etc. Ao Me. Ronei Fonseca por sempre me orientar com clareza e de forma objetiva ajudando a tornar este trabalho em realidade. E por último, agradeço a todos os professores, técnicos em laboratórios, servidores da limpeza, servidores do refeitório, amigos e amigas que de alguma forma agregaram em meu crescimento.

A motivação deste trabalho não se baseia apenas no sentimento de ser responsável pela mudança do mundo. Além disso se funda no amor pela profissão, do sentimento de querer o melhor para o próximo, de incluir o seu semelhante na sociedade, de disseminar o conhecimento de forma democrática, pois todos têm o direito à igualdade de forma ponderada a sua realidade. (Ítalo R. M. Borges, 2019)

Resumo

O grupo de deficientes auditivos representam 5,10% da população brasileira e ocupa o terceiro lugar em tipos de deficiências. O presente trabalho visa a construção de uma luva que permite traduzir sinais de LIBRAS em sinais de áudio em língua portuguesa. Para projetar e implementar essa luva utilizou-se: o método matemático conhecido como redes neurais artificiais multicamada *perceptron* com retropropagação, um *MPU-6050* como sensor inercial, um micro *SD card* para armazenar os pesos da rede neural utilizada, um módulo MP3 para processar o sinal de áudio, um auto falante como saída sonora e confeccionou-se sensores flexíveis resistivos para adquirir e identificar o sinal de LIBRAS. Com o intuito de verificar a qualidade do sensor flexível resistivo, recorreu-se ao método de caracterização e para averiguar o bom funcionamento do sensor *MPU-6050* usou-se o *software processing 3D* que auxiliou na calibração do sensor. Desta maneira, verificou-se os indicadores de qualidade da rede neural artificial com a matriz de confusão que analisa a relação de dados na previsão ideal (dados de sinais em LIBRAS extraídos do *dataset*) com os dados previstos (sinais de LIBRAS classificado pelo sistema implementado) e utilizou-se a acurácia e a função erro *mean square error* como indicadores de qualidade do treino que resultaram respectivamente em 99,97% e 0,0021 . Em vista desses testes, identificou-se quais componentes, materiais e combinações que seriam mais apropriadas para o sistema. Os indicadores auxiliaram na escolha da arquitetura da rede neural artificial que possibilitou a implementação de uma solução tecnológica que beneficiará a comunidade de deficientes auditivos.

Palavras-chaves: LIBRAS, Rede Neural Artificial,Tradução, Luva.

Abstract

The hearing impaired group represents 5.10% of the Brazilian population and ranks third in types of disabilities. The present work aims at the construction of a glove that allows translating LIBRAS signals into audio signals in Portuguese. To design and implement this used glove: Mathematical method known as artificial neural networks multilayer perceptron backpropagation, *MPU-6050* as inertial sensor, micro *SD card* for weight storage, MP3 module for audio signal control , speaker as a sound output and flexible resistive sensors to obtain and identify or signal LIBRAS. In order to verify the quality of the sensor, the flexible resistor used the method of characterization and good functioning of the sensor *MPU-6050* used by the software *processing 3D* that helped in the calibration of the sensor. In this way, the quality indicators of the artificial neural network were verified with a confusion matrix that analyzes a relationship of data in the ideal forecast (LIBRAS sign data extracted from the *dataset*) with the data provided (LIBRAS signs classified by the implemented system) and accuracy and the mean square error function were used as indicators of training quality, which resulted in 99.97% and 0.0021, respectively. In view of these tests, it was identified which components, materials and combinations would be most appropriate for the system. In view of these tests, it was identified which components, materials and combinations are most appropriate for the system. The auxiliary indicators for choosing the artificial neural network architecture allow the implementation of a technological solution that benefits a community of hearing impaired people.

Key-words: LIBRAS, Artificial Neural Network, Translation, Glove .

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

PPD	Pessoa Portadora de Deficiência
PNE	Pessoa com Necessidades Especiais
DAs	Deficientes auditivos
LIBRAS	Língua Brasileira de Sinais
INES	Instituto de Educação dos Surdos
DACs	*Digital-To-Analog Converter*
VCC	Voltage in Current Continuous
VLOGIC	Voltage Logic
Constantan	(45% Níquel, 55% Cobre)
Nicromo	(80% Níquel, 20% Crômio)
ALU	Arithmetic Logic Unit
MIFA	*Meandered Inverted-F Antenna*
PWM	*Pulse Width Modulation*
SPI	*Serial Peripheral Interface*
UART	*Universal Asynchrounous Receiver/Transmiter*
ms	milissegundos
ns	nanossegundos
MSE	*Mean Square Error*
µ*m*	micrometro

cm	centímetro
RNA	Rede Neural Artificial
MP3	*MPEG-1 Audio Layer-3*
WAV	*WAVE Form audio format*
WMA	*Windows Media Audio*
FAT16	*File Allocation Table of 16 Bits*
FAT32	*File Allocation Table of 32 Bits*
SD card	*Secure Digital Card*
SDHC card	*Secure Digital High Capacity Card*
SDA	*Serial Data*
SCL	*Serial Clock*
GND	*Ground*
USB	*Universal Serial Bus*
API	*Application Programming Interface*
LUTLI	Luva Tradutora de LIBRAS Inteligente
IoT	*Internet of Things*

Sumário

1 Introdução

1.1 Motivação

A deficiência auditiva, também conhecida como hipoacusia ou surdez, caracteriza a perda ou redução, parcial ou total, da habilidade de detectar sons. Esse tipo de deficiência traz para o ser humano determinadas limitações e dificuldades em ouvir diálogos e outros sons. Ela pode se apresentar desde o nascimento, através da má formação do aparelho auditivo humano, mas também pode ser causada por lesões, doenças provocadas por ruídos ou pelo envelhecimento humano, que é uma consequência natural do organismo. Na maioria dos casos em que ela se apresenta, a deficiência auditiva não tem cura, porém em muitos deles, realiza-se o tratamento com o uso de aparelhos auditivos (SOARES, 2015).

O termo técnico para se referir a pessoa com necessidade especial de forma genérica é Pessoa Portadora de Deficiência (PPD), sendo mais utilizado do que Pessoa com Necessidades Especiais (PNE) (HENRIQUE, 2017). Há também uma definição específica de surdos que são os Deficientes Auditivos (DAs), porém essa não é muito usada. Os surdos têm sua linguagem chamada 'identidade surda', ou seja, comunicam-se através da Língua Brasileira de Sinais (LIBRAS) (HENRIQUE, 2017).

A surdez funcional é dividida entre surdez moderada, na qual as pessoas escutam em um nível de 41 a 55 decibéis; surdez acentuada, na qual o nível é de 56 a 70 decibéis; surdez severa, na qual o nível é de 71 a 90 decibéis e surdez profunda, na qual a surdez é acima de 91 decibéis (SOARES, 2015). Alguma dessas divisões podem ser corrigidas com aparelhos auditivos e as outras não conseguem ser beneficiados com aparelho e nem há uma cura, pelo fato de ser severa ou profunda (SOARES, 2015).

Para essas pessoas diagnosticadas com surdez severa ou profunda é extramente difícil ou quase impossível realizar ações nas quais para um pessoa que não tem deficiência são simples, como pedir informação para um motorista de ônibus,

frequentar uma escola, ser atendida por um defensor público ou que outra ação que dependa da comunicação.

Segundo dados do censo de 2010, 23,9% da população residente no país possuía pelo menos uma das deficiências seguintes: visual, auditiva, motora e mental ou intelectual (FERNANDES, 2019). A prevalência da deficiência variou de acordo com a natureza dela. A deficiência auditiva está em terceiro lugar dos casos apurados no brasil afetando 5,10% da população brasileira (FERNANDES, 2019).

No âmbito familiar a percepção da pessoa com deficiência auditiva adquirida, ficou prejudicada, sendo identificadas alterações importantes no seu comportamento e o desconhecimento do problema por parte dos deficientes e dos familiares, contribuindo para o agravamento em muitas situações de conflito e estresse (FERNANDES, 2019). O relato dos próprios entrevistados com deficiência auditiva verificou que eles passaram a enfrentar os mais diferentes tipos de dificuldades no relacionamento social, dentre os quais, a descriminação. Esse comportamento está relacionado ao desconhecimento dessa deficiência. As pessoas com deficiência compõem um grupo social heterogêneo, sobre o qual recaem, ao longo da história da humanidade, o estigma, o preconceito, a rejeição e a discriminação, as quais necessitam de uma proposta integrada a políticas públicas.

Devido essas circunstâncias e acrescentando que existe a falta de um equipamento assistencial para esses grupos de pessoas, percebeu-se que é necessário o desenvolvimento desse sistema assistencial.

1.2 Objetivo Geral

Projetar e implementar um sistema de inclusão social para os PPDs que permita traduzir, com o auxílio de uma luva inteligente, os movimentos em LIBRAS realizadas pelas mãos e os traduz de forma sonora para o idioma português brasileiro.

1.3 Objetivo Específico

- Confeccionar sensores flexíveis;
- Calibrar os sensores;
- Desenvolver um sistema de áudio;
- Desenvolver um sistema de alimentação;
- Integrar os sistemas com a luva;
- Consultar uma pessoa que sabe LIBRAS;
- Adquirir o *Dataset*;
- Projetar e treinar uma arquitetura de Rede Neural Artificial;
- Implementar a arquitetura no microcontrolador;
- Custo acessível a todas as classes sociais.

2 Fundamentação Teórica

Este capítulo expõe os conceitos e as ferramentas tecnológicas que fornecem a base para o desenvolvimento de um projeto tecnológico de cunho social. Serão apresentados conceitos referentes a LIBRAS, projetos similares, redes neurais artificias, sensores e atuadores.

2.1 Língua Brasileira de Sinais (LIBRAS)

LIBRAS é uma forma de se comunicar com gestos e sinais que permitem a comunicação dos PPDs. Durante a história da humanidade as escolas sempre influenciaram o aprendizado da língua desde a pronúncia até a escrita. No entanto, quando se tratava de língua de sinais era proibido. Os alunos usavam como uma forma alternativa e secreta de comunicação em banheiros e dormitórios para não receberem as punições (STOCK, 2018).

A verdadeira educação de surdos iniciou-se com Pedro Ponce De Léon (1520-1584), na Europa que era dirigida à educação de filhos Nobres. Pedro Ponce de Léon era Monge beneditino da Onã, na Espanha, onde estabeleceu a primeira escola para surdos em um monastério. Ele ensinava latim, grego e italiano, conceitos de física e astronomia aos dois irmãos surdos (MORI; SANDER, 2015).

Um professor surdo francês, Eduard Huet (1822-1882), veio para o Brasil sob os cuidados do imperador D. Pedro II. Os surdos até no final do século XV, eram considerados incapazes de se educar e com isso teve a intenção de inaugurar uma escola com modelo parecido com a educação dos surdos na Europa. Desse modo, LIBRAS tem como origem a língua francesa de sinais.

Os primeiros passos da linguagem em LIBRAS foi com o alfabeto manual, de origem francesa, os próprios alunos surdos de vários lugares do Brasil difundiram essa novidade por onde viviam. Em 26 de setembro de 1857, fundou-se no Rio de Janeiro a primeira escola para surdos no Brasil, intitulada Instituto de Educação dos Surdos (INES) e nesse mesmo dia comemora-se o Dia Nacional dos Surdos no

Brasil (STOCK, 2018).

2.2 Tradução de LIBRAS

Historicamente, os PPDs ficaram sem o direito de comunicarem-se durante 16 séculos. Após conquistar esse direito, apenas alguns indivíduos que sabiam LIBRAS podiam se comunicarem. Os PPDs sofrem muito com falta de tradutor em supermercados, defensoria pública, bancos, etc. A tecnologia vem avançando para resolver esse problema, no brasil existem algumas iniciativas:

- Trabalho 1: Em 2011, Na universidade de Brasília (UnB), Luan Caius Ramos desenvolveu um sistema de reconhecimento de gestos usando redes neurais artificias. Esse sistema usava técnicas de processamento e análise de imagem para conseguir classificar as seguintes letras em LIBRAS: 'A','B','C','D','H','I','L', 'V','W' e 'Y' (RAMOS, 2011).

- Trabalho 2: Em 2015, Na Universidade Federal da Bahia (UFB), Igor Leonardo Oliveira Bastos desenvolveu um sistema de reconhecimento de gestos que utilizava redes neurais artificias. Ele usou técnicas de processamento e análise de imagens para classificar 40 sinais de LIBRAS. Esses correspondem às letras do alfabeto 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'I','L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y'. Além destes, foram usados sinais que correspondem aos algarismos '1', '2', '4', '5', '7', e '9', assim como as palavras 'Adulto, 'América', 'Avião', 'Casa', 'Gasolina', 'Identidade', 'Juntos', 'Lei', 'Palavra', 'Pedra', 'Pequeno' e 'Verbo' (BASTOS, 2015).

- Trabalho 3: Em 2017, Na Universidade de Brasília (UnB), Ítalo Rodrigo Moreira Borges e Anderson Sales Rodrigues Pinto desenvolveram uma luva tradutora do alfabeto de LIBRAS que era formada por uma luva de lã, 4 sensores flexíveis resistivos caseiro, acelerômetros nos 3 eixos, giroscópio nos 3 eixos e módulo *bluetooth* que permitia enviar textos para um *smartphone* traduzidos pela luva.

- Trabalho 4: Em 2018, Na Universidade Federal do Amazonas (UFAM), Marcelo Chamy Machado desenvolveu um sistema de reconhecimento de sinais de

LIBRAS usando Redes Neurais Convolucionais. Classificou 10 sinais diferentes: FORÇA, FRENTE, ABACATE, ABACAXI, ABAFADO, ABANAR-SE, ABANDONAR, ABATIDO, ABELHA e ABENÇOAR (MACHADO et al., 2018).

Todos esses trabalhos são grandes iniciativas para resolver as dificuldades que os PPD's enfrentam diariamente. No entanto alguns desses trabalhos diferem quanto a solução obtida. O trabalho 1, trabalho 2 e trabalho 4 utilizaram a mesma solução que é a classificação por meio de imagens, já o trabalho 3 e esse trabalho proposto utilizam a mesma solução que é por meio de sensores flexíveis resistivos. A solução com o uso de imagem tem a vantagem do usuário não utilizar uma luva porque as vezes incomoda e não haveria a necessidade de trocar pilha ou recarregar o produto, no entanto a desvantagem em relação a solução que conta com os sensores flexíveis está no preço, dependência de uma empresa ou instituição que deveria instalar esse sistema para comunicar-se com os PPD's, necessita de um local que tenha energia e necessita de um técnico para manusear o sistema. E a solução proposta desse trabalho é um avanço em relação ao trabalho 2, porque essa solução proposta utiliza Redes Neurais artificias que permitem uma maior classificação de sinais, tem um sistema de áudio e a luva é mais confortável ao usuário.

2.3 *Sensor MPU-6050*

O *Sensor MPU-6050* é um sensor composto por um acelerômetro que é um instrumento capaz de medir a aceleração $[\frac{m}{s^2}]$ e um giroscópio que é um instrumento que mede velocidade angular $[\frac{^\circ}{s}]$.

Esse é um dispositivo integrado que detecta movimento combinando um giroscópio e um acelerômetro, ambos com 3 eixos, além de ter integrado um processador digital de movimento e todo esse conjunto de característica se comporta em um tamanho de 4 x 4 x 0.9 mm^3. Com seu barramento de I2C dedicado, ele aceita diretamente entradas de uma bússola externa de 3 eixos para fornecer uma saída completa de informações sobre os 9 eixos (LAGE; SEGUNDO, 2016).

Além de fornecer informações sobre os eixos o *sensor MPU-6050* contém uma interface com múltiplos sensores digitais não inerciais, alguns sensores de pressão em sua porta I2C auxiliar, possui três *Digital-To-Analog Converter* (DACs) de 16 bits para a digitalização das saídas do giroscópio e três DACs de 16 bits para digitalizar as saídas do acelerômetro. Para o rastreamento de precisão de movimentos rápido e movimentos lentos, a tabela 1 exemplifica as faixas em que o acelerômetro e giroscópio fornece de informação para o usuário:

Tabela 1 – Escala completa de movimentos lentos e rápidos do giroscópio e acelerômetro para programação no sistema. Fonte: (LAGE; SEGUNDO, 2016).

Instrumento	**Unidade**	**Faixas**
Acelerômetro	[g]	± 250,± 500, ± 1000 e ± 2000
Giroscópio	$[\frac{\circ}{s}]$	± 2, ± 4, ± 8 e ± 16

O sensor apresenta no chip um sistema para reduzir o consumo de energia do sistema, permitindo que o processador do sistema possa ler os dados do sensor em pacotes maior, e em seguida, entrar em um modo de baixo consumo de energia à medida que o *MPU-6050* coleta mais dados, oferecendo uma vida útil mais longa (LAGE; SEGUNDO, 2016).

A comunicação com todos os registros do dispositivo é realizada usando I2C a 400 kHz. A peça apresenta uma robusta tolerância a choques de 10.000 g, e possui filtros de baixa frequência para os giroscópios, acelerômetros e o sensor de temperatura no chip. Para a flexibilidade da fonte de alimentação, o *sensor MPU-6050* opera à partir da faixa de tensão da fonte de alimentação *Voltage in Current Continuous* (VCC) de 2.37 - 3.46 volts. Além disso, o *sensor MPU-6050* fornece um pino de *Voltage Logic* (VLOGIC), que define os níveis lógicos de sua interface I2C. A tensão VLOGIC pode ser de 1,8 volts ± 5% ou VCC (LAGE; SEGUNDO, 2016).

2.4 Sensor Flexível Resistivo

O conceito de resistência leva em consideração as dimensões do elemento que pode ser por exemplo, o carbono de um lápis distribuído sobre uma folha de papel formando uma camada fina de carbono, na qual apresenta uma resistência que é representada pela seguinte expressão (BOYLESTAD; NASHELSKY, 2013):

$$R = \frac{\rho.l}{S} = \frac{\rho.l}{t.w} \tag{2.1}$$

onde:

- R= Resistência;
- ρ = Resistividade do Material;
- l = Comprimento;
- S = Área do Material;
- t = Espessura;
- w = Largura;

Como t é uma constante para este modelo matemático, então pode-se representar a constante resistência de folha como:

$$R_f = \frac{\rho}{t} \tag{2.2}$$

Esse conceito pode ser aplicado para confecção de sensores flexíveis resistivo de forma controlada, levando em consideração o material com ρ e a espessura t . Ao utilizar condutores em diferentes extremidades da folha resistiva e mensurar a resistência, apresentará uma resistência definida pela equação 2.1 que ao encapsular esse condutores com a folha resistiva terá um sensor flexível que ao ser flexionado varia a resistência.

O extensômetro elétrico de resistência é outro tipo de sensor resistivo flexível que é um elemento sensível que ao ser flexionado transforma pequenas variações

físicas em variações de resistência elétrica. O extensômetro é fabricado com uma camada muito fina de metal (3 a 8 micrômetros de espessura) parte da qual é removida com corrosão até ficar no padrão da figura 1. Essa grade metálica é fixada em um apoio isolante. As ligas resistivas utilizadas para fabricação são *Constantan* e *Nicromo* que permitem obter resistências padronizadas de 60 , 120, 240, 350, 500 e 1 kΩ (THOMAZINI, 2018).

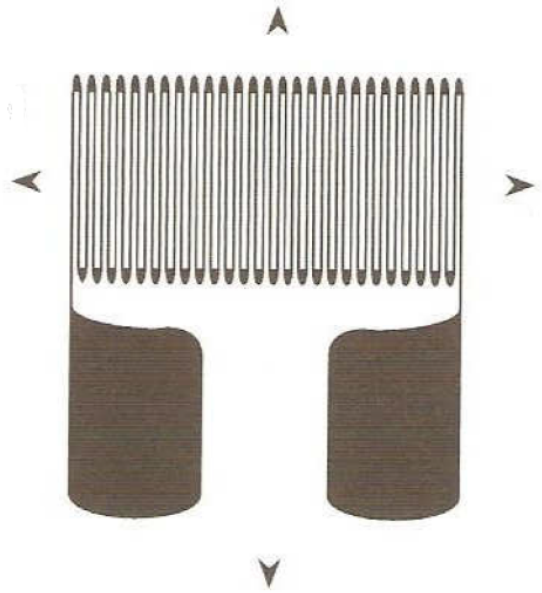


Figura 1 – Modelo de extensômetro: Esse é um *strain gage* normal, que tem como característica evitar as deformações nos sentidos das setas. Fonte: (THOMAZINI, 2018).

Apesar do extensômetro ser um ótimo sensor para aplicações como: deformações, compressão e tensão de cisalhamento. Ele deixa a desejar quanto a deformações de flexão e a área do sensor é menor do que a necessária para abranger os dedos da mão (THOMAZINI, 2018). Então para aplicação de uma luva com sensores flexíveis, a solução mais interessante seria o sensor da folha resistiva.

2.5 Microcontrolador *ESP32-WROOM-32*

Um microcontrolador é um sistema que executa funções específicas ao serem programadas em um chip. Esse chip contém um processador do tipo *Arithmetic Logic Unit* (ALU), memória, periféricos de entrada e de saída, temporizadores, dispositivos de comunicação serial e entre outros de acordo com figura 2 (SENAI-SP, 2018).

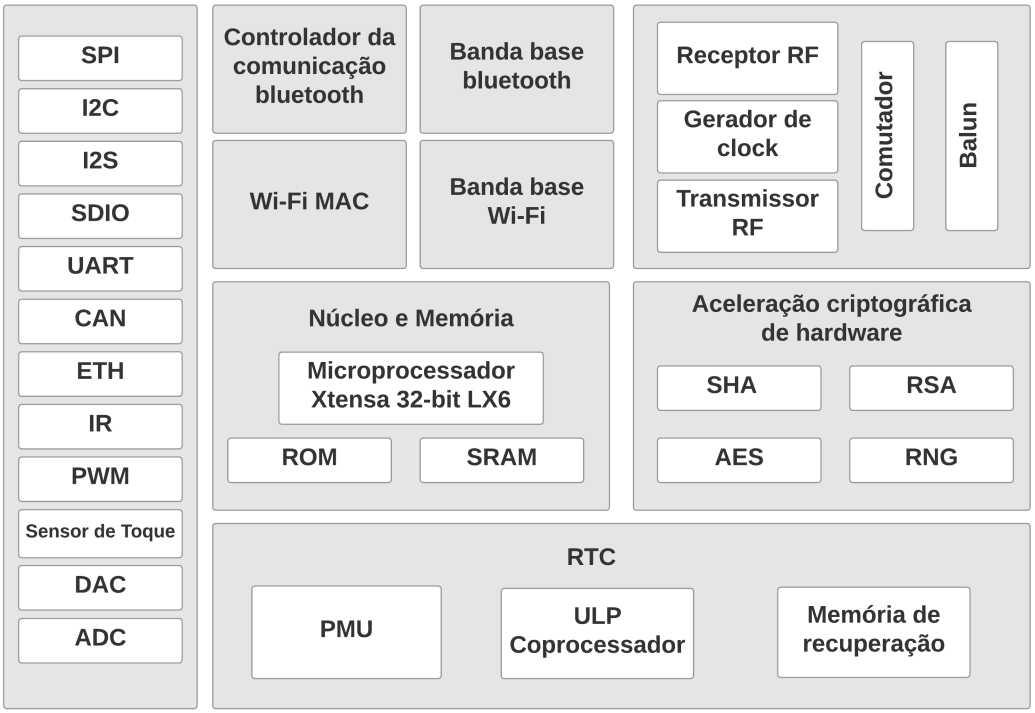


Figura 2 – Representação da arquitetura da *ESP32-WROOM-32* e seus periféricos. Fonte: (ESPRESSIF, 2019).

Cada microcontrolador tem suas particularidades como portas de entrada, de saída, digitais, analógicas ou memórias flash. Abaixo, a tabela 2 apresenta um dos principais modelos da família ESP-32 e suas configurações diferentes (ESPRESSIF, 2019):

Utilizando a tabela 2 para analisar a memória *flash* e sabendo que a *ESP32-WROOM-32* é a mais comercializada, logo, ela se torna ideal para esse projeto. Ela conta com processador dual core, 4 MB de memória flash, 36 pinos, sendo 16

Tabela 2 – Principais dos módulos da família ESP-32. Fonte: (ESPRESSIF, 2019).

Módulo	**Antena**	**Flash, MB**
ESP32-WROOM-32	MIFA	4
ESP32-WROOM-32U	U.FL	4, 8, ou 16
ESP32-WROVER (IPEX)	U.FL	8
ESP32-WROVER-B	MIFA	4, 8, ou 16

ADC de 12 bits, 16 PWM, 4 SPI, 2 I2C, 2 UART como apresenta a figura 3. O principal requisito que direcionou a escolha do microcontrolador foi o poder de processamento, visto que o projeto exige embarcar uma arquitetura de inteligência artificial (ESPRESSIF, 2019).

Figura 3 – Chip que contém o microcontrolador *ESP32-WROOM-32*, portas de entrada, saída, memória e entre outros periféricos. A esquerda encontra-se a vista superior do chip e a direita a vista inferior com a sua pinagem. Fonte: (ESPRESSIF, 2019).

2.6 Rede Neural Artificial

Rede neural artificial (RNA) é um paradigma da inteligência artificial que se baseia no funcionamento do neurônio biológico em interação com outros neurônios, como por exemplo, as ativações sinápticas do cérebro resultando de uma ação. A rede neural é definida como um processador paralelamente distribuído constituído de unidades de processamento simples (neurônio), que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso. Ela se assemelha ao cérebro em dois aspectos (HAYKIN, 2007):

1. O conhecimento é adquirido pela rede a partir do seu ambiente através de um processo de aprendizagem.

2. Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

2.6.1 Neurônio Biológico

Através do trabalho pioneiro de (Ramón e Cajál , 1911), que introduziram a ideia dos neurônios como unidades constituintes estruturais do cérebro. Tipicamente, os neurônios são de cinco a seis ordens de grandeza mais lentos que as portas lógicas de silício, os eventos neurais biológicos ocorrem na ordem de *ms*, enquanto os eventos em circuitos de silício acontecem na ordem de *ns*. No entanto, apesar dos circuitos de silício serem mais rápido do que as sinapses cerebrais, esses formam Redes Neurais Artificiais (RNA) mais rápidas do que o outro em termos de processamento, porém, até o momento não mais eficientes que as redes neurais biológicas pelo fato da estrutura neural ser mais simples.

A estrutura biológica contém aproximadamente 86 bilhões de neurônios no córtex humano e mais de 60 trilhões de sinapses ou conexões tornando-a uma rede mais complexa em termos de conexões. Outra curiosidade é que a eficiência energética do cérebro é de aproximadamente (10^{-16}) por operação por segundo, enquanto nos melhores computadores são (10^{-6}) por operação por segundo (HAYKIN, 2007; LENT, 2010).

O neurônio biológico é um célula com 100 μm de extensão formada por um corpo celular, dendritos, axônio e terminais sinápticos como demonstra a figura 4 (HAYKIN, 2007; LENT, 2010).

- Corpo Celular (Soma): É o local onde o neurônio codifica as suas saídas de acordo com uma série de pulsos breves de tensão e o local onde é produzido as proteínas necessárias.
- Dendrito: São as zonas receptoras do neurônio e onde são processadas as informações. Sinais recebidos podem ser excitatórios, fazendo o neurônio disparar (gerar um impulso elétrico) ou inibitórios, tendendo a evitar que o neurônio dispare.
- Axônio: São linhas de transmissão interna do neurônio.
- Ramificações do Axônio: Permite que o neurônio faça mais de uma conexão sináptica.
- Terminais Sinápticos: São terminais que permitem a conexão sinápticas entre neurônios.

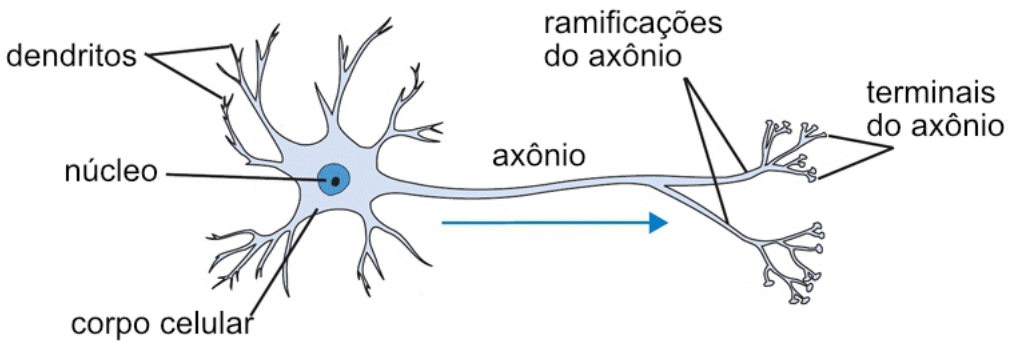


Figura 4 – Representação de um neurônio biológico, na qual apresenta os componentes e a seta azul que indica a direção após uma sinapse. Fonte: (BEZERRA, 2016).

As sinapses são unidades estruturais e funcionais elementares que permitem as interações entre neurônios. Elas são um fenômeno químico e elétrico, cujo um processo pré-sináptico libera uma sustância transmissora que se difunde através da junção sináptica entre nerônios e então age sobre um processo pós-sináptico, assim uma sinapse converte um sinal elétrico pré-sináptico num sinal químico e em seguida de volta em um sinal pós sináptico (LENT, 2010) (HAYKIN, 2007).

2.6.2 Modelo de McCulloch e Pitts

A busca por um modelo computacional que se assemelha a uma rede neural começou com o trabalho de (McCulloch e Pitts, 1943). Eles propuseram modelos de neurônios na forma de dispositivos de limiarização binária e algoritmos estocásticos, envolvendo mudanças súbitas de 0 para 1 e 1 para 0 dos estados dos neurônios como base para a modelagem das redes neurais. Os trabalhos posteriores de (Hebb, 1949) foram importantes para mostrar a importância do conceito de aprendizagem por reforço ou associação (GONZALES; WOODS, 2011; HAYKIN, 2007).

Em meados dos anos 1950 e início dos anos 1960, uma classe das chamadas máquinas de aprendizagem criadas por (Rosenblatt, 1959-1962) causou empolgação entre pesquisadores e profissionais na área de reconhecimento de padrões, pois era um modelo teórico bem fundamentado de aprendizado, em que máquinas eram chamadas de *perceptrons* (modelo matemático de um neurônio) e quando treinadas com um conjunto de dados linearmente separáveis, iriam convergir a uma solução em um número finito de passos iterativos conhecido como épocas (GONZALES; WOODS, 2011; HAYKIN, 2007).

De acordo com a figura 5 o primeiro modelo matemático robusto é definido como:

- Vetores de Características (*dataset*): São chamadas de entradas do neurônio artificial. A funcionalidade dela dentro do neurônio é fornecer uma saída ou que a chamamos de classe de padrão. Então, para cada vetores de características semelhantes fornecem um probabilidade de uma classe de padrão. Ex:

$$X_i = [X_1, X_2, X_3, ..., X_n] \tag{2.3}$$

- Classe: É a saída do neurônio onde informa o padrão correspondente a entrada, podendo ser binária ou multiclasse. Ex:

$$O_i = [O_1, O_2, O_3, ..., O_n] \tag{2.4}$$

- Função de Ativação: É a função na qual fornece uma probabilidade na saída (classe do padrão). Ela tem uma característica em que a saída geralmente fornece valores entre 1 e 0 ou entre 1 e -1. Ex:

$$f(n) = \begin{cases} 1, & \text{se } n \text{ d(x)} > 0 \\ -1, & \text{se } n \text{ d(x)} < 0 \end{cases} \tag{2.5}$$

- Função de decisão: É a função fornecida ao final do treinamento do neurônio que determina uma classe a partir de um vetor característico. Ex:

$$d(x) = \sum_{k=1}^{N} W_i X_i + W_{n+1} \tag{2.6}$$

- Pesos: São variáveis que se ajustam no processo de aprendizagem afim de encontrar a melhor função de decisão. Inicialmente atribuídas com valores aleatórios, a ideia é que com tempo de treinamento ela aprende o peso ideal. Ex:

$$W_i = [W_1, W_2, W_3, ..., W_n] \tag{2.7}$$

- Bias: São pesos com valor constante igual a 1. É a constante que multiplica o W_{n+1}.

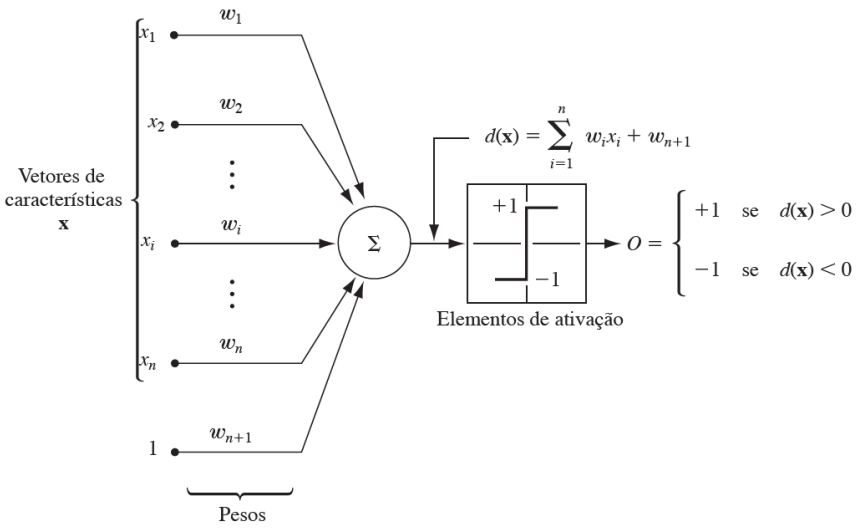


Figura 5 – Modelo *perceptron* para duas classes de padrões, no qual demonstra de forma genérica a quantidade de entradas relacionadas com os pesos e a função de ativação. Fonte: (GONZALES; WOODS, 2011).

2.6.2.1 Porta OU - Problema linearmente separável

A figura 6 demonstra que as classes são linearmente separáveis, ou seja, ao traçar um segmento de reta na diagonal, pode-se separar a classe 1 representada pelo losango de cor avermelhada de saída igual à 0 e a classe 2 representada pelo círculo azulado de saída igual a 1. A lógica dessa porta é descrita pela tabela 5, na qual basta uma das entradas serem de nível lógico alto, representado por '1' que a saída será de nível lógico alto, caso contrário, a saída será de nível lógico baixo representado por '0'.

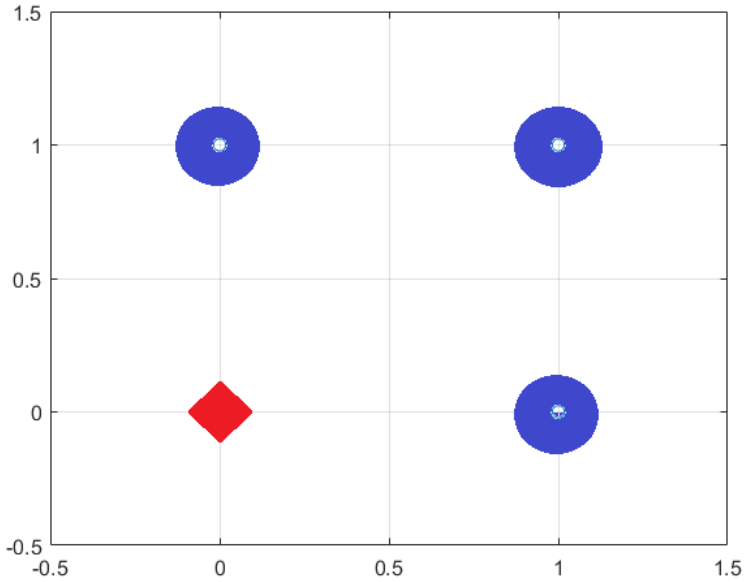


Figura 6 – Análise gráfica de classe da porta OU, na qual o eixo das abcissas e ordenadas representam os valores lógicos de entrada e o losango vermelho representa nível lógico baixo na saída e o círculo azul nível lógico alto na saída. Fonte: Autor.

A figura 7 mostra o modelo do neurônio para o problema da porta OU, essa modelagem apresenta duas entradas e uma saída, onde a entrada apresenta 4 possibilidades e a saída 2 possibilidades.

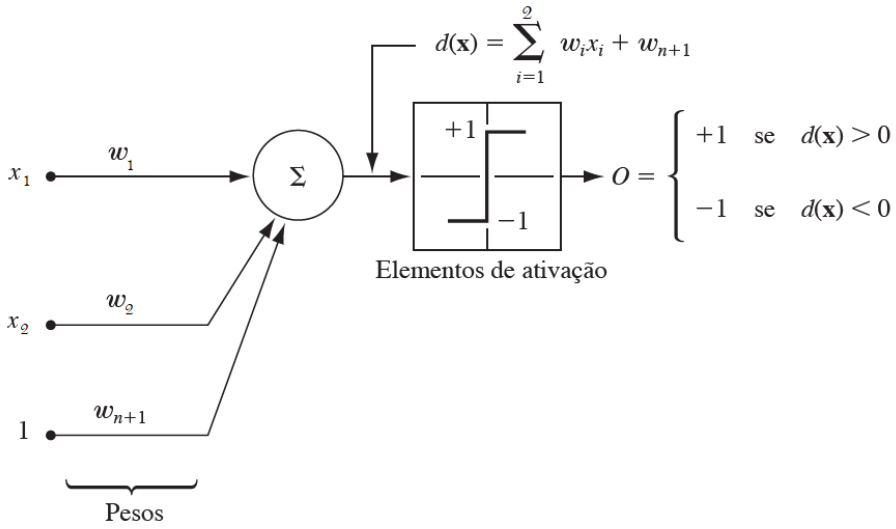


Figura 7 – Modelo *perceptron* do Neurônio da porta OU, no qual demonstra de forma particular as 2 entradas relacionadas com os pesos e a função de ativação. Fonte: (GONZALES; WOODS, 2011).

Tabela 3 – Tabela verdade da porta OU. A tabela relaciona a saída em função dos valores das entradas X1 e X2.

X_1	X_2	**Saída**
0	0	0
0	1	1
1	0	1
1	1	1

Sabendo que a função de decisão é:

$$d(x) = \sum_{k=1}^{N} W_i X_i + W_{n+1} \tag{2.8}$$

E sabendo que as entradas são $X_i = [X_1, X_2, Bias] = [0,0,1]; [0,1,1]; [1,0,1]; [1,1,1]$ e que a saída correspondente é $O_i = [Não, Sim, Sim, Sim]$ ou em binário $O_i =$

$[0, 1, 1, 1]$ de acordo com a tabela 3. Agora falta determinar os pesos, logo, pode-se utilizar a seguinte técnica para o aprendizado (GONZALES; WOODS, 2011):

- Para a classe 1, ou seja, a classe que $O_1 = 0$:

$$W^T(K)X(K) \leq 0 \longrightarrow W(K+1) = W(K) + cX(K) \tag{2.9}$$

- Caso contrário:

$$W^T(K)X(K) > 0 \longrightarrow W(K+1) = W(K) \tag{2.10}$$

  Onde K é quantidade de iteração e C é constante de ajuste.

- Para a classe 2, ou seja, a classe que $O_2 = O_3 = O_4 = 1$:

$$W^T(K)X(K) \geq 0 \longrightarrow W(K+1) = W(K) + cX(K) \tag{2.11}$$

- Caso contrário:

$$W^T(K)X(K) < 0 \longrightarrow W(K+1) = W(K) \tag{2.12}$$

  Algoritmo de aprendizagem com o ajuste c=1:

- K=1 e sabendo que a classe 1 será utilizada e inicializando o pesos em zero, tem-se que:

$$W(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.13}$$

- Logo:

$$W^T(1)X(1) = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0 \tag{2.14}$$

- Então:

$$W(2) = W(1) + X(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.15}$$

- K=2 e sabendo que a classe 2 será utilizada:

$$W(2) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.16}$$

- Tem-se:

$$W^T(2)X(2) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = 1 \tag{2.17}$$

- Então:

$$W(3) = W(2) - X(2) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \tag{2.18}$$

- K=3 e sabendo que a classe 2 será utilizada:

$$W(3) = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \tag{2.19}$$

- Tem-se:

$$W^T(3)X(3) = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = 0 \tag{2.20}$$

- Então:

$$W(4) = W(3) - X(3) = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \tag{2.21}$$

- K=4 e sabendo que a classe 2 será utilizada:

$$W(4) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \tag{2.22}$$

- Tem-se:

$$W^T(4)X(4) = \begin{bmatrix} -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = -3 \tag{2.23}$$

- Então:

$$W(5) = W(4) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} \tag{2.24}$$

- O processo de aprendizagem da máquina continua fazendo X(5) = X(1), X(6) = X(2), X(7) = X(3) e X(8) = X(4), prosseguindo da mesma maneira. Uma das convergência é atingida para k = 17, levando a uma solução $W(17)^T$ = (–2, -2, 1). A função de decisão correspondente é d(x)= -2X_1-2X_2 + 1.

Fazendo um teste simples para verificar se a função de decisão satisfaz as combinações da porta OU:

- A Classe 1 pode-se interpretada como números positivos de acordo com a figura 7:

$$X_1 = X_2 = 0 \longrightarrow d(x) = d(x) = -2X_1 - 2X_2 + 1 = 1 \tag{2.25}$$

- A classe 2 pode-se ser interpretada como números negativos de acordo com a figura 7:

$$X_1 = 0, X_2 = 1 \longrightarrow d(x) = d(x) = -2X_1 - 2X_2 + 1 = -1 \tag{2.26}$$

$$X_1 = 1, X_2 = 0 \longrightarrow d(x) = d(x) = -2X_1 - 2X_2 + 1 = -1 \tag{2.27}$$

$$X_1 = X_2 = 1 \longrightarrow d(x) = d(x) = -2X_1 - 2X_2 + 1 = -3 \tag{2.28}$$

2.6.3 Rede Neural Artificial Multicamada *Perceptron* com Retropropagação

O modelo de McCulloch e Pitss foi um grande passo para o conceito de RNA, no entanto, havia problemas não linearmente separável, logo esse modelo era ineficiente. Alguns anos mais tarde, (Minsky e Papert, 1969) apresentaram uma análise desanimadora das limitações das máquinas tipo *perceptron*. Essa opinião foi mantida até meados dos anos 1980 (GONZALES; WOODS, 2011).

Em 1986, Rumelhart, Hinton e Williams desenvolveram novos algoritmos de treinamento para os *perceptrons*, porém, agora usando múltiplas camadas, nos quais mudaram-se consideravelmente a história de aprendizado de máquina. O método básico deles, geralmente chamado de regra generalizada delta ou gradiente descendente para o aprendizado por retropropagação, fornece um método de treinamento eficaz para as máquinas de múltiplas camadas. A regra generalizada delta tem sido usada com sucesso em diversos problemas de interesse prático, incluindo os não linearmente separável. Esse sucesso fez com que as máquinas de múltiplas camadas do tipo *perceptron* fossem um dos principais modelos de redes neurais atualmente em uso (GONZALES; WOODS, 2011).

Pode-se constatar que o modelo de McCulloch e Pitts se manteve com algumas generalizações (veja o anexo A). Esse modelo propõe a interconexões entre os neurônios com um função erro baseada no gradiente descendente capaz de direcionar um realimentação mais eficiente afim de que o aprendizado seja eficaz e rápido.

- Entrada da rede:

$$S_a = \sum_{i=1}^{N} W_{ia} X_i + B_a \tag{2.29}$$

- Função de ativação, da qual pode existir diversas função de ativação como:

$$h_a = f(S_a) \tag{2.30}$$

- Entrada na primeira camada oculta:

$$S_b = \sum_{i=1}^{N} W_{ab} h_a + B_b \tag{2.31}$$

- Função de ativação na primeira camada oculta:

$$h_b = f(S_b) \tag{2.32}$$

- Entrada na segunda camada oculta:

$$S_c = \sum_{i=1}^{N} W_{bc} h_b + B_c \tag{2.33}$$

- Função de ativação na Segunda camada oculta:

$$h_d = f(S_c) \tag{2.34}$$

- Fazendo com a mesma lógica:

$$... \tag{2.35}$$

- Entrada na N-ésima camada oculta :

$$S_z = \sum_{i=1}^{N} W_{yz} h_{n-1} + B_z \tag{2.36}$$

- Saída da RNA após a função de ativação:

$$Y = f(S_z) \tag{2.37}$$

Fase da Retropropagação

- Função erro:

$$e_l = Y_e - Y \tag{2.38}$$

- Onde Y_e é a a saída esperada e Y é a saída da RNA.

Função Custo ou função perda:

- A função custo com a função erro como variável de entrada forma, o que conhece como MSE.

$$J = \frac{1}{2}\sum_{l=1}^{N} e_l^2 \tag{2.39}$$

- Gradiente da camada de saída:

$$\nabla_s = e_l \frac{\partial f(S_z)}{\partial W_{zy}} \tag{2.40}$$

- Gradiente da N-ésima camada oculta:

$$\nabla_z = \frac{\partial h_{n-1}}{\partial W_{xy}} W_{yz} \nabla_s \tag{2.41}$$

$$... \tag{2.42}$$

- Gradiente da segunda camada oculta:

$$\nabla_c = \frac{\partial f(S_b)}{\partial W_{ab}} W_{bc} \nabla_d \tag{2.43}$$

- Gradiente da primeira camada oculta:

$$\nabla_b = \frac{\partial f(S_a)}{\partial W_{ia}} W_{ab} \nabla_c \tag{2.44}$$

- Atualizando os pesos:

$$W_{ia} = (W_{ia}\eta) + (X_i \alpha \nabla_b) \tag{2.45}$$

$$W_{ab} = (W_{ab}\eta) + (f(S_a)\alpha \nabla_c) \tag{2.46}$$

$$W_{bc} = (W_{bc}\eta) + (f(S_b)\alpha\nabla_d) \tag{2.47}$$

$$... \tag{2.48}$$

$$W_{yz} = (W_{yz}\eta) + (f(S_y)\alpha\nabla_z) \tag{2.49}$$

- No qual η é o momento e α é a taxa de aprendizagem que são fatores de ajuste (otimizador) que permitem chegar mais rápido ao erro mínimo ou mais lento. Quanto menor o erro, melhor será o modelo da RNA.
- É comum os algoritmos utilizarem os otimizadores (atualizadores de peso) e definir os *bacth size* para que a função erro convirja de forma mais rápida e seja mais genérica.
- O otimizador adam é configurado com a $\alpha = 0{,}001$; β_1=0,9; β_2=0,999 e $\epsilon = e^{-7}$
- No qual β_1 e β_2 são as taxas de decaimento da exponencial, ϵ é uma constante para estabilidade numérica e α é a taxa de aprendizagem.

Após esse processo, repete-se desde o início desse algorítimo com os pesos atualizados pelas equações (2.45) até (2.49) e continua até que a função erro seja bem próxima de zero, onde pode-se afirmar que a RNA aprendeu. A função erro é uma variável fundamental para determinar a acurácia do modelo e além dela existe outro índice de desempenho que é a matriz de confusão. Ela é um matriz que as linhas indicam a quantidade de classes reais e as colunas mostram a quantidade das classes da RNA treinada. A diagonal principal desta matriz informa quantas classes da RNA treinada são iguais as classes reais, ou seja, a quantidade de classificação feita com sucesso (KERAS...,).

A figura 8 demonstra a importância de utilizar mais de uma camada de neurônios. A camada única sempre ficará restrita a um hiperplano e ao aumentar a quantidade de camadas ocultas, aumenta-se a capacidade de resolver problemas mais complexos como os problemas não lineares.

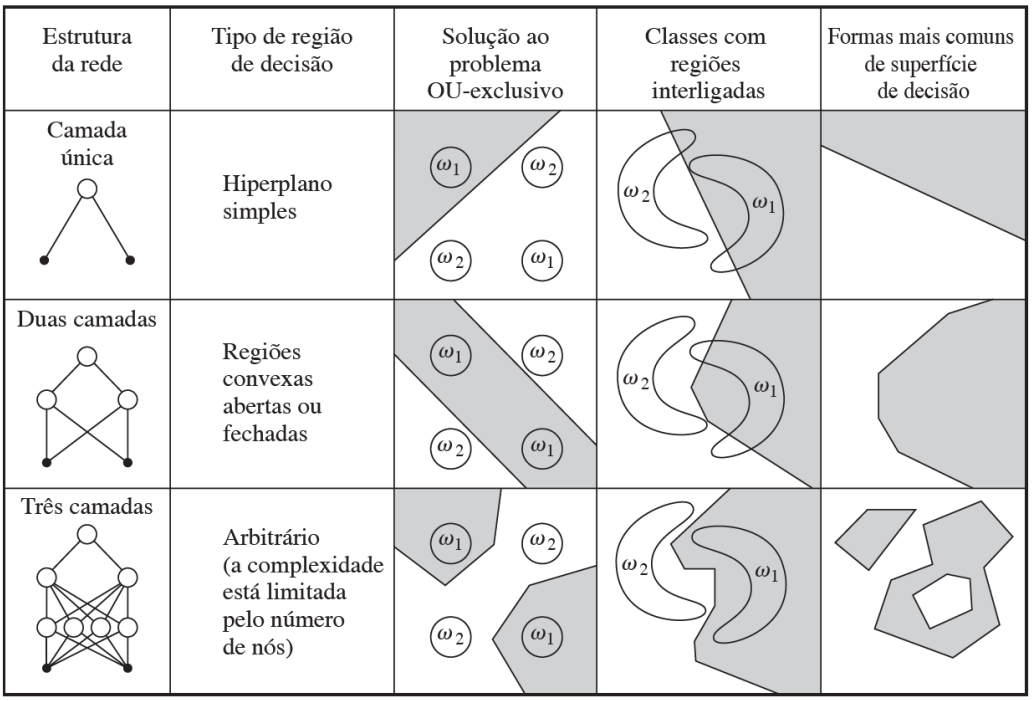


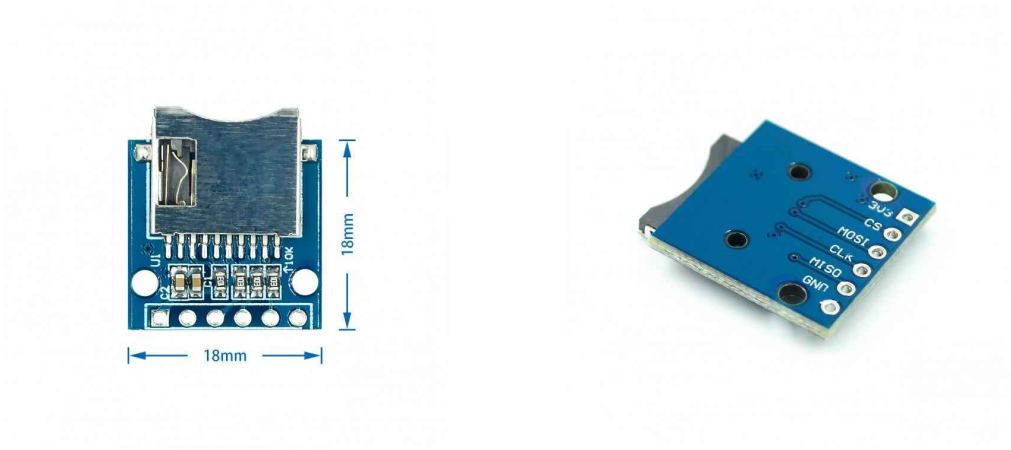
Figura 8 – Efeitos da quantidade de camada oculta em três situações distintas que impactam no resultado do treino da RNA. Fonte: (GONZALES; WOODS, 2011).

2.7 Sistema de Aquisição e Armazenamento de dados

O módulo *SD card* ou leitor micro *SD card* na figura 9 é um dispositivo muito eficiente desenvolvido para facilitar o trabalho de projetistas e no desenvolvimento de projetos que necessitam da gravação ou escrita de texto em documentos. Ele possui um leitor de cartão *SD* integrado que se comunica através do sistema de arquivos e do *driver* de interface SPI e um sistema que completa o arquivo de leitura e escrita (USINAINFO, 2019).

O módulo leitor micro *SD card* permite comunicação com dois tipos de cartões: o cartão micro *SD card* tradicional e o micro *SDHC card*, conhecido por ser um cartão de alta velocidade, o que torna sua aplicação muito simples e útil. A interface de comunicação é uma interface SPI, permitindo o microcontrolador trabalhar com diversos módulos *SD card*, sem haver conflito de dados. A alimentação

do módulo está entre 3.3 volts até 5 volts (USINAINFO, 2019).



(a) Visão Superior (b) visão Inferior

Figura 9 – Módulo micro *SD card* é apresentado com a dimensão da área, os pinos e a entrada *slot SD card*. Fonte:(USINAINFO, 2019).

3 Materiais e Métodos

Este capítulo descreve o desenvolvimento dos métodos e materiais utilizados a fim de desenvolver um sistema eletrônico capaz de transformar os sinais de LIBRAS em língua portuguesa brasileira em formato de áudio. Para isto, houve a necessidade de definir métodos para calibrar o *MPU-6050*; confeccionar os sensores flexíveis resistivos, visando alcançar um custo e benefício; integrar a *ESP32-WROOM-32* com o sistema de áudio que faz o processamento de sinais de áudio e com módulo micro *SD card* para adquirir e armazenar o *dataset* e as pilhas que tem a função da alimentação do sistema.

Visando definir a quantidade e os tipos de sinais em LIBRAS, consultou-se uma pessoas que se comunica nesta linguagem. Após isto definiu-se 22 sinais. E em seguida, executou-se a aquisição do *dataset*, projetou-se a arquitetura da RNA, treinou-la e por fim, testou a arquitetura com os pesos treinados no microcontrolador *ESP32-WROOM-32*. O apêndice 10 apresentar um pequeno fluxograma do passo a passo dos métodos utilizados.

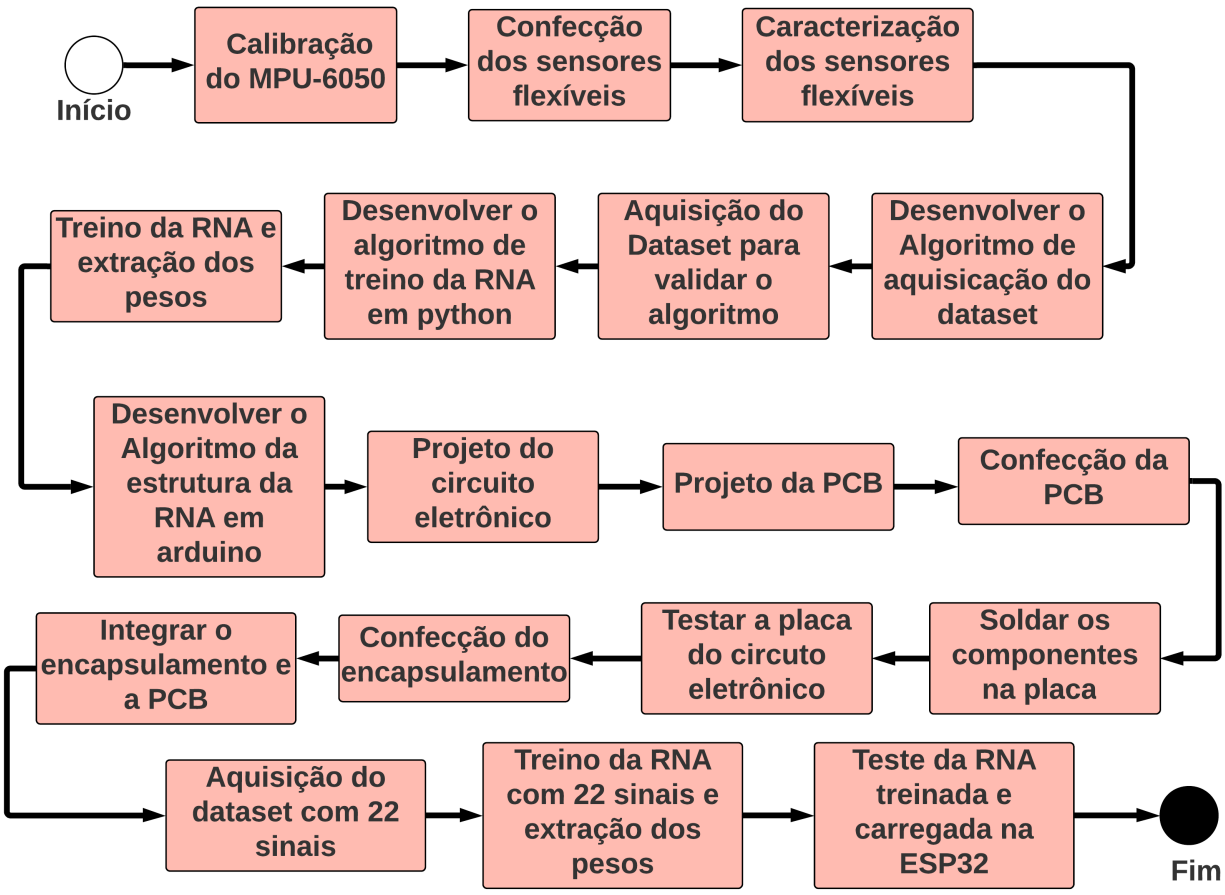


Figura 10 – Fluxograma de Materiais e Métodos. Fonte: Autor.

3.1 Visão Geral

O sistema de inclusão social chamado Luva Tradutora de LIBRAS Inteligente LuTLI é uma composição de duas funções. Uma delas é a de aquisição de dados demonstrado na figura 11, nessa função foi necessário calibrar o sensor MPU-6050, confeccionar sensores flexíveis, incluir no módulo SD no sistema e conectá-los a um microcontrolador com alimentação 3.3 volts. Já a segunda função, demonstrado pela figura 12, é um sistema de atuação que utiliza todos os elementos da função de aquisição, exceto módulo *SD* e acrescentado dos módulo

MP3 e auto-falante para exibir o áudio da tradução de sinas. Por fim, confeccionou uma PCB híbrida que fosse possível utilizar tanto a função aquisição quanto na função de atuação e um encapsulamento para fixar-se ao braço do usuário.

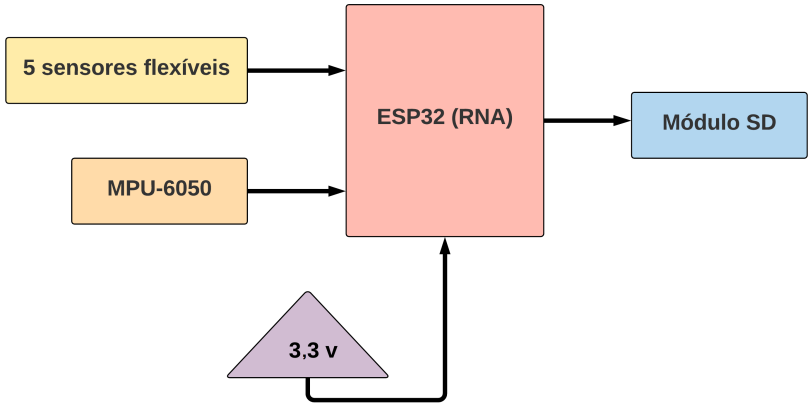


Figura 11 – Diagrama do circuito de aquisição de dados, no qual os sensores flexíveis, *MPU-6050* e alimentação são entradas e o módulo *SD card* saída. Fonte: Autor.

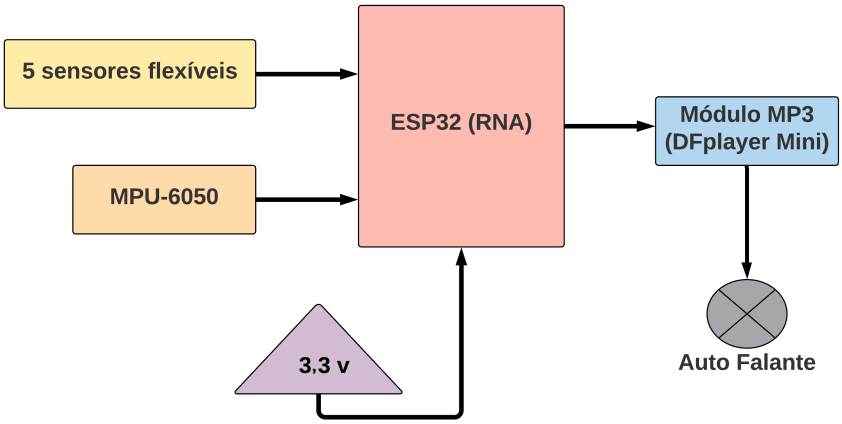


Figura 12 – Diagrama do circuito de atuação, no qual os sensores flexíveis, *MPU-6050* e alimentação são entradas e o módulo MP3 *DFplayer mini* saída. Fonte: Autor.

3.2 Calibração do Sensor *MPU-6050*

A funcionalidade do *MPU-6050* ao projeto é fornecer parâmetros do giroscópio e acelerômetro para identificar a posição espacial da mão do usuário. Tendo em vista a necessidade desta medição que terá um impacto nos resultados do projeto, percebe-se uma necessidade para executar a calibração do sensor.

De forma geral, a calibração do sensor é essencial para evitar erros de medição como viés de medida, acurácia e precisão. Para aplicar nesse projeto é necessário identificar a posição espacial dos dedos, consequentemente, identifica o sinal executado pelo usuário. Vale lembrar que para obter posição espacial do sensor, deve-se aplicar uma integral na velocidade e deste parâmetro uma integral na aceleração. Só pelo fato de executar uma integral, há uma perca de informação que impacta direto no valor mensurado, justificando que a calibração é indispensável (LAGE; SEGUNDO, 2016).

Para calibrar o sensor *MPU-6050* foi necessário utilizar o *arduino uno* como ferramenta auxiliar. O sensor foi calibrado com o *arduino uno* e testado na *ESP32-WROOM-32* para verificar a calibração, na qual utilizou os materiais apresentados no anexo B.

A necessidade de realizar a calibração do sensor *MPU-6050* de forma eficiente, impulsionou a escolha de um software que facilita a calibração, logo escolheu-se *processing 3D*. A figura 13 apresenta a interface do software. A princípio encontrou-se descalibrado devido os ângulos $psi = 172.82057^o$, $theta = 2.6807625^o$ e $phi = -178.9414^o$ estarem diferentes dos ângulos $psi = 180^o$, $theta = 0^o$ e $phi = 180^o$, nos quais esses últimos representam o sensor calibrado.

Figura 13 – O software *Processing 3D* apresenta os ângulos de referência do sensor *MPU-6050* que se demonstra descalibrado. Fonte: Autor.

Apesar de escolher o microcontrolador *ESP32-WROOM-32*, utilizou-se o *arduino uno* para auxiliar nesse procedimento, pois não havia biblioteca de calibração para a *ESP32-WROOM-32*. Diante desta adversidade optou-se pelo uso do *arduino uno* acompanhado da biblioteca padrão do protocolo serial I^2C de comunicação referente ao *MPU-6050*. A partir desta biblioteca, modificou-se alguns parâmetros e determinou-se os pinos para configurar as funções de SCL e SDA para assim estabelecer os requisitos da comunicação entre o microcontrolador e o sensor. Com auxílio desse algoritmo encontrou-se o endereço '$0x68$' da *MPU-6050*. A fim de saber o destino na comunicação do *MPU-6050* como *slave* entre *arduino uno* como *master*

No entanto, o *MPU-6050* não estava calibrado e enviava valores diferentes

do esperado. Então utilizando as bibliotecas i2cDEV.h que se encontra no apêndice A, *MPU6050.h* no apêndice B e o software *processing 3D*, obteve-se a calibração. Após carregar e executar o código implementado no microcontrolador *arduino uno*, houve uma adaptação nesse código para utilizar no microcontrolador *ESP32-WROOM-32* com a finalidade de fornece os valores do acelerômetro e giroscópio e assim verificar a calibração do microcontrolador. Logo, a calibração foi efetuada com sucesso como é demonstrado na figura 14.

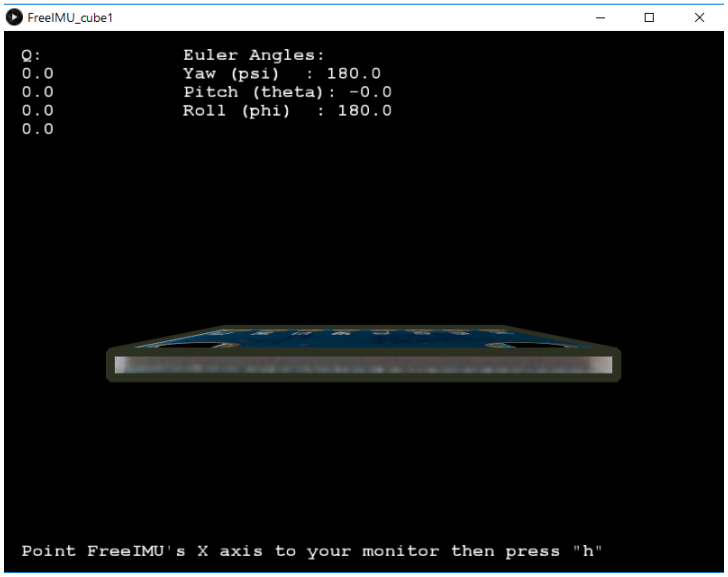


Figura 14 – O *software Processing 3D* apresenta os ângulos de referência do sensor *MPU-6050* que se demonstra calibrado. Fonte: Autor.

Com relação ao âmbito *hardware*, as figuras 15 apresenta as conexões do *MPU-6050* com *arduino uno*, nas quais (A) *Jumper* verde: Pino do SCL do *MPU-6050* conectado o pino A5 do *Arduino Uno*, (B) *Jumper* azul: Pino do SDA do *MPU-6050* conectado a porta A4 do *Arduino Uno* (C) *Jumper* vermelho: Pino VCC do *Sensor MPU-6050* conectado no pino VCC do *Arduino Uno* e (D)*Jumper* Preto: Pino GND do *MPU-6050* conectado no pino GND do *Arduino Uno*. Já a figura 16 apresenta as conexões do *MPU-6050* com o *ESP32-WROOM-32*, nas quais (P1) Pino VCC do *Sensor MPU-6050* conectado com o pino 3.3 volts da *ESP32-WROOM-32*, (P2) Pino GND do *Sensor MPU-6050* conectado com o pino GND da *ESP32-WROOM-32*, (P3) Pino SCL do *Sensor MPU-6050* conectado

ao pino GPIO22 da *ESP32-WROOM-32* e (P4) Pino SDA do *Sensor MPU-6050* conectado com o pino GPIO21 da *ESP32-WROOM-32*, respectivamente. E por fim, a figura 17 apresenta a posição espacial utilizada como referência para a satisfazer as premissas da calibração.

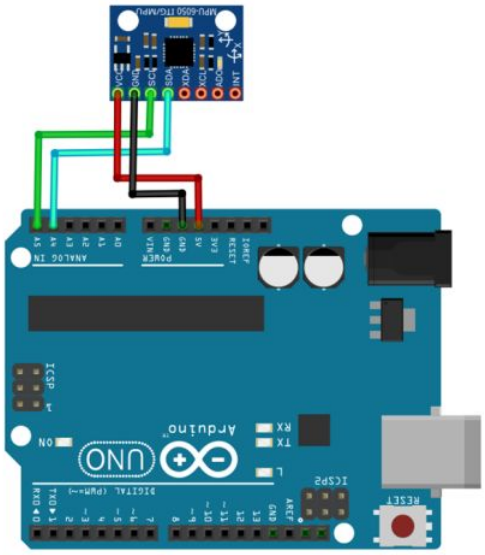


Figura 15 – Circuito de calibração do sensor *MPU-6050* com *arduino uno* . Fonte: Autor.

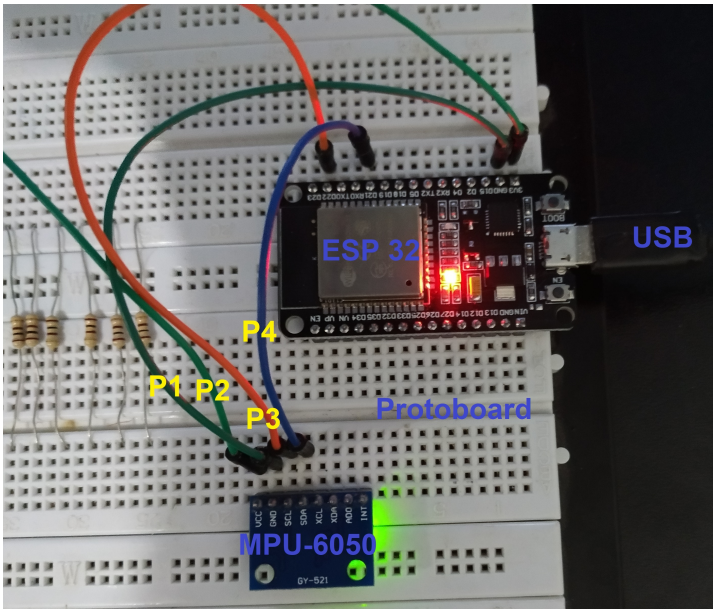


Figura 16 – Esquemático da Calibração do sensor *MPU-6050* com o *ESP32-WROOM-32*. Fonte: Autor.

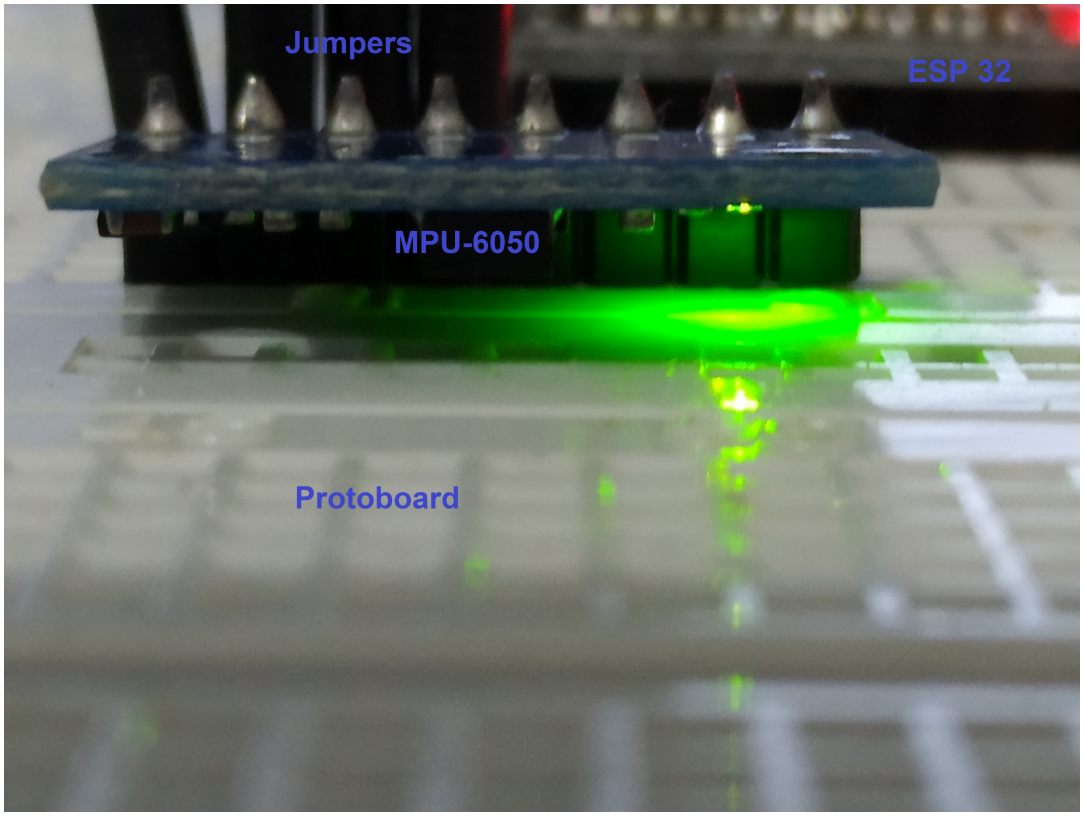


Figura 17 – Calibração do sensor *MPU-6050*: A figura apresenta uma visão lateral, que mostra a relação paralela entre a *protoboard* e o sensor para atingir os ângulos de calibração. Fonte: Autor.

3.3 Confecção do Sensor Flexível Resistivo

A leitura e interpretação dos movimentos é feita por meio de uma divisão de tensão de duas resistências, sendo a primeira de magnitude fixa e a segunda de magnitude variável proporcionalmente a flexão do sensor. Então, ao flexionar o sensor, esse varia a segunda resistência que retorna um valor de tensão proporcional a flexão do sensor. Com essa tensão pode-se inferir que o sensor está flexionado ou não. A figura 18 mostra o circuito de divisão de tensão no qual escolheu-se R1 e R3 com o valor 1 kω porque o sensor flexível resistivo tem o valor máximo de 1 kω, para que o divisor de tensão fornecer a maior faixar de variação facilitando a leitura do microcontrolador e da mesma forma foi feita a escolha dos resistores R2,R4 e R5.

Na sua construção utilizou-se os materiais descritos no anexo C. O procedimento de confecção será descrito a seguir e foi inspirado em (INSTRUCTABLES, 2016).

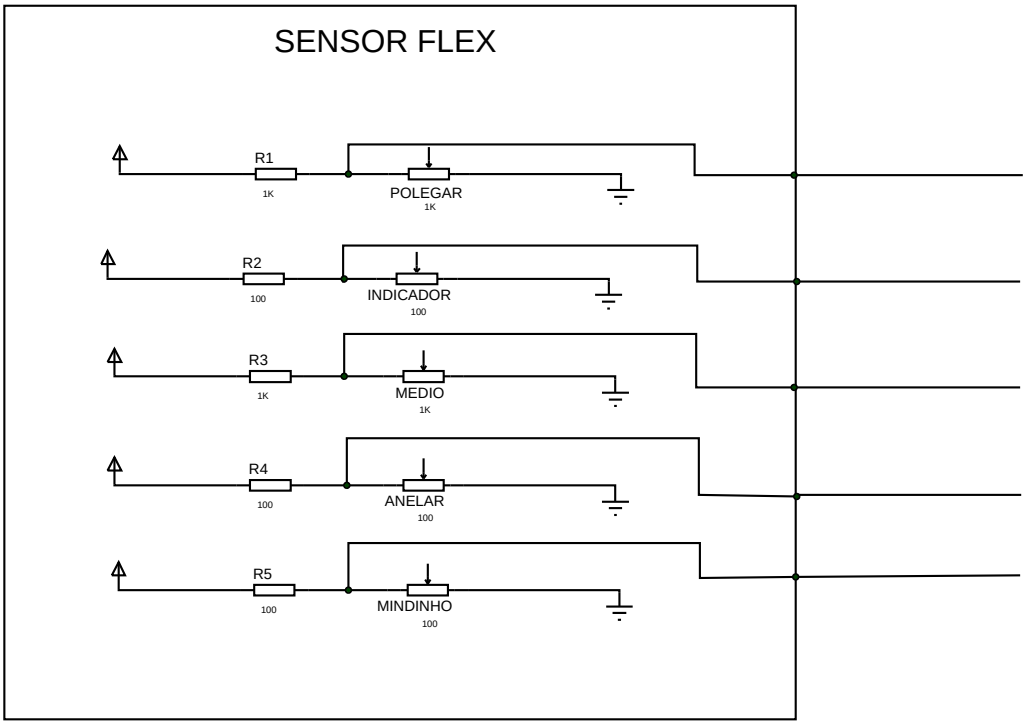


Figura 18 – Circuito do sensor Flex: São 5 divisores, um para cada dedo. Fonte: Autor.

Sabendo que o projeto exige um bom funcionamento desse sensor flexível resistivo e deve ser de baixo custo, iniciou-se um procedimento de confecção. Então, utilizou-se uma fita de cobre condutora de 15 cm x 2cm, que é muito utilizada para fazer a blindagem em guitarra. Conseguinte, pintou-se um trecho da folha A4 com as dimensões de 15 cm x 2 cm com lápis *faber castell* até criar uma resistência de folha, de acordo com a figura 19. Com o auxílio do multímetro verificou-se a resistência de folha, nesse caso a resistência obtida foi de $R_{folha} = 3{,}2\ \text{k}\Omega$.

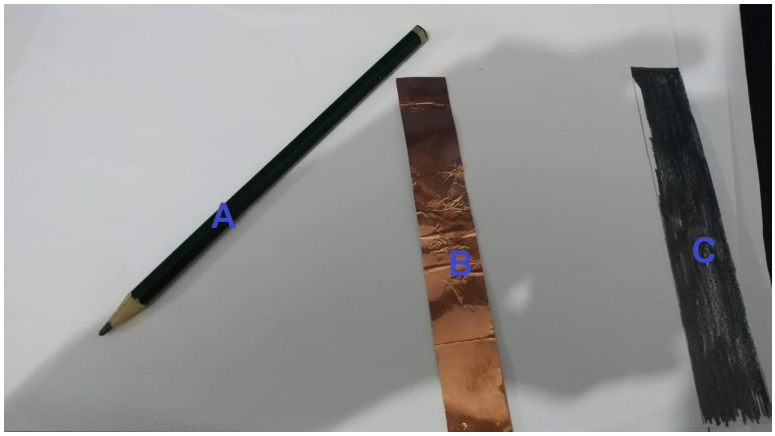


Figura 19 – Resistência de Folha: (A) lápis *Faber Castell* com graduação 9b, (B) Fita de cobre 15 cm x 2 cm e (C) Folha A4 15 cm x 2 cm sendo pintada por lápis. Fonte: Autor.

Em seguida, ao utilizar o esquadro, executou-se marcações de 15 cm x 0,5 cm na fita de cobre, de acordo com a figura 20, para que o sensor forneça dois terminais, a fim de apresentar uma resistência no final do procedimento. Com auxílio da tesoura cortou-se a fita de cobre em pedaços menores, como pode-se ver na figura 21, de acordo com as marcações feitas na figura 20. Fazendo 2 novas marcações no papel cartão, agora com dimensões 15 cm x 2 cm para funcionar como a camada interna de encapsulamento do sensor. Observe a figura 22 que mostra como será a montagem do sensor com a resistência de folha em contato com a fita de cobre e duas camadas de papel cartão para encapsular juntamente com a fita isolante.

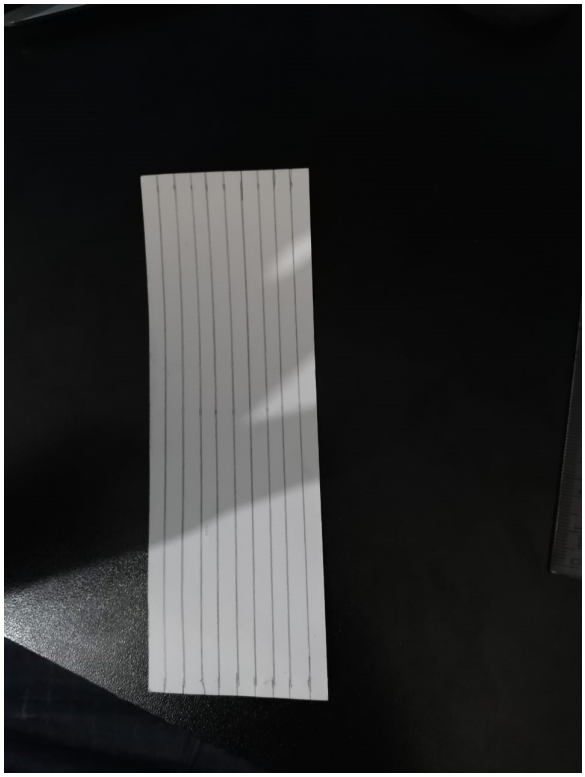


Figura 20 – Marcações no verso da Fita de Cobre para serem cortadas. Fonte: Autor.

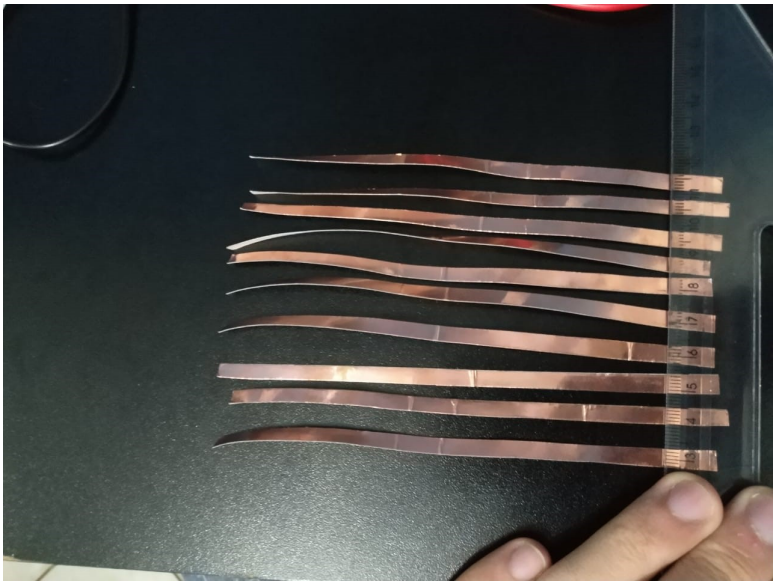


Figura 21 – Pedaços da fita de cobre cortados a fim de utilizar 2 pedaços para confecção de um sensor flexível resistivo. Fonte: Autor.

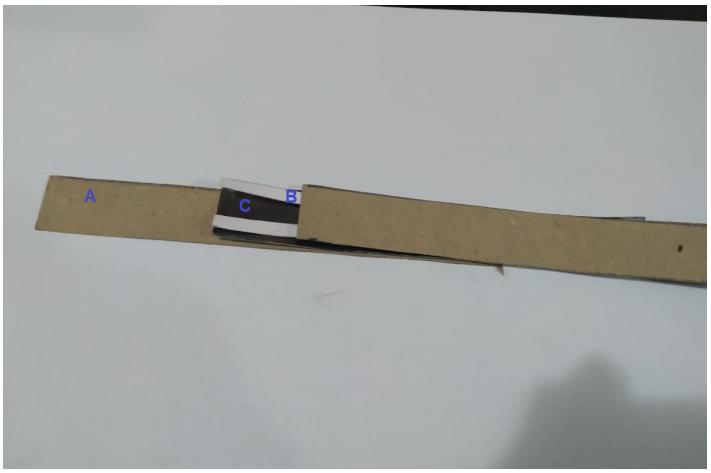


Figura 22 – Camadas do sensor: (A) papel cartão, (B) fita de cobre e (C) folha resistiva. Fonte: Autor.

Agora começa o procedimento de encapsular, no qual o sensor encapsulado tem uma camada exterior formada a partir da fita isolante, deixando as 2 extremidades de cobre livres. Logo, houve o ato de soldar os terminais livres da fita de cobre com os pedaços de *jumpers* para criar terminais mais fáceis de manusear. Em

seguida vedou-se a solda com duas camadas de papel cartão e fita isolante como mostra a figura 23. Com intuito de verificar o funcionamento do sensor, mensurou-se sem flexioná-lo e adquiriu uma resistência de 279 Ω e ao flexionar obteu-se a resistência de 70 Ω de acordo com a figura 24 validando o funcionamento do sensor.

Figura 23 – Sensor flexível resistivo confeccionado, nas quais as extremidades apresentam resistência em função da flexão. Fonte: Autor.

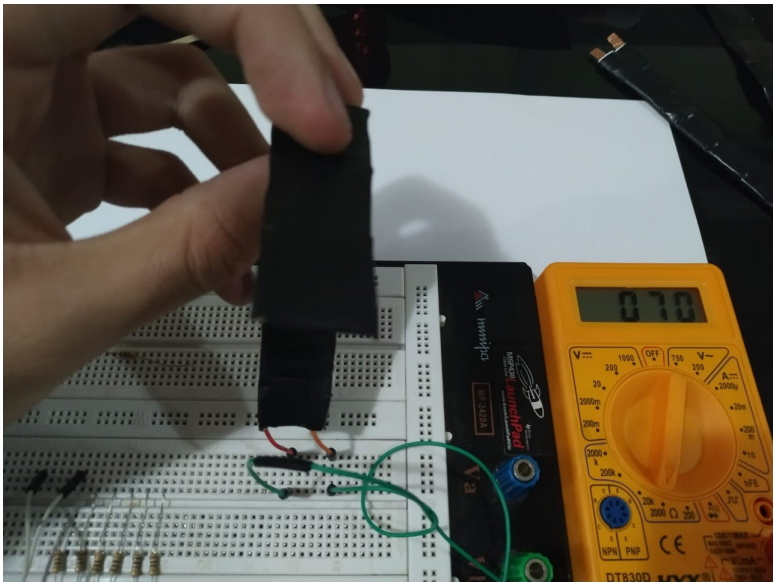


Figura 24 – Sensor flexionado com o valor da resistência apresentado. A utilização da *protoboard* auxiliou na mensuração. Fonte: Autor.

Após o passo de testar o funcionamento do sensor e tendo em vista a neces-

sidade do sensor tornar um produto, fez-se uma montagem com 5 sensores flexíveis em um cabo *flat* para assim serem integrados a uma luva. Em seguida testou-se os sensores juntos com auxílio do multímetro para verificar o bom funcionamento do conjunto de sensores de acordo com a figura 25.

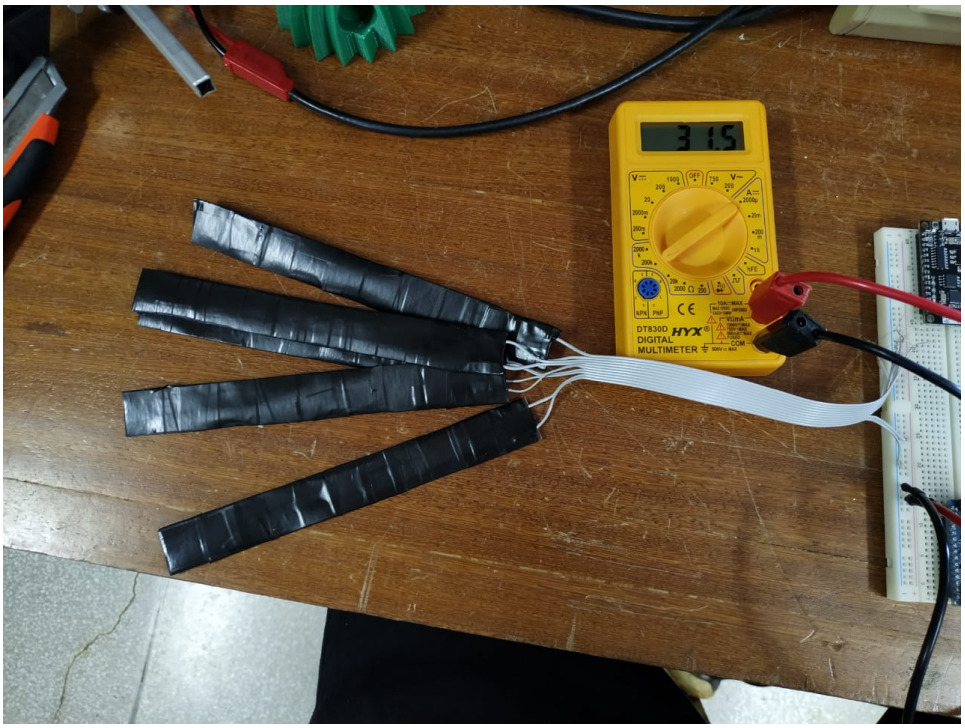


Figura 25 – Mensuração do sensor resistivo flexível com auxílio do multímetro. Fonte: Autor.

3.4 Integração da Luva com os sensores flexíveis

Sabe-se que a leitura e interpretação dos movimentos são feitas por meio de uma divisão de tensão. Para que a flexão do sensor ocorra ao mesmo tempo da flexão do dedo, optou-se por usar uma luva, a fim de satisfazer esta condição. Na sua construção utilizou-se os materiais descritos no anexo D.

Com o desejo de integrar a luva ao sensor, utilizou-se cianoacrilato que é a famosa super cola para fixar os sensores a uma luva de lã. Ao testar a luva integrada com os sensores flexíveis e com os dedos relaxados de acordo com a

figura 26, observou-se o bom funcionamento dos sensores. Após esse teste, foi feito a caracterização dos sensores flexíveis com a finalidade de avaliar o desempenho apresentado na seção 4.1.2.

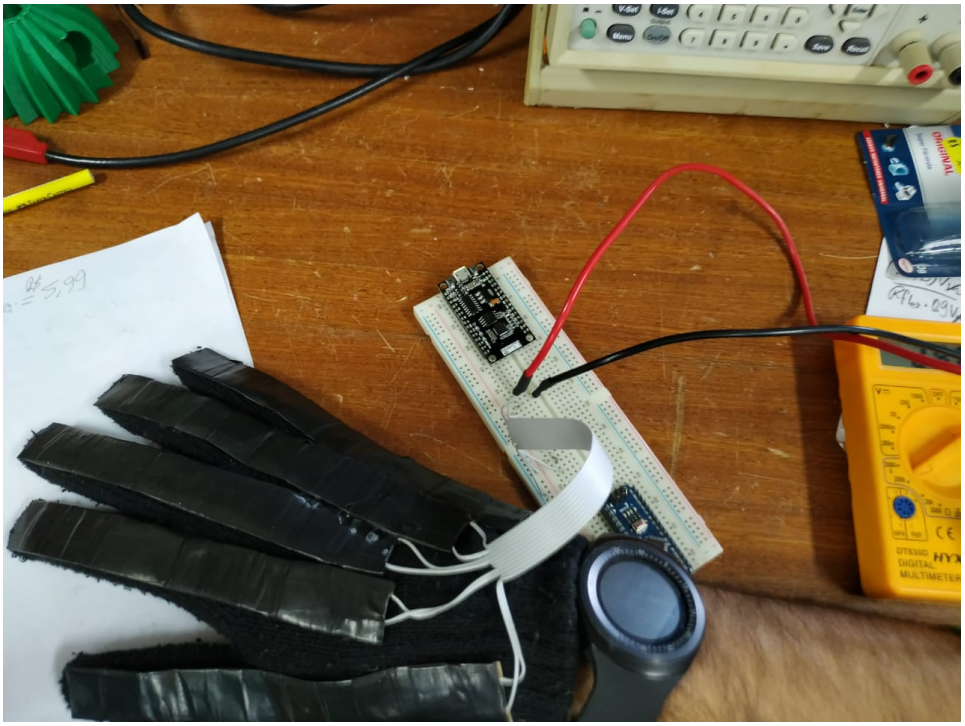


Figura 26 – Verificação do funcionamento da luva integrada com sensores flexíveis resistivos. Fonte: Autor.

3.5 Sistema de Áudio

A figura 27 mostra o *DFPlayer Mini* que é módulo MP3 pequeno e de baixo custo com uma saída simplificada para o alto-falante. O módulo pode ser usado como um módulo autônomo com bateria, alto-falante e botões de pressão conectados, ou usado em combinação com um *Arduino UNO* ou qualquer outro com a comunicação UART, por exemplo, *ESP32-WROOM-32*. O módulo apresenta decodificação rígida, que suporta formatos de áudio comuns: MP3, WAV e WMA. Além disso, também suporta sistema de arquivos: FAT16 e FAT32. Por meio de uma porta serial simples, pode executar o arquivo de música designado (DIGIKEY, 2019).

Ele suporta taxas de amostragem de 8/11.025/12/16/22.05/24/32/44.1/48 KHz, tem saída DACs de 24 bits, suporta cartões micro *SD card* de até 32GB nos formatos FAT16 e FAT32, reconhece até 100 pastas com 255 músicas cada uma, e tem também 30 níveis de volume e 6 formatos de equalização. E por fim, ele processa os sinais de áudio e transforma-os em ondas sonoras ao se conectar à auto-falantes de até 3 watts sem a necessidade de um amplificador, justificando a escolha (ARDUINO_E_CIA, 2017).

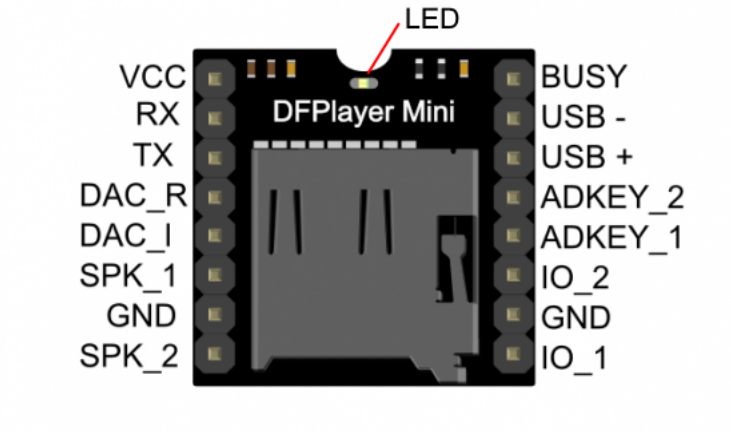


Figura 27 – Pinagem do *DFPlayer mini* . Fonte:(ARDUINO_E_CIA, 2017).

3.6 Montagem do circuito e testes dos algoritmos

Inicialmente fez-se testes em *protoborad* para validar o circuito projetado e em seguida desenvolveu-se algoritmos para a aquisição do *dataset*, treino da RNA e o algoritmo da estrutura da RNA com os pesos treinados que se encontram nos apêndices D, E e F respectivamente. Na sua construção utilizou-se os materiais descritos no anexo E.

Foi necessário a montagem do circuito para testar a aquisição de dados, que por conseguinte obteria o *dataset* que são os dados utilizados para treinar a RNA apresentado na figura 28. Após esse teste, o circuito funcionou perfeitamente, havendo a extração dos dados (*dataset*), onde o apêndice D apresenta o código utilizado para fazer a aquisição do *dataset* que futuramente fará o treinamento.

Após a montagem do circuito e o desenvolvimento desse código, foi carre-

gado no microcontrolador *ESP32-WROOM-32* o algorítimo de adquirir o *dataset*, no qual pode-se executar os sinais de LIBRAS, mantendo a mão imóvel por 10 minutos a fim de adquiri-los em uma taxa de 115200 $\frac{bits}{s}$, ou seja, 150 sinais adquiridos a cada segundo com a finalidade de construir o *dataset* de acordo com a figura 29. O código desenvolvido permitia armazenar os dados em 'data.txt' que posteriormente foi necessário transformar em outro tipo de documento 'data.csv' que é um documento separado por vírgula, muito utilizado em treino de RNA. Optou-se por transformar pelo fato da facilidade dos dados serem lidos na API escolhida.

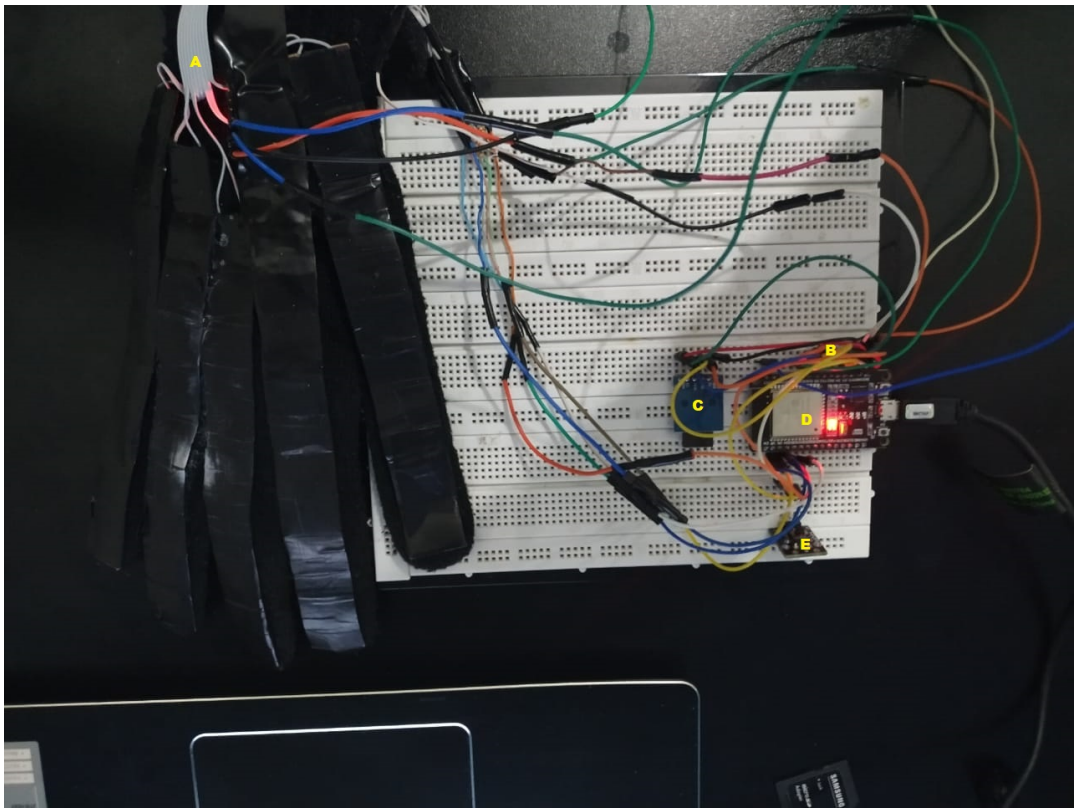


Figura 28 – Circuito de aquisição do *dataset*: (A) Sensor *Sensor MPU-6050*, (B) *Jumpers*, (C) Módulo micro *SD card*, (D) *ESP32-WROOM-32* e (E) *Push Button*. Fonte: Autor.

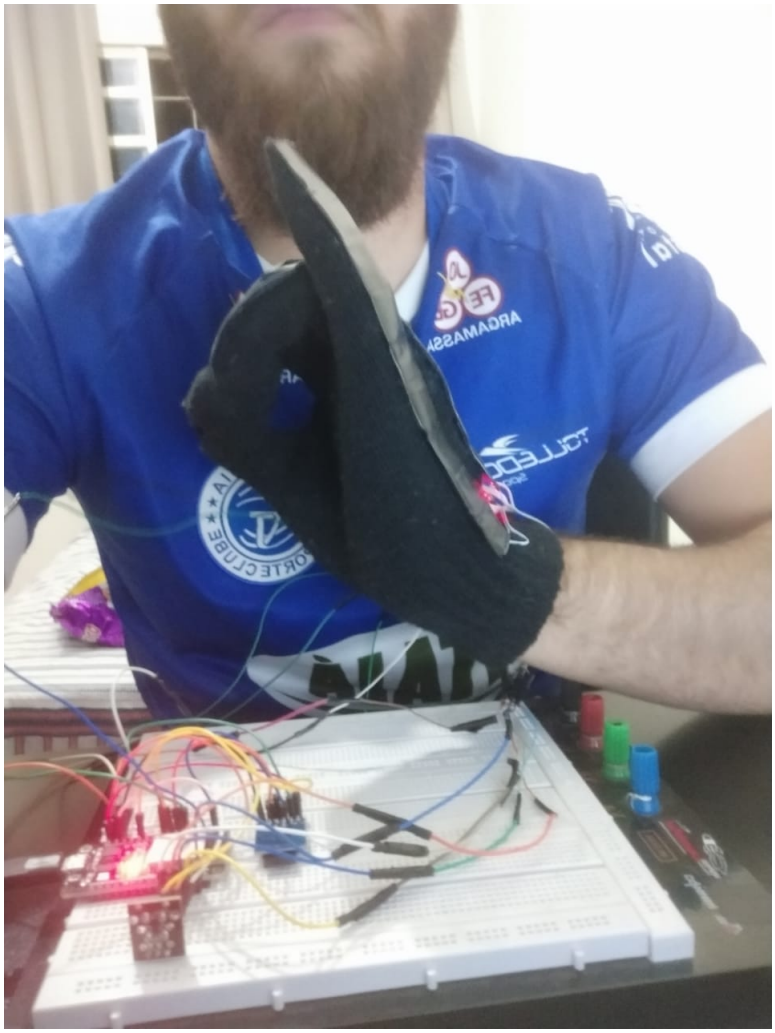


Figura 29 – Aquisição do *dataset*: Para isso, é necessário fazer o sinal de interesse e manter a mão imóvel na posição desejada por 10 minutos. Fonte: Autor.

API*Keras* foi escolhida para o projeto, pelo fato do desempenho nos treinos da RNA e a facilidade de programar. Ela se caracteriza como API porque executa um *framework tensorflow* que foi desenvolvido pelos pesquisadores da *google* a fim de facilitar o treinamento de RNA. Esta API é compatível com a linguagem *Python*. No desenvolvimento do algoritmo de treino da RNA utilizou-se o ambiente de desenvolvimento *Spyder* 3 que aceita a versão 3 do *Python*. No apêndice E apresenta o código de treinamento desenvolvido em API *Keras*, cujo esse código

permite a extração dos pesos treinados, que por conseguinte tem a função de caracterizar o modelo que classifica os 22 sinais de LIBRAS definido a partir de uma consulta técnica.

A arquitetura apresentada no treinamento pode ser vista na figura 30. Na qual definiu-se: 11 dados de entrada, pois eram os dados relacionado aos 3 eixos do acelerômetro, 3 eixos do giroscópio e 5 dos sensores flexíveis de cada dedo da mão; 2 camadas ocultas com 30 neurônios cada, nas quais a escolha se baseou pelo o desempenho constatado de forma empírica e 22 neurônios de saída que correspondem aos sinais a serem traduzidos que podem ser conferido na tabela 4.

Tabela 4 – Relação de classes com seus respectivos Sinais traduzidos pela LuTLI. Fonte: Autor.

Número da Classe	**Sinal**
1	Obrigado
2	U
3	N
4	B
5	Oi
6	Tudo bem?
7	Tchau!
8	Deus
9	Abraço
10	Desculpa
11	Brincar
12	Barato
13	Café
14	Dormir
15	Futuro
16	Fácil
17	Dois
18	Música
19	Igual
20	Inteligente
21	Luva
22	Sinal

A função de ativação nas camadas ocultas é a *ReLU*, porque ela transforma os valores negativos de entrada em zeros, caso contrário mantém o valor da entrada, permitindo a rapidez em treinar. Já a *softmax* foi escolhida na saída pelo fato dela se comportar melhor do que as outras funções de ativação quando se trata de uma RNA multiclasses, que neste caso, são 22 classes (saídas). Vale lembra que a saída de toda função de ativação é a probabilidade da classificação ser efetiva.

A normalização foi necessária para que o treinamento dos pesos fosse mais rápido e mais efetiva. A escolha foi dividir por 2500 porque o maior dado é 2500. Na metodologia de treino, foi separado 75% do *dataset* para treino e 25% para teste, na qual pode-se gerar uma matriz de confusão que constatou um ótimo funcionamento do modelo da RNA denominada Luva Tradutora de LIBRAS Inteligente (LUTLI) demonstrada na figura 30 que será comentada em resultados com auxílio da figura 51.

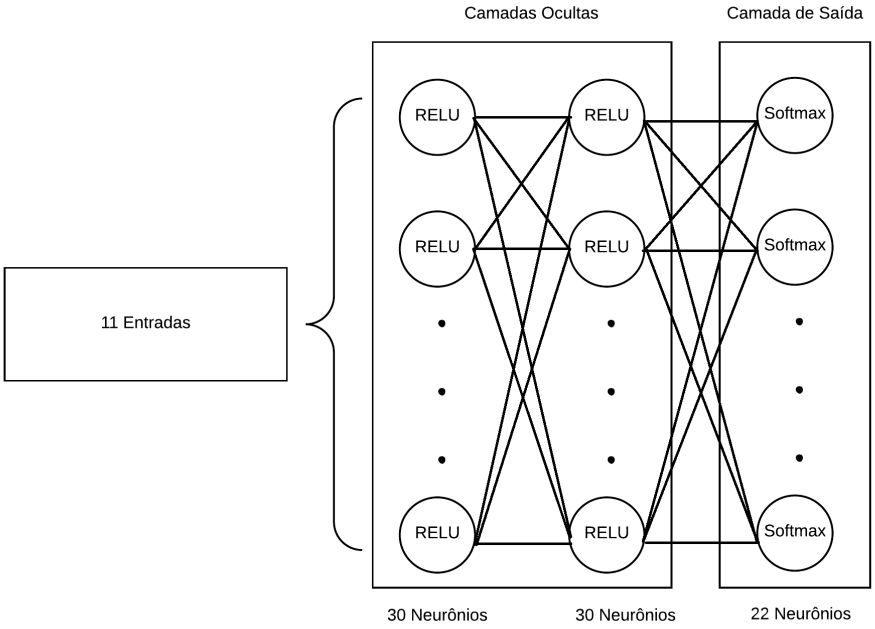


Figura 30 – Estrutura da RNA da LUTLI, na qual apresenta as 11 entradas, as camadas ocultas com a função de ativação RELU e a camada de saída com a função de ativação softmax com seus respectivos números de neurônios em cada camada. Fonte: Autor.

3.6.1 Modelagem Matemática da RNA LuTLI

- Entrada da rede:

$$S_j = \sum_{i=1}^{11} W_{ij} X_i \tag{3.1}$$

- Função de ativação escolhida nessa camada foi a $ReLU$:

$$h_j = f(S_j) \tag{3.2}$$

- Onde a função $ReLU$ é:

$$f(x) = \begin{cases} x, & \text{se x} > 0 \\ 0, & \text{se x} < 0 \end{cases} \tag{3.3}$$

- Entrada na segunda camada oculta:

$$S_k = \sum_{i=1}^{30} W_{jk} f(S_j) \tag{3.4}$$

- Função de ativação na segunda camada oculta:

$$h_k = f(S_k) \tag{3.5}$$

- Entrada na camada de saída:

$$S_p = \sum_{i=1}^{30} W_{kp} f(S_k) \tag{3.6}$$

- Saída prevista 'Y_p' pela Rede Neural, passa pela função de ativação *softmax*:

$$Y_p = G(S_p) \tag{3.7}$$

- Onde a função *softmax* é:

$$G(x) = \frac{e^x}{\sum_{k=1}^{N} e^{x_k}} \tag{3.8}$$

Fase da Retropropagação

- Função erro:

$$e_l = Y_e - Y_p \tag{3.9}$$

- Onde Y_e é a a saída esperada e Y_p é a saída da RNA.

Função Custo ou função perda

- A função custo com a função erro como variável de entrada forma o que conhece como MSE.

$$J = \frac{1}{2}\sum_{l=1}^{N} e_l^2 \tag{3.10}$$

- Gradiente da camada de saída:

$$\nabla W_{kp} = \frac{\partial J}{\partial W_{kp}} = \frac{\partial J}{\partial e_l}\frac{\partial e_l}{\partial W_{kp}} = \frac{\partial \frac{1}{2}\sum_{l=1}^{11} e_l^2}{\partial e_l}\frac{\partial Y_e - Y_p}{\partial Y_p}\frac{\partial Y_p}{\partial W_{kp}} \tag{3.11}$$

- Sabendo que Y_p= $G(S_p)$:

$$\nabla W_{kp} = -e_l\frac{\partial G(S_p)}{\partial S_p}\frac{\partial S_p}{\partial W_{kp}} = -e_l\frac{\partial G(S_p)}{\partial S_p}\frac{\partial \sum_{i=1}^{30} W_{kp} f(S_k)}{\partial W_{kp}} \tag{3.12}$$

$$\nabla W_{kp} = -e_l\frac{\partial G(S_p)}{\partial S_p} f(S_k) \tag{3.13}$$

- Gradiente da segunda camada oculta:

$$\nabla W_{jk} = \frac{\partial J}{\partial W_{jk}} = \frac{\partial J}{\partial e_l}\frac{\partial e_l}{\partial W_{jk}} = \frac{\partial \frac{1}{2}\sum_{l=1}^{11} e_l^2}{\partial e_l}\frac{\partial Y_e - Y_p}{\partial Y_p}\frac{\partial Y_p}{\partial W_{jk}} \tag{3.14}$$

$$\nabla W_{jk} = -e_l\frac{\partial G(S_p)}{\partial S_p}\frac{\partial S_p}{\partial W_{jk}} = -e_l\frac{\partial G(S_p)}{\partial S_p}\frac{\partial \sum_{i=1}^{30} W_{kp} f(S_k)}{\partial f(S_k)}\frac{\partial f(S_k)}{\partial S_k}\frac{\partial (S_k)}{\partial W_{jk}} \tag{3.15}$$

$$\nabla W_{jk} = -e_l\frac{\partial G(S_p)}{\partial S_p}\sum_{i=1}^{30} W_{kp}\frac{\partial f(S_k)}{\partial S_k}\frac{\partial \sum_{i=1}^{30} W_{jk} f(S_j)}{\partial W_{jk}} \tag{3.16}$$

$$\nabla W_{jk} = -e_l\frac{\partial G(S_p)}{\partial S_p}\sum_{i=1}^{30} W_{kp}\frac{\partial f(S_k)}{S_k} f(S_j) \tag{3.17}$$

- Gradiente da primeira camada oculta:

$$\nabla W_{ij} = \frac{\partial J}{\partial W_{ij}} = \frac{\partial J}{\partial e_l}\frac{\partial e_l}{\partial W_{ij}} = \frac{\partial \frac{1}{2}\sum_{l=1}^{11} e_l^2}{\partial e_l}\frac{\partial Y_e - Y_p}{\partial Y_p}\frac{\partial Y_p}{\partial W_{ij}} \tag{3.18}$$

$$\nabla W_{ij} = -e_l \frac{\partial G(S_p)}{\partial S_p}\frac{\partial S_p}{\partial W_{ij}} = -e_l \frac{\partial G(S_p)}{\partial S_p}\frac{\partial \sum_{i=1}^{30} W_{kp} f(S_k)}{\partial f(S_k)}\frac{\partial f(S_k)}{\partial S_k}\frac{\partial (S_k)}{\partial W_{ij}} \tag{3.19}$$

$$\nabla W_{ij} = -e_l \frac{\partial G(S_p)}{\partial S_p}\sum_{i=1}^{30} W_{kp}\frac{\partial f(S_k)}{\partial S_k}\frac{\partial \sum_{i=1}^{30} W_{jk} f(S_j)}{\partial f(S_j)}\frac{\partial f(S_j)}{\partial S_j}\frac{\partial S_j}{\partial W_{ij}} \tag{3.20}$$

$$\nabla W_{ij} = -e_l \frac{\partial G(S_p)}{\partial S_p}\sum_{i=1}^{30} W_{kp}\frac{\partial f(S_k)}{\partial S_k}\sum_{i=1}^{30} W_{jk}\frac{\partial f(S_j)}{\partial S_j}\frac{\partial \sum_{i=1}^{11} W_{ij} X_i}{\partial W_{ij}} \tag{3.21}$$

$$\nabla W_{ij} = -e_l \frac{\partial G(S_p)}{\partial S_p}\sum_{i=1}^{30} W_{kp}\frac{\partial f(S_k)}{\partial S_k}\sum_{i=1}^{30} W_{jk}\frac{\partial f(S_j)}{\partial S_j} X_i \tag{3.22}$$

- Utilizou-se o otimizador adam para atualizar os pesos.
- Adam é configurado com a taxa de aprendizagem = 0,001; β_1=0,9; β_2=0,999 e $\epsilon = e^{-7}$.
- Utilizou-se *batch size* igual à 10.
- O processo da atualização dos pesos foi repetido por 70 vezes, ou seja, 70 épocas.

No apêndice F apresenta o código da estrutura da rede neural (algoritmo de atuação) da figura 30 com o pesos treinados pelo código do apêndice E. No entanto, esse código tem biblioteca que processa o áudio, na qual pode-se exibir o som do sinal classificado pela RNA treinada. Além disso, habilitou-se as portas de entradas para os sensores flexíveis e o *sensor MPU-6050*. Após carregar o código na

ESP32-WROOM-32 pode-se confirmar o êxito deste trabalho, foi possível traduzir todos os 22 sinais.

Ao utilizar o produto, verificou-se uma recomendação de usabilidade. Toda vez que for fazer um sinal, clica no botão de *reset*, logo em seguida faça o sinal na posição que foi feita a aquisição do *dataset*. Após isso, escutará um áudio com a tradução do sinal feito.

3.7 Confecção da Placa de Circuito Impresso (PCB)

Após validar o circuito em *protoboard*, utilizou-se o *software proteus* para projetar a PCB e assim confecciona-la. Nela será possível reunir e conectar todos os componentes em uma área pequena em relação ao tamanho do circuito testado em *protoboard* e além disto será viável para o usuário utilizar o produto no antebraço e na mão. Na sua construção utilizou-se os materiais descritos no anexo F.

O circuito eletrônico geral da LuTLI projetado no *software proteus* pode ser visto no apêndice G, no qual mostra as disposições e conexões dos componentes utilizados na confecção. De forma mais restringida, o circuito utilizado para adquirir os dados dos sensores flexíveis é mostrado na figura 18, onde foi utilizado um divisor de tensão. Esse divisor irá fornecer sinais de tensão que dependem do valores de resistência dos sensores flexíveis. Foi necessário fazer esse divisor pelo fato do microcontrolador ler em sinais de tensão e não de resistência.

O *layout* da PCB pode ser visto na figura 31. As linhas que estão destacas em azul representam as trilhas da parte superior da placa e as linhas em vermelho representam as trilhas na parte inferior da placa caracterizando-a dupla face. A opção por dupla face foi necessária pelo fato de que teria que ter uma placa pequena e havia muitas conexões. Como não havia biblioteca para o sensores utilizados, logo, o autor criou cada componente no *software proteus*, exceto os resistores. A previsão da placa foi gerada pelo *software proteus*, na qual pode ser visualizada na figura 32 . Nela apresenta os furos e a organização das trilhas tanto na parte superior quanto na inferior.

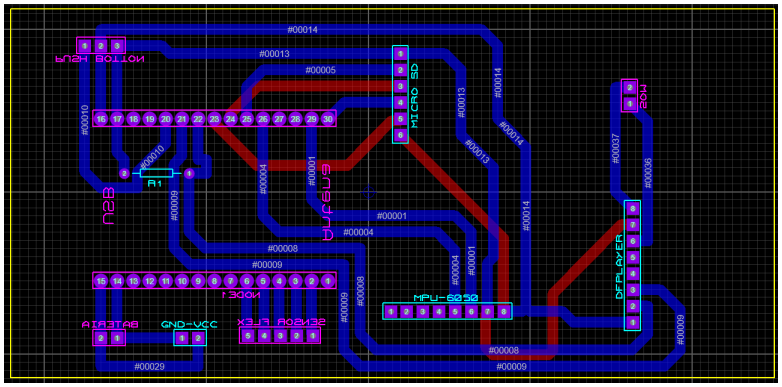
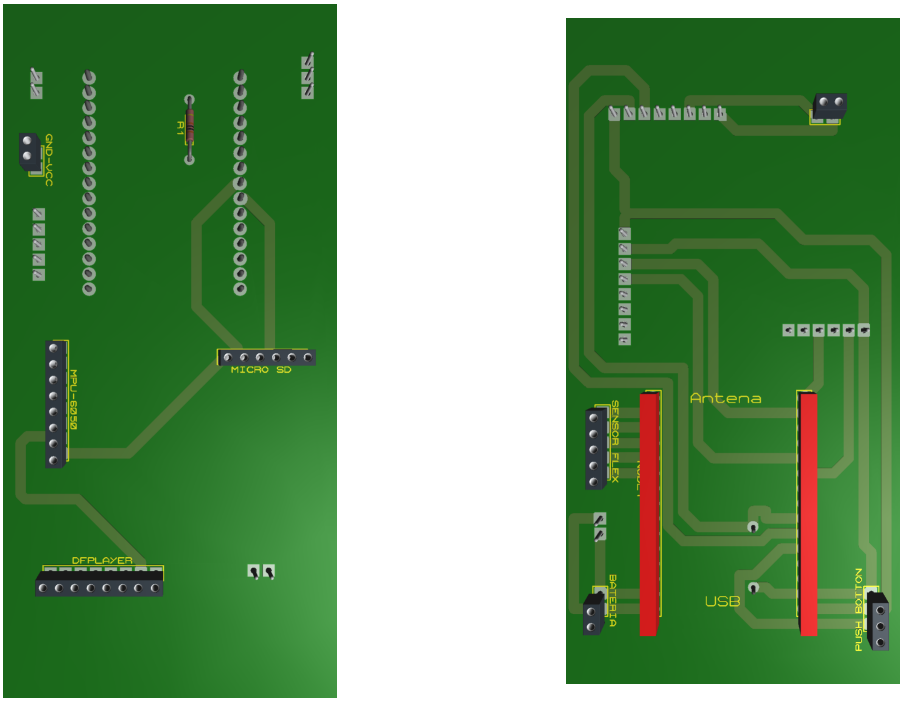


Figura 31 – Layout da PCB com a conexões entre os componentes por meio das trilhas. Fonte: Autor.



(a) visão inferior

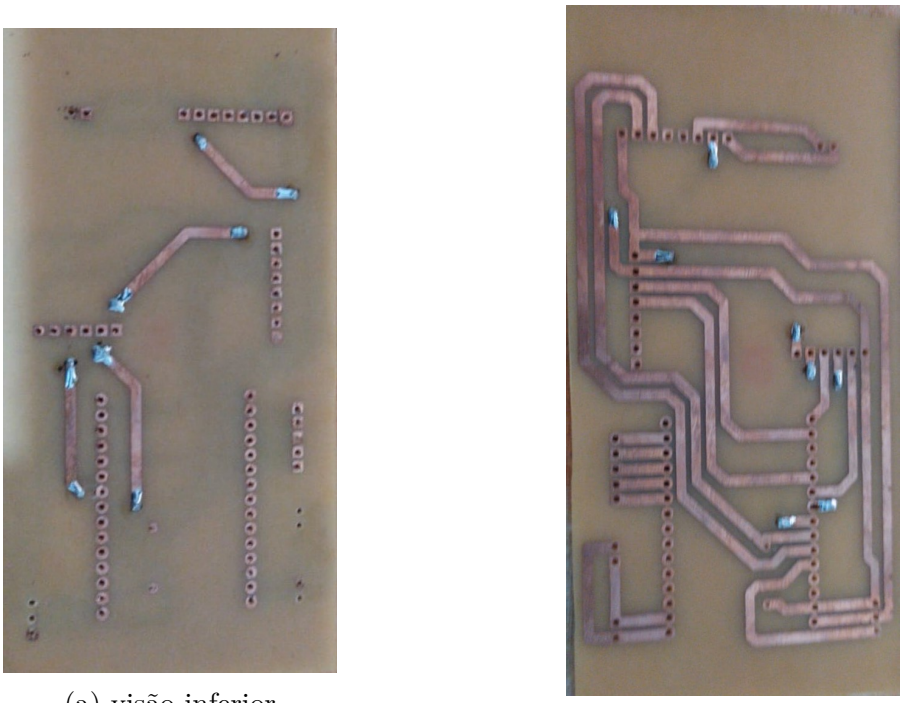
(b) visão superior

Figura 32 – Prévia da PCB no *software proteus* em formato 3D com animação dos componentes soldados na placa. Fonte: Autor.

O circuito da LuTLI na função aquisição do *dataset* pode ser visualizado na figura 11 que mostra as conexões de forma resumida, cujo os 5 sensores flexíveis,

alimentação e *sensor MPU-6050* são entradas do sistema e o módulo *SD card* é a saída. Já o circuito da LuTLI na função atuação, pode ser visualizado na figura 12 que mostra as conexões de forma resumida, cujo os 5 sensores flexíveis, alimentação e *sensor MPU-6050* são entradas do sistema e o módulo MP3 juntamente com o auto-falante é a saída.

A placa confeccionada pode ser vista na figura 33, ela é uma placa híbrida porque contém o circuito de aquisição de dados e o circuito de atuação. Logo, a fase de aquisição de dados e a de atuação (funcionamento do produto) estão contidas na placa confeccionada. É importante ressaltar que foi necessário inserir pingos de solda para conectar as trilhas da parte inferior com as trilhas da parte superior.



(a) visão inferior

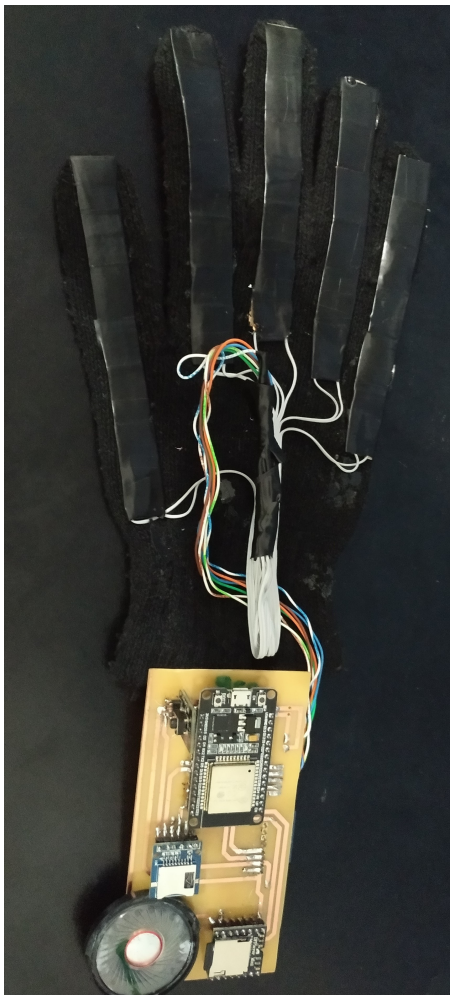
(b) visão superior

Figura 33 – Placa confeccionada com os pingos de solda conectando a trilha da parte superior com a parte inferior. Fonte: Autor.

A placa confeccionada soldada com os sensores e módulos do sistema de aquisição do *dataset* e o sistema de atuação podem ser vistos na figura 33. Já o encapsulamento da luva é mostrado na figura 35 que é composta por borracha

poliamida e velcro, ambos costurados de forma que pudesse englobar a placa e fixar no braço do usuário.

(a) visão inferior



(b) visão superior

Figura 34 – Placa confeccionada soldada com os componentes e integrada com a luva. Fonte: Autor.

(a) visão inferior

(b) visão superior

Figura 35 – Encapsulamento confeccionado: Material (borracha de poliamida) costurado com o velcro. Fonte: Autor.

A LuTLI pode ser visualizada na figura 36, onde esta foi utilizada para aquisição final do *dataset* usando o algoritmo D. Para treinar usou-se o algoritmo E em *python 3*, logo em seguida, utilizou-se o algoritmo de atuação com os pesos treinados do passo anterior na *ESP32-WROOM-32* usando o algoritmo F. Isso foi necessário porque o sensor *MPU-6050* mudou de posição em relação ao teste feito em *protoborad* e o sistema da RNA anterior só identifica os sinais que foram treinados na mesma posição anterior, no entanto, mudou-se a posição devido a integração dos sensores à luva. Todos os 22 sinais utilizados como *dataset* para o treino da RNA podem ser visualizado na figura 50. E como consequência, são os sinais traduzidos pelo novo modelo da RNA da LuTLI.

Figura 36 – Luva Tradutora de Libras Inteligente (LuTLI). Fonte: Autor.

4 Resultados

Esse capítulo trata-se da descrição dos resultados obtidos em testes de bancada. Nesse sentido, apresenta-se os resultados dos testes de melhor camada do sensor flexível, melhor fita de cobre, melhor distribuição da resistência de folha e a integração do sistema.

Pode-se conferir o preço dos materiais utilizados para a construção da luva LuTLI na tabela 4 em anexo G . O produto foi cotado com os materiais encontrados no brasil, com exceção do microcontrolador *ESP32-WROOM-32* que foi importado da China em 17/02/2018.

4.1 Resultados dos Sensores Flexíveis Resistivos

Todos os sensores flexíveis analisados nessa seção foram confeccionados de acordo com a seção 3.3. A princípio foram testados 3 tipos de camadas para encapsular os sensores: látex, fita elástica e papel cartão. Logo após, foi feita a caracterização para avaliar se os sensores confeccionados atende as necessidades do projeto.

4.1.1 Comparando os sensores flexíveis Resistivos

Foram escolhidos 3 camada diferentes para encapsular o sensor flexível resistivo: O látex por ser extremamente elásticos entre os 3 , a fita elástica por ser confortável ao usuário e o papel cartão por ser um material barato e duro em comparação aos outros dois materiais. No entanto, ao fazer os testes desses sensores, percebeu-se que os sensores que continham o látex e a fita elástica demonstrados nas figuras 37 e 39 respectivamente, comprometiam a leitura da resistência por serem materiais extremamente dúctil comparado ao papel cartão, cujo teve um bom desempenho informando os valores de resistência sem comprometer o funcionamento do sensor flexível resistivo apresentado na figura 22 .

Figura 37 – Materiais com o sensor flexível de látex: (A) material látex e (B) folha resistiva. Fonte: Autor.

Figura 38 – Camada de fita elástica: (A) material elástico e (B) folha resistiva. Fonte: Autor.

Houve uma comparação de 2 tipos de condutores para confecção do sensor flexível resistivo. Escolheu-se uma fita de cobre com uma espessura maior do que a outra demonstrada na figura 39. Esse teste está preocupado com o conforto do usuário e se a diferença da espessura entre elas poderia afetar a estabilidade da resistência.

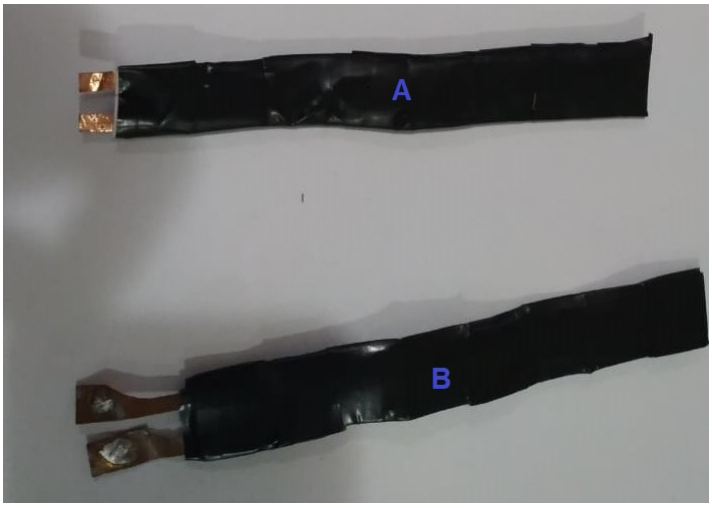


Figura 39 – Teste da fita de cobre: (A) Sensor que contém fita com a espessura de 0,074 milímetros e (B) Sensor que contém fita com a espessura de 0,500 milímetros . Fonte: Autor.

A fita de cobre de maior espessura teve quase o mesmo desempenho da outra em relação a estabilidade da resistência. No entanto, quanto maior a espessura da fita de cobre mais o sensor causa desconforto para o usuário, machucando as juntas da mão. Por esse motivo a fita de menor espessura foi escolhida.

A fim de encontrar o melhor lápis para a confecção do sensor flexível, adotou-se um método empírico testando uma grande variedade de lápis. A princípio pintou-se a folha em determinadas regiões separadas pelo tipo de lápis, com o auxílio do multímetro mensurou-se a resistência de folha e percebeu-se que os lápis forneciam resistência de 2 MΩ variando até 100 Ω.

Na confecção da luva percebeu-se também que quanto maior a resistência melhor será o sensor flexível resistivo, porque se a resistência tem uma magnitude alta, logo, terá um resolução alta que facilita na leitura do microcontrolador. De acordo com a figura 40 os lápis (E) e (G) tiveram bons resultados alcançando resistências próximas de 700 kΩ, (K) e (L) tiverem excelentes resultados alcançando até 2 MΩ e o resto abaixo do desejado que seria 500 kΩ. Logo, escolheu o lápis (K) e (L). No entanto os lápis com resistência baixa tem um papel importante para reduzir a resistência dos lápis de alta resistência de folha, porque assim pode-se

dosar e alcançar a resistência desejada.

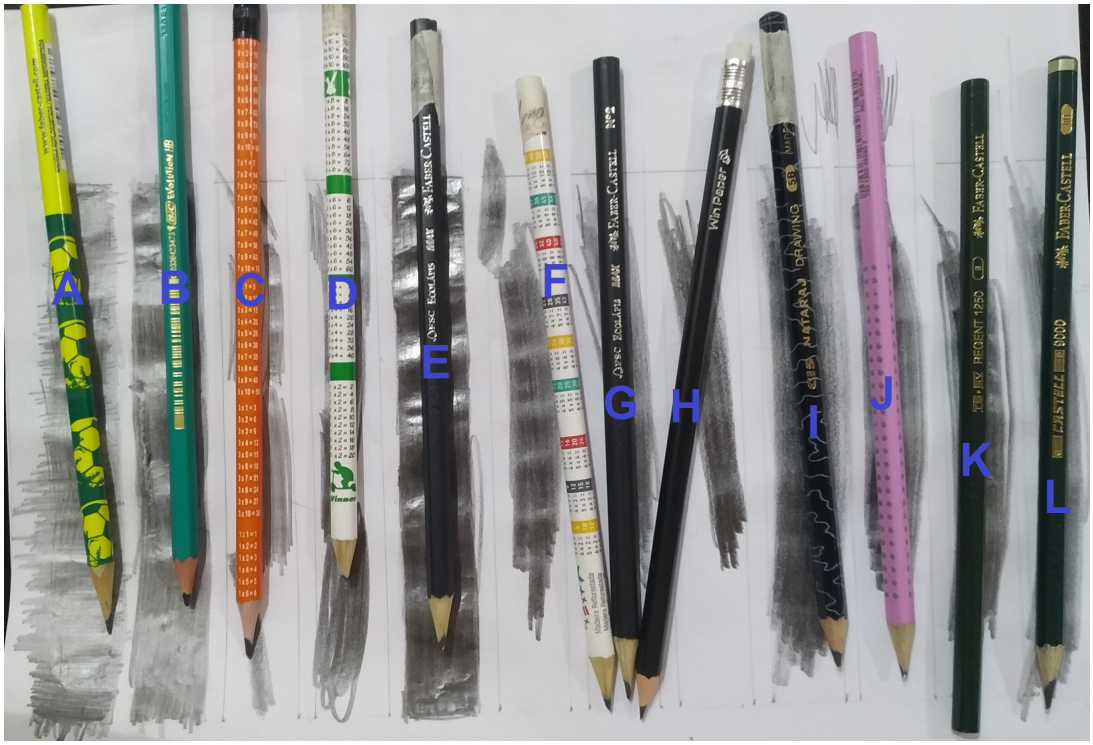


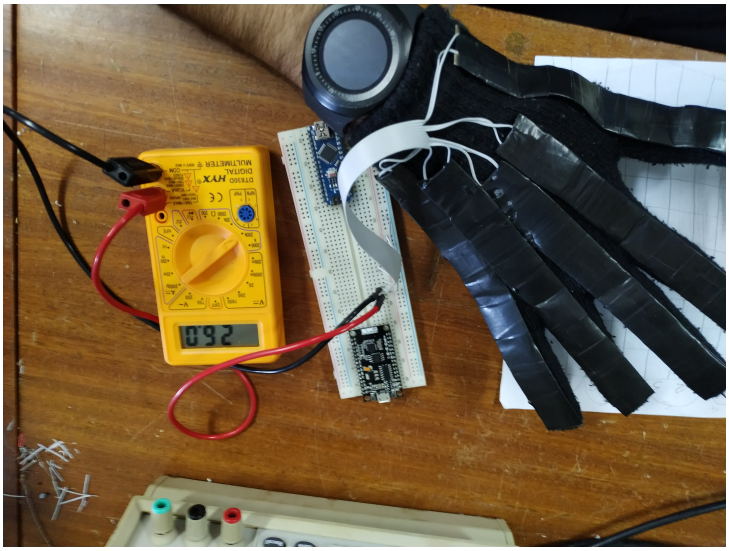
Figura 40 – Teste dos seguintes lápis: (A) *Faber Castell* Eco, (B) *Bic Evolution HB*, (C) Leo Leo tabuada, (D) *Winner* tabuada, (E)*Faber Castell* Ecolápis MAX cilíndrica, (F)*Faber Castell* tabuada, (G)*Faber Castell* Eco lápis MAX, (H) *Win Paper*, (I) *CIS NATARAJ* 5B, (J) *Faber Castell* Eco GRIP 2001, (K) *Faber Castell REGENT 1250* e (L)*Faber Castell* graduação 8B. Fonte: Autor.

4.1.2 Caracterização dos sensores flexíveis Resistivos

A caracterização constatou que sensor respondeu bem as 3 variações de flexão. Em todos os dedos houve regiões que os 3 tipos de sinais foram totalmente separados, apenas o dedo polegar que não atingiu esse objetivo. No entanto, após as aquisições do *dataset*, treino da RNA e extração dos pesos para carregar a estrutura da RNA no microcontrolador *ESP32-WROOM-32*, percebeu-se que o polegar não influenciou na classificação dos 22 sinais.

Na caracterização utilizou-se 3 posições dos dedos da mão com o intuito de

verificar o comportamento dos sensores flexíveis, fez-se 15 experimentos para casa posição e mensurou-se o valor da resistência em cada experimento. A primeira foi com os dedos da mão relaxados ou esticados, após isto, com o dedos da mão parcialmente dobrados ou quase dobrados e por fim com os dedos da mão totalmente dobrado. A luva esteve totalmente esticada de acordo com a figura 41a, parcialmente dobrada de acordo com a figura 41b e totalmente dobrada de acordo com a figura 41c.



(a) Dedos esticados

(b) Dedos parcialmente dobrados

(c) Dedos totalmente dobrados

Figura 41 – 3 formas de posicionar os dedos para obter os valor da resistência em cada experimento. Fonte: Autor.

Após fazer cada uma das 3 posições, obteu-se os resultados. O resultado da luva totalmente esticada é apresentado na figura 42, parcialmente dobrada é apresentado na figura 43 e totalmente dobrada é apresentado na figura 44. O intuito destes resultados é verificar a variação de resistência associado a flexão do dedo, ou seja, saber o limiar do dedo parcialmente dobrado para que não seja confundido com o dedo totalmente dobrado ou dedo esticado.

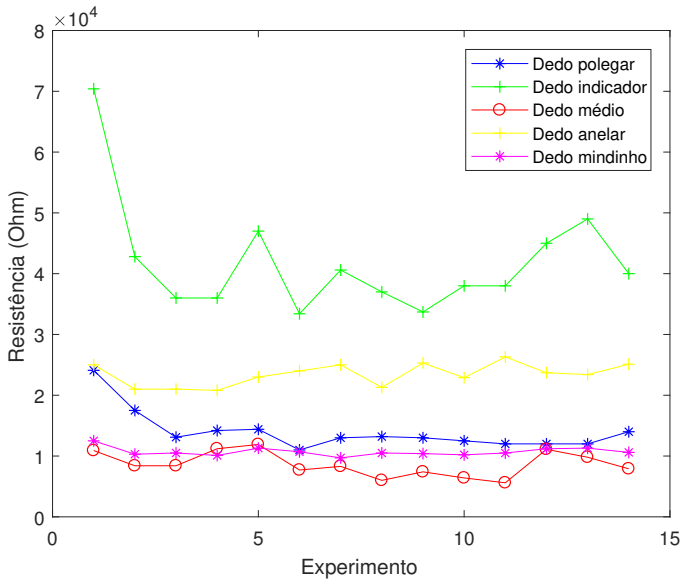


Figura 42 – Comparando os 5 dedos esticados, no qual o eixo das abcissas indica a quantidade de experimento e as ordenadas indica o magnitude da resistência referente ao experimento respectivo. Fonte: Autor.

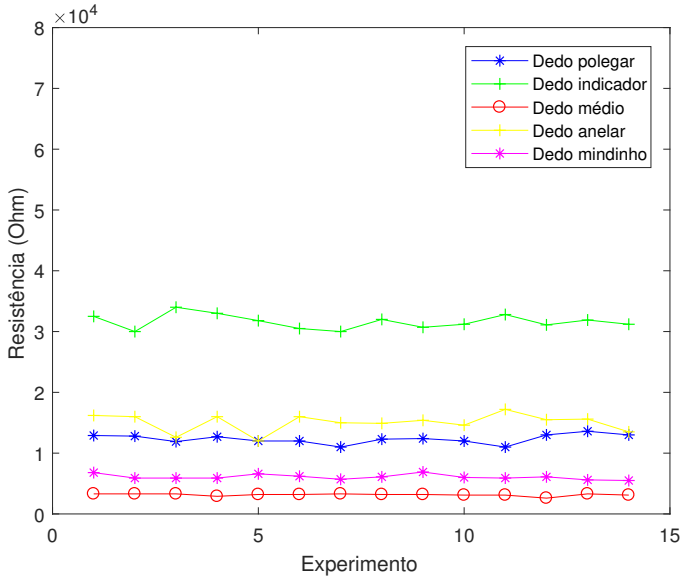


Figura 43 – Comparando os 5 dedos parcialmente dobrados, no qual o eixo das abcissas indica a quantidade de experimento e as ordenadas indica o magnitude da resistência referente ao experimento respectivo. Fonte: Autor.

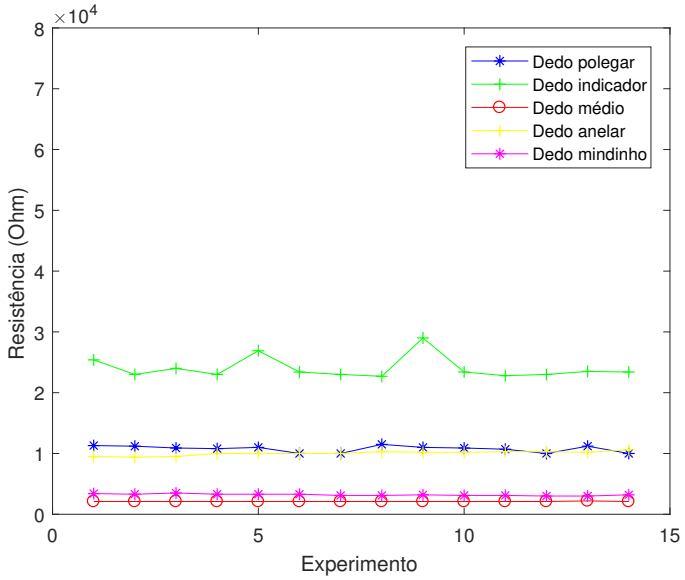


Figura 44 – Comparando os 5 dedos dobrados, no qual o eixo das abcissas indica a quantidade de experimento e as ordenadas indica o magnitude da resistência referente ao experimento respectivo. Fonte: Autor.

Executou-se uma análise individual do comportamento de cada dedo. Essa análise é responsável para informar qual sensor deve ser trocado ou mantido. Para o sensor ser trocado, ele tem que apresentar o mesmo valor nos 3 modos (dedo 0%, 50% e 100% dobrado) na maioria dos experimentos. O resultado do dedo polegar, dá um alerta para que seja trocado, no entanto, houve um teste com o algoritmo de treino e o mesmo não interferiu nas traduções. Já os outros dedos não apresentam este alerta, pois o valores do experimentos nos 3 modos estão na sua maioria diferentes.

O resultado do dedo polegar totalmente esticado, parcialmente dobrado e totalmente dobrado é apresentado na figura 45 :

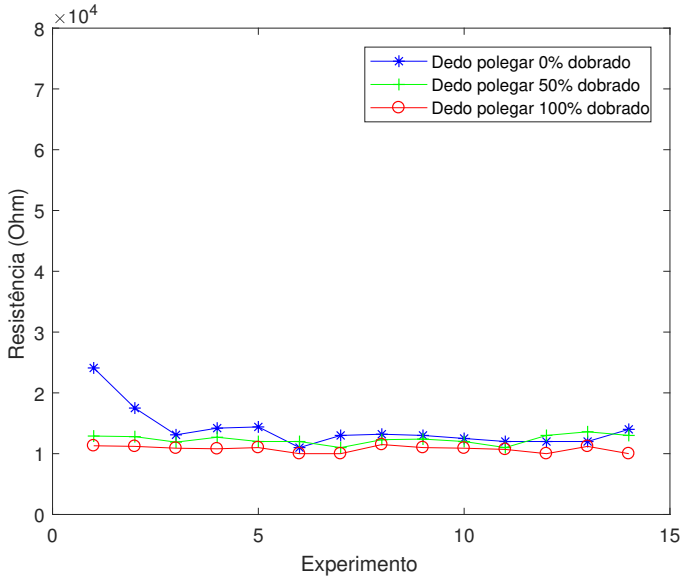


Figura 45 – Polegar em 3 condições de flexão distintas., no qual o eixo das abcissas indica a quantidade de experimento e as ordenadas indica o magnitude da resistência referente ao experimento respectivo. Fonte: Autor.

O resultado do dedo indicador totalmente esticado, parcialmente dobrado e totalmente dobrado é apresentado na figura 46 :

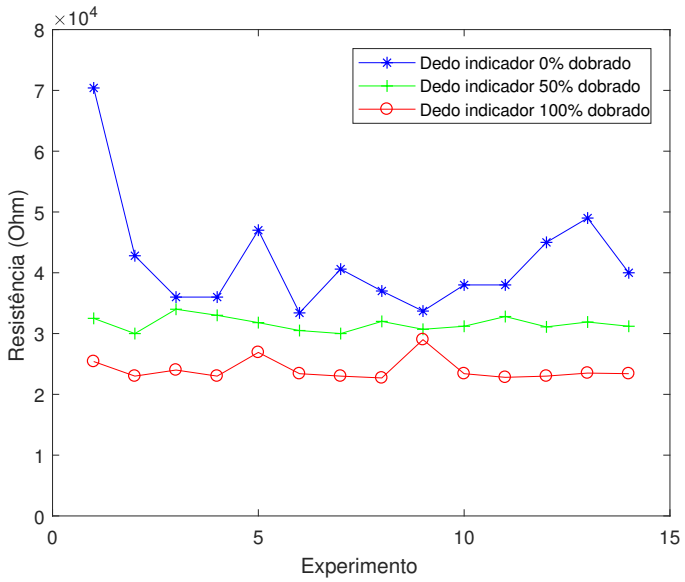


Figura 46 – Indicador em 3 condições de flexão distintas, no qual o eixo das abcissas indica a quantidade de experimento e as ordenadas indica o magnitude da resistência referente ao experimento respectivo. Fonte: Autor.

O resultado do dedo médio totalmente esticado, parcialmente dobrado e totalmente dobrado é apresentado na figura 47 :

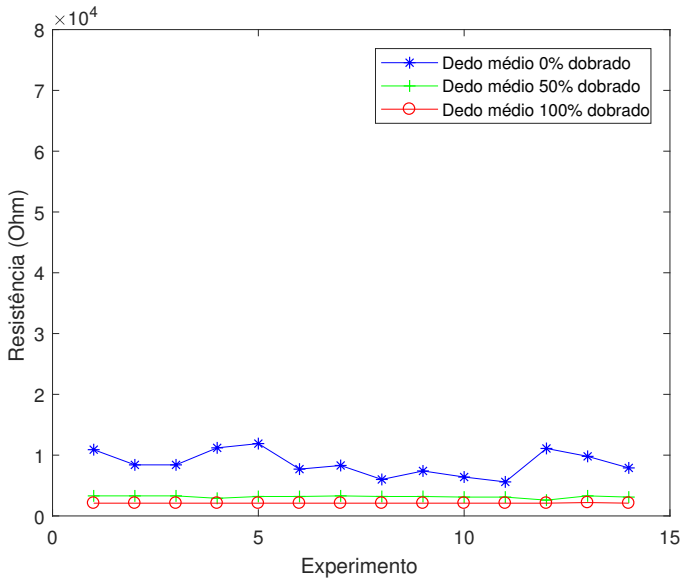


Figura 47 – Médio em 3 condições de flexão distintas, onde o eixo das abcissas indica a quantidade de experimento e as ordenadas indica o magnitude da resistência referente ao experimento respectivo. Fonte: Autor.

O resultado do dedo anelar totalmente esticado, parcialmente dobrado e totalmente dobrado é apresentado na figura 48 :

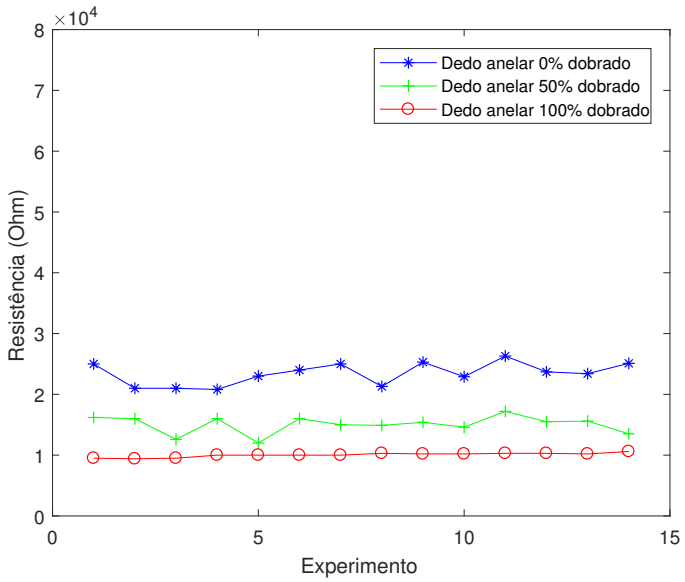


Figura 48 – Anelar em 3 condições de flexão distintas, no qual o eixo das abcissas indica a quantidade de experimento e as ordenadas indica o magnitude da resistência referente ao experimento respectivo. Fonte: Autor.

O resultado do dedo mindinho totalmente esticado, parcialmente dobrado e totalmente dobrado é apresentado na figura 49 :

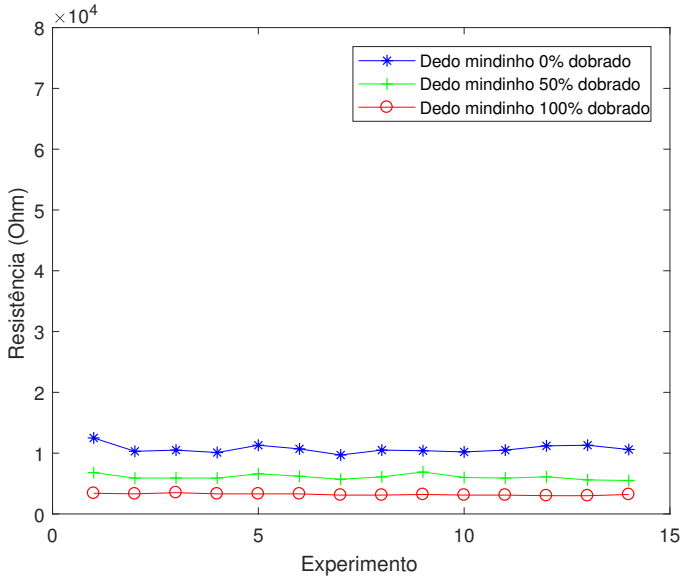


Figura 49 – Mindinho em 3 condições de flexão distintas, no qual o eixo das abcissas indica a quantidade de experimento e as ordenadas indica o magnitude da resistência referente ao experimento respectivo. Fonte: Autor.

4.2 Resultados Referentes a Rede Neural Artificial

A estrutura da rede neural artificial da LuTLI encontra-se na figura 30. Esta estrutura apresenta 11 entradas que representam os 3 eixos 'x','y' e 'z' do acelerômetro, os 3 eixos 'x','y' e 'z' do giroscópio e os 5 sensores flexíveis resistivos que representam o dedo polegar, indicador, médio, anelar e mindinho. A escolha de 2 camadas e 30 neurônios em cada, *batch size* igual à 10 e otimizador *adam* foram motivadas de forma empírica, que ao fazer o teste obteve-se excelentes índices de desempenho: acurácia de 99,97% e a função erro de 0,0021. Pelo mesmo motivo utilizou-se 70 épocas para atingir os mesmos índices de desempenhos. As épocas são as repetições dos dados de entrada que passam pela RNA até a atualização dos pesos.

A matriz de confusão é um dos indicadores de qualidade da rede neural artificial encontrado na figura 51. No momento do treinamento obteve 4798 dados de entrada do *dataset* de 22 classes diferentes demonstradas na figura 50, e dividiu-

se em 75% em dados de treinamento e 25% em dados de teste. Utilizando os 75% do *dataset* para treino, esses dados foram passados 70 vezes na RNA e os pesos atualizados até chegar nos índices de desempenho apresentados anteriormente. Logo após, definiu-se a estrutura da RNA com os pesos treinados, passou-se os 25% do *dataset* nesse modelo treinado e identificou os dados que a RNA conseguiu classificar e os dados que ela não classificar.

Após isso montou-se a matriz de confusão, na qual cada linha dela é referente a quantidade de classes reais e as colunas são referente a quantidade das classes da RNA treinada. A diagonal principal indica quantas classes da RNA treinada são iguais as classes reais, ou seja, a quantidade de classificação feita com sucesso. Observa-se que figura 51 apresenta somente magnitude na diagonal principal, logo, a classificação foi bem sucedida. Além disso, a acurácia não foi 100%, ou seja, não houver *overfit* que é o vício do modelo pelo *dataset* treinado, ou seja, se colocar outros dados de entrada a rede não consegue classificar, se restringindo apenas ao *dataset* treinado.

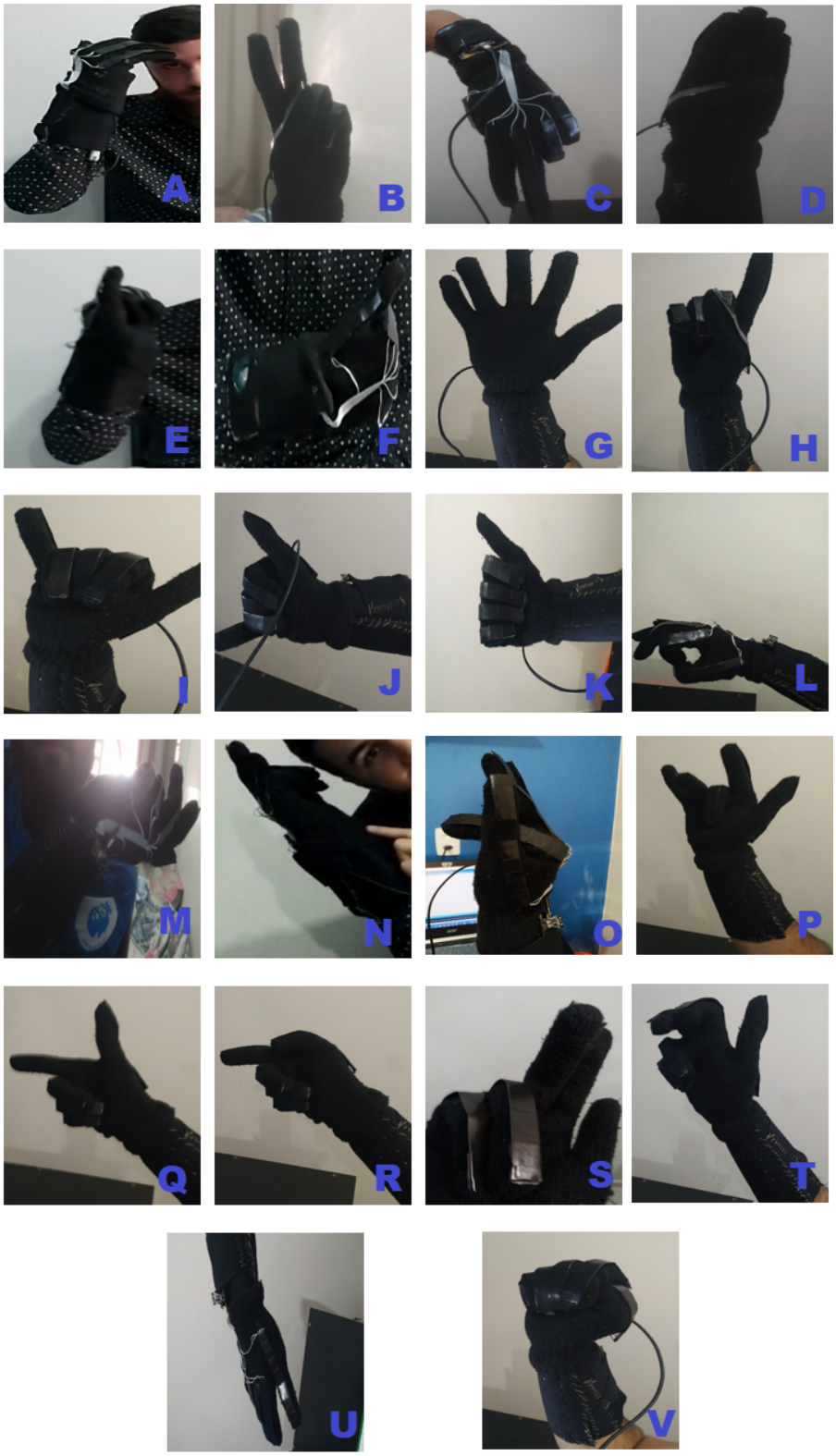


Figura 50 – 22 sinais que foram traduzidos pela LuTLI: (A) ‘Obrigado’, (B) ‘U’ (C) ‘N’, (D) ‘B’, (E) ‘Oi’, (F) ‘Tudo Bem?’ , (G) ‘Tchau’, (H) ‘Deus’, (I) ‘Abraço’, (J) ‘Desculpa’, (K) ‘Brincar’, (L) ‘Barato’, (M) ‘Café’, (N) ‘Dormir’, (O) ‘Futuro’, (P) ‘Fácil’, (Q) ‘Dois’, (R) ‘Música’, (S) ‘Igual’, (T) ‘Inteligente’, (U) ‘Luva’, (V) ‘Sinal’. Fonte: Autor

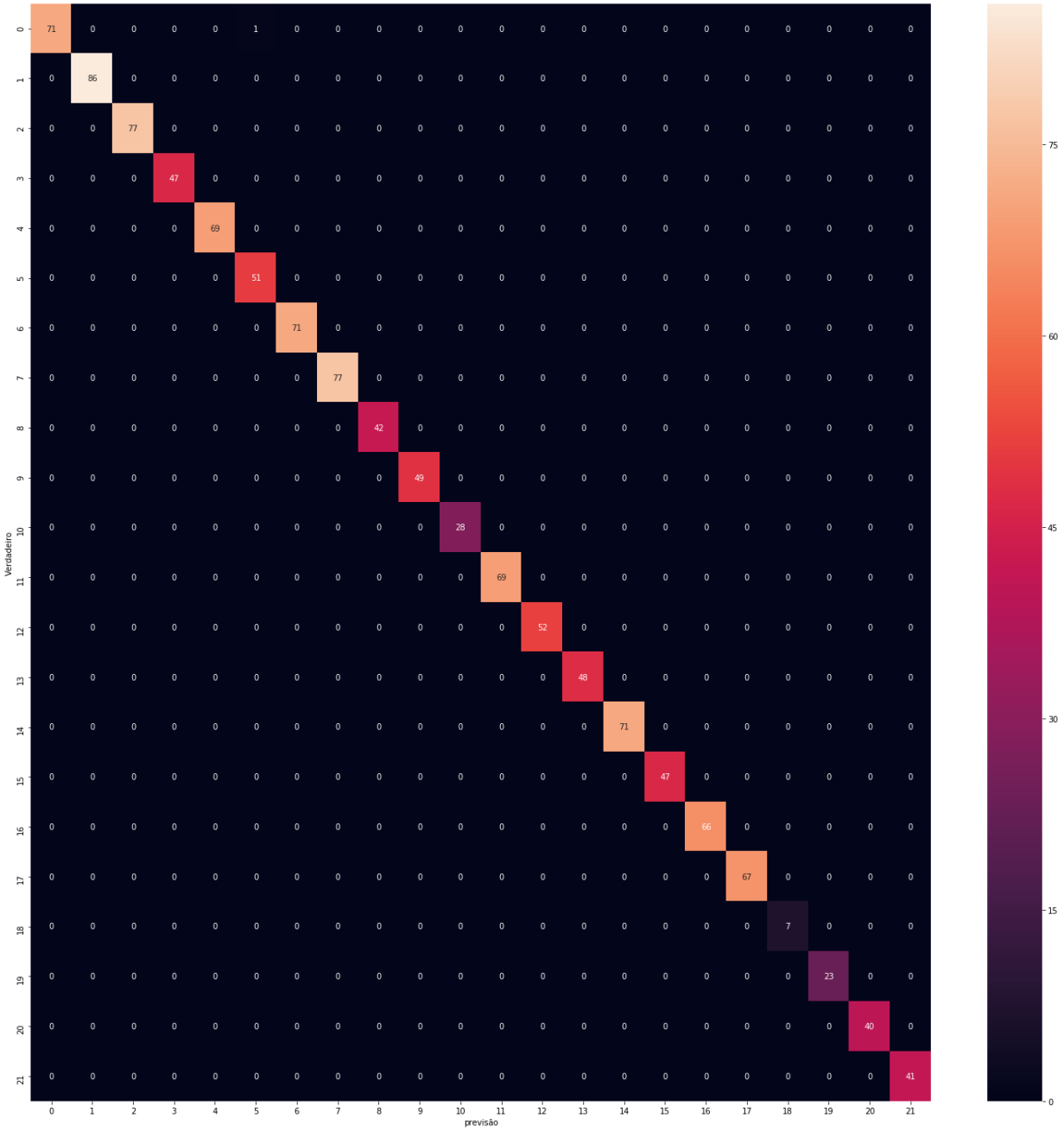


Figura 51 – Matriz de confusão, na qual a diagonal apresenta magnitudes que representam a quantidade de predições classificadas corretamente. Fonte: Autor.

4.3 Percepção do Usuário

Esse trabalho não teve a validação de um usuário PPD's. No entanto houve uma usuária que foi a mesma que auxiliou na classificação dos 22 sinais considerados básicos para LIBRAS. A percepção da usuária foi de que o produto tem grande potencial para ser comercializado e que hoje já atenderia boa parte da necessidade dos PPD's, no entanto deve haver outra configuração capaz de traduzir assim que executa o sinal de forma automática, sem a necessidade de apertar o reset. Porque o fato de ter que apertar o esse botão toda a vez que for usar é algo trabalhoso e demorado para se comunicar. Logo, deve-se ajustar esse detalhe do *reset* no âmbito funcional e melhorar o design para o âmbito comercial.

5 Conclusão

5.1 Considerações finais

Esse trabalho tratou da definição geral do sistema, que compreende informações relevantes ao processo de construção da Luva tradutora de LIBRAS inteligente (LuTLI). No primeiro momento teve-se a motivação que era resolver um problema que afeta muitos brasileiros, após isso, definiu-se os requisitos do projeto, materiais, as metodologias e em seguida definiu-se o sistema eletrônico. Priorizou a confecção dos sensores flexíveis, pois estes tinha uma função essencial no sistema.

Foram realizadas simulações, testes em *protoboard*, projeto de PCB e implementação final em placas de circuito impresso. Percebeu-se que o sistema de sensoriamento devia obrigatoriamente ter sensores que fornecessem sinais do movimento da mão e dos dedos, assim, a escolha foi direcionada para os sensores flexíveis e o *MPU-6050*. Já o sistema de atuação tem dois papeis importantes, um é adquirir os dados com o módulo *SD card* e o outro é exibir em forma de som a tradução com o módulo MP3. O sistema de processamento foi um dos sistema mais delicados, pois é ele que permitia o microcontrolador ser capaz de identificar os padrões dos sinais, para isso usou-se o método de redes neurais artificiais. A calibração do sensor *MPU-6050* foi essencial com auxílio do software *processing 3D* para os dados serem mais próximos da realidade. A confecção dos sensores flexíveis resistivos foi validada com os resultados da caracterização e o preço deste sensor é muito abaixo do encontrado no mercado. E por fim, com os índices de desempenho da RNA pode-se validar a melhor relação arquitetura e pesos.

O presente produto foi projetado visando a usabilidade e conforto do usuário, visto que no momento da confecção dos sensores optou-se por materiais que não agredissem a mão do usuário. A usabilidade do sistema é personalizada, pois assim como as pessoas falam com sotaques diferentes, as pessoas que comunicam-se em LIBRAS fazem os sinais com suas particularidades. A recomendação de uso de

sinais da LuTLI é que sejam executados com um *reset* antes da ação de um sinal.

5.2 Trabalhos Futuros

- **Ensaio clínico com usuário final:** O sistema projetado nesse trabalho tem característica de produto assistencial, logo, necessita de uma ensaio clínico seguindo todos os protocolos exigidos pelo órgão competente de ensaios clínicos nacional com a finalidade de validar o produto como comercial.
- **Sensor Acelerômetro e Giroscópio mais precisos:**
  Durantes a aquisição de dados e treinos da rede neural artificial percebeu-se uma certa limitação do sensor *MPU-6050*, o mesmo não tem um range suficiente para que haja um distinção clara de padrões entre os sinais. O sensor teve um funcionamento razoável a ponto de distinguir os sinais. No entanto, para trabalhos futuros, se aumentar a quantidade de sinais a serem classificados mais o sensor *MPU-6050* será obsoleto.
- **Integração *Web* e *Mobile*:**
  Esse sistema tem um potencial para ser integrado a sistema *web*, visto que os PPDs poderiam fazer uma entrevista de emprego ou até mesmo apresentarem-se via videoconferência para outros idiomas. Logo, as informações seriam passadas a um computador via USB ou até mesmo sem fio, havendo um programa que traduzisse os sinais para outros idiomas, tornando-o mais prático para usuário. O sistema poderia ter uma integração *mobile* para traduzir os sinais da luva em texto a fim de serem escritos em redes sociais. A comunicação poderia ser feita via *bluetooth*.
- **Integração com sistema IoT**
  A luva poderia ser utilizada não só para traduzir sinais, mas poderia ser utilizada para controlar eletrodomésticos a partir de um servidor que receberia os sinais da luva. Logo, a luva poderia controlar persianas, ligar televisores, abrir portas, portão eletrônicos e entre outras função de uma casa inteligente.
- **Educativo:**
  A luva poderia ter um cunho educativo, ensinando LIBRAS aos usuários

dela. No entanto seria necessário desenvolver uma plataforma que permitisse um desenvolvimento de aprendizagem progressivo com níveis.

- **Alimentação Renovável:**
  Em questão de *hardware*, poderia haver uma atualização que seria a alimentação por meio de mini placas solares. Tornando o produto mais sustentável.

Referências

ARDUINO_E_CIA. Como usar o módulo mp3 dfplayer mini | arduino e cia. 2017. (Acessado em 31/12/2019). Citado 2 vezes nas páginas 12 e 64.

BASTOS, I. L. O. Reconhecimento de sinais da libras utilizando descritores de forma e redes neurais artificiais. Instituto de Matemática. Departamento de Ciência da Computação, 2015. Citado na página 25.

BEZERRA, S. Reservoir computing com hierarquia para previsão de vazões médias diárias. 07 2016. Citado 2 vezes nas páginas 10 e 33.

BOYLESTAD, R. L.; NASHELSKY, L. Dispositivos eletrônicos e teoria de circuitos. Prentice-Hall do Brasil, v. 11, 2013. Citado na página 28.

DIGIKEY. Dfplayer - um mini mp3 player para arduino - dfrobot - placas de expansão | catálogo online | digikey electronics. 2019. (Acessado em 23/12/2019). Citado na página 63.

ESPRESSIF. esp32_datasheet_en.pdf. 2019. (Acessado em 04/12/2019). Citado 4 vezes nas páginas 10, 15, 30 e 31.

FERNANDES, F. S. diversidade na perda auditiva. v. 4, n. 2, p. 318–336, 2019. Citado na página 22.

GONZALES, R. C.; WOODS, R. E. Digital image processing. Prentice hall New Jersey, 2011. Citado 9 vezes nas páginas 10, 20, 34, 36, 38, 39, 42, 46 e 155.

HAYKIN, S. Redes neurais: princípios e prática. Bookman Editora, 2007. Citado 3 vezes nas páginas 32, 33 e 34.

HENRIQUE, D. R. Língua de sinais brasileira: análise de campanhas do ministério da saúde na perspectiva da pessoa surda. 2017. Citado na página 21.

INSTRUCTABLES. Make a flex sensor for robotic hand (cheap and simple): 4 steps (with pictures). 2016. (Acessado em 18/05/2019). Citado na página 57.

KERAS: the Python deep learning API. <https://keras.io/>. (Acessado em 25/01/2020). Citado na página 45.

LAGE, V. N.; SEGUNDO, A. K. R. O uso de giroscópios e acelerômetros para a modelagem matemática de uma plataforma com dois graus de liberdade. 2016. Citado 4 vezes nas páginas 15, 26, 27 e 51.

LENT, R. Cem bilhões de neurônios. v. 2, p. 631–639, 2010. Citado 3 vezes nas páginas 32, 33 e 34.

MACHADO, M. C. et al. Classificação automática de sinais visuais da língua brasileira de sinais representados por caracterização espaço-temporal. Universidade Federal do Amazonas, 2018. Citado na página 26.

MORI, N. N. R.; SANDER, R. E. História da educação dos surdos no brasil. v. 2, 2015. Citado na página 24.

RAMOS, L. C. Reconhecimento de gestos usando redes neurais artificais. 2011. Citado na página 25.

SENAI-SP. Sistemas digitais. SESI SENAI Editora, 2018. Citado na página 30.

SOARES, M. A. L. A educação do surdo no brasil. Autores Associados (Editora Autores Associados LTDA), 2015. Citado na página 21.

STOCK, I. M. Língua brasileira de sinais. 2018. Citado 2 vezes nas páginas 24 e 25.

THOMAZINI, D. Sensores industriais: fundamentos e aplicações. Saraiva Educação SA, 2018. Citado 2 vezes nas páginas 10 e 29.

USINAINFO. Comprar sd card arduino / leitor micro sd card - usinainfo. 2019. (Acessado em 24/01/2020). Citado 3 vezes nas páginas 10, 46 e 47.

Apêndices

APÊNDICE A – Código para escanear o endereço do MPU-6050 com auxílio da biblioteca i2cDEV

```
// --------------------------------------
// i2c_scanner
//
// Version 1
//    This program (or code that looks like it)
//    can be found in many places.
//    For example on the Arduino.cc forum.
//    The original author is not known.
// Version 2, Juni 2012, Using Arduino 1.0.1
//     Adapted to be as simple as possible by Arduino.cc user Krodal
// Version 3, Feb 26  2013
//    V3 by louarnold
// Version 4, March 3, 2013, Using Arduino 1.0.3
//    by Arduino.cc user Krodal.
//    Changes by louarnold removed.
//    Scanning addresses changed from 0...127 to 1...119,
//    according to the i2c scanner by Nick Gammon
//    http://www.gammon.com.au/forum/?id=10896
// Version 5, March 28, 2013
//    As version 4, but address scans now to 127.
//    A sensor seems to use address 120.
//  Modificado para ESP8266 E ESP32: Ítalo Rodrigo Moreira Borges
// Data: 13/04/2019
```

```
30 // GND -> GND
31 // SDA -> D7
32 // SCL -> D6
33 //
34 //
35 //Pinagem MPU6050 COM NODECMU ESP32
36 // VCC -> VIN
37 // GND -> GND
38 // SDA -> 22
39 // SCL -> 21
40 //
41 //
42
43 //Definindo os pi os SDA E SCL para comunicar com MPU-6050
44 #define SDA 22
45 #define SCL 21
46
47 //Bilboteca Modificada que permite utilizar o protocolo serial I2C
48 #include <Wire.h>
49
50 //Região de Configuração
51 void setup()
52 {
53   //Configurando os pinos para receber dados e o clock
54   Wire.begin(SDA,SCL);
55   //Configurando a taxa de transmissão de bit/s
56   Serial.begin(115200);
57   //Print para indicar o fim configuração
58   Serial.println("\nI2C Scanner");
59 }
60
61 // Região de repetição
62 void loop()
63 {
64
65   //Definindo as variáveis
66   byte error, address;
67   int nDevices;
68   //verificando o scanear
69   Serial.println("Scanning...");
```

```
70
71    //Algoritmo de scanear
72    nDevices = 0;
73    for(address = 1; address < 127; address++ )
74    {
75
76      Wire.beginTransmission(address);
77      error = Wire.endTransmission();
78
79      if (error == 0)
80      {
81        Serial.print("I2C device found at address 0x");
82        if (address<16)
83          Serial.print("0");
84        Serial.print(address,HEX);
85        Serial.println("  !");
86
87        nDevices++;
88      }
89      else if (error==4)
90      {
91        Serial.print("Unknow error at address 0x");
92        if (address<16)
93          Serial.print("0");
94        Serial.println(address,HEX);
95      }
96    }
97    if (nDevices == 0)
98      Serial.println("No I2C devices found\n");
99    else
100     Serial.println("done\n");
101
102   delay(5000);
103 }
```

APÊNDICE B – Código de Calibração da MPU-6050

```
// Arduino sketch that returns calibration offsets for MPU6050
//   Version 1.1  (31th January 2014)
// Done by Luis Ródenas <luisrodenaslorda@gmail.com>

/* ==========  LICENSE  ================================
I2Cdev device library code is placed under the MIT license
Copyright (c) 2011 Jeff Rowberg
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
===============================================================
//ADAPTADO PARA ESP32
//MODIFICADO POR: Ítalo Rodrigo Moreira Borges
// DATA: 12/05/2019
*/
// I2Cdev and MPU6050 must be installed as libraries
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

```

```
33      //ESP32
34      #define SDA 21
35      #define SCL 22
36  ////////////////////////   CONFIGURATION   //////////////////////////
37  //Change this 3 variables if you want to fine tune the skecth to your needs.
38  int buffersize=1000;//Amount of readings used to average, make it higher to
39  //get more precision but sketch will be slower  (default:1000)
40  int acel_deadzone=8;//Acelerometer error allowed, make it lower to get more
41  //precision, but sketch may not converge  (default:8)
42  int giro_deadzone=1;//Giro error allowed, make it lower to get more
43  //precision, but sketch may not converge  (default:1)
44  // default I2C address is 0x68
45  // specific I2C addresses may be passed as a parameter here
46  // AD0 low = 0x68 (default for InvenSense evaluation board)
47  // AD0 high = 0x69
48  //MPU6050 accelgyro;
49  MPU6050 accelgyro(0x68); // <-- use for AD0 high
50  int16_t ax, ay, az,gx, gy, gz;
51  int mean_ax,mean_ay,mean_az,mean_gx,mean_gy,mean_gz,state=0;
52  int ax_offset,ay_offset,az_offset,gx_offset,gy_offset,gz_offset;
53  ////////////////////////////   SETUP   ////////////////////////////////
54  void setup() {
55  // join I2C bus (I2Cdev library doesn't do this automatically)
56  Wire.begin(SDA,SCL);
57  // COMMENT NEXT LINE IF YOU ARE USING ARDUINO DUE
58  long int TWBR = 24;
59  // 400kHz I2C clock (200kHz if CPU is 8MHz). Measured 250kHz.
60  // initialize serial communication
61  Serial.begin(115200);
62  // initialize device
63  accelgyro.initialize();
64  // wait for ready
65  while (Serial.available() && Serial.read()); // empty buffer
66  while (!Serial.available()){
67  Serial.println(F("Send any character to start sketch.\n"));
68  delay(1500);
69  }
70  while (Serial.available() && Serial.read()); // empty buffer again
71  // start message
72  Serial.println("\nMPU6050 Calibration Sketch");
```

```
73  delay(2000);
74
75  Serial.println("\nYour MPU6050 should be placed in horizontal position, with);
76  Serial.println("letters facing up. \nDon't touch it until you see a end msg.");
77  delay(3000);
78  // verify connection
79  Serial.print(accelgyro.testConnection() ? "MPU6050 connection successful: " );
80  Serial.println("MPU6050 connection failed");
81  delay(1000);
82  // reset offsets
83  accelgyro.setXAccelOffset(0);
84  accelgyro.setYAccelOffset(0);
85  accelgyro.setZAccelOffset(0);
86  accelgyro.setXGyroOffset(0);
87  accelgyro.setYGyroOffset(0);
88  accelgyro.setZGyroOffset(0);
89  }
90  ////////////////////////   LOOP   ////////////////////////////
91  void loop() {
92  if (state==0){
93  Serial.println("\nReading sensors for first time...");
94  meansensors();
95  state++;
96  delay(1000);
97  }
98  if (state==1) {
99  Serial.println("\nCalculating offsets...");
100 calibration();
101 state++;
102 delay(1000);
103 }
104 if (state==2) {
105 meansensors();
106 Serial.println("\nFINISHED!");
107 Serial.print("\nSensor readings with offsets:\t");
108 Serial.print(mean_ax);
109 Serial.print("\t");
110 Serial.print(mean_ay);
111 Serial.print("\t");
112 Serial.print(mean_az);
```

```
113 Serial.print("\t");
114 Serial.print(mean_gx);
115 Serial.print("\t");
116 Serial.print(mean_gy);
117 Serial.print("\t");
118 Serial.println(mean_gz);
119 Serial.print("Your offsets:\t");
120 Serial.print(ax_offset);
121 Serial.print("\t");
122 Serial.print(ay_offset);
123 Serial.print("\t");
124 Serial.print(az_offset);
125 Serial.print("\t");
126 Serial.print(gx_offset);
127 Serial.print("\t");
128 Serial.print(gy_offset);
129 Serial.print("\t");
130 Serial.println(gz_offset);
131 Serial.println("\nData is printed as: acelX acelY acelZ giroX giroY giroZ");
132 Serial.println("Check that your sensor readings are close to 0 0 16384 0 0 0");
133 Serial.println("write your offsets in mpu.setXAccelOffset(youroffset)");
134 while (1);
135 }
136 }
137 ////////////////////////   FUNCTIONS   ////////////////////////////
138 void meansensors(){
139 long i=0,buff_ax=0,buff_ay=0,buff_az=0,buff_gx=0,buff_gy=0,buff_gz=0;
140 while (i<(buffersize+101)){
141 // read raw accel/gyro measurements from device
142 accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
143 if (i>100 && i<=(buffersize+100)){ //First 100 measures are discarded
144 buff_ax=buff_ax+ax;
145 buff_ay=buff_ay+ay;
146 buff_az=buff_az+az;
147 buff_gx=buff_gx+gx;
148 buff_gy=buff_gy+gy;
149 buff_gz=buff_gz+gz;
150 }
151 if (i==(buffersize+100)){
152 mean_ax=buff_ax/buffersize;
```

```
153 mean_ay=buff_ay/buffersize;
154 mean_az=buff_az/buffersize;
155 mean_gx=buff_gx/buffersize;
156 mean_gy=buff_gy/buffersize;
157 mean_gz=buff_gz/buffersize;
158 }
159 i++;
160 delay(2); //Needed so we don't get repeated measures
161 }
162 }
163 void calibration(){
164 ax_offset=-mean_ax/8;
165 ay_offset=-mean_ay/8;
166 az_offset=(16384-mean_az)/8;
167 gx_offset=-mean_gx/4;
168 gy_offset=-mean_gy/4;
169 gz_offset=-mean_gz/4;
170 while (1){
171 int ready=0;
172 accelgyro.setXAccelOffset(ax_offset);
173 accelgyro.setYAccelOffset(ay_offset);
174 accelgyro.setZAccelOffset(az_offset);
175 accelgyro.setXGyroOffset(gx_offset);
176 accelgyro.setYGyroOffset(gy_offset);
177 accelgyro.setZGyroOffset(gz_offset);
178 meansensors();
179 Serial.println("...");
180 if (abs(mean_ax)<=acel_deadzone) ready++;
181 else ax_offset=ax_offset-mean_ax/acel_deadzone;
182 if (abs(mean_ay)<=acel_deadzone) ready++;
183 else ay_offset=ay_offset-mean_ay/acel_deadzone;
184 if (abs(16384-mean_az)<=acel_deadzone) ready++;
185 else az_offset=az_offset+(16384-mean_az)/acel_deadzone;
186 if (abs(mean_gx)<=giro_deadzone) ready++;
187 else gx_offset=gx_offset-mean_gx/(giro_deadzone+1);
188 if (abs(mean_gy)<=giro_deadzone) ready++;
189 else gy_offset=gy_offset-mean_gy/(giro_deadzone+1);
190 if (abs(mean_gz)<=giro_deadzone) ready++;
191 else gz_offset=gz_offset-mean_gz/(giro_deadzone+1);
192 if (ready==6) break; } }
```

APÊNDICE C – Código de que extrai os parâmetros do sensor MPU-6050

```
1  //Programa : Teste MPU-6050 na ESP32
2  //Alteracoes e adaptacoes : Ítalo Rodrigo Moreira Borges
3  //DATA: 16/06/2019
4  //Baseado no programa original de JohnChi e Arduino e Cia
5
6  //Carrega a biblioteca Wire
7  #include<Wire.h>
8  #include "MPU6050.h"
9  //#include "I2Cdev.h"
10
11 //Endereco I2C do MPU6050
12 const int MPU=0x68;
13 //ESP32
14 #define SDA 21
15 #define SCL 22
16
17 //Variaveis para armazenar valores dos sensores
18 int AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
19 //Definindo objeto
20 MPU6050 mpu;
21
22 void setup()
23 {
24   Serial.begin(9600);
25   mpu.setXAccelOffset (-196);
26   mpu.setYAccelOffset (-2711);
27   mpu.setZAccelOffset (5192);
28
29   mpu.setXGyroOffset (31);
30   mpu.setYGyroOffset (5);
31   mpu.setZGyroOffset (8);
32   Wire.begin(SDA,SCL);
```

```
33    Wire.beginTransmission(MPU);
34    Wire.write(0x6B);
35
36    //Inicializa o MPU-6050
37    Wire.write(0);
38    Wire.endTransmission(true);
39
40  }
41
42  void loop()
43  {
44    Wire.beginTransmission(MPU);
45    Wire.write(0x3B);  // starting with register 0x3B (ACCEL_XOUT_H)
46    Wire.endTransmission(false);
47
48    //Solicita os dados do sensor
49    Wire.requestFrom(MPU,14,true);
50
51    //Armazena o valor dos sensores nas variaveis correspondentes
52    AcX=Wire.read()<<8|Wire.read(); //0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
53    AcY=Wire.read()<<8|Wire.read(); //0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
54    AcZ=Wire.read()<<8|Wire.read(); //0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
55    Tmp=Wire.read()<<8|Wire.read(); //0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
56    GyX=Wire.read()<<8|Wire.read(); //0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
57    GyY=Wire.read()<<8|Wire.read(); //0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
58    GyZ=Wire.read()<<8|Wire.read(); //0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
59
60
61    //Mostra os valores na serial
62    Serial.print("Acel. X = "); Serial.print(AcX);
63    Serial.print(" | Y = "); Serial.print(AcY);
64    Serial.print(" | Z = "); Serial.print(AcZ);
65    Serial.print(" | Gir. X = "); Serial.print(GyX);
66    Serial.print(" | Y = "); Serial.print(GyY);
67    Serial.print(" | Z = "); Serial.print(GyZ);
68    Serial.print(" | Temp = "); Serial.println(Tmp/340.00+36.53);
69
70    //Aguarda 300 ms e reinicia o processo
71    delay(300);
72  }
```

APÊNDICE D – Código de Aquisição do dataset para o treinamento da Rede Neural Artificial em arduino

```
1  // Autor: Ítalo Rodrigo Moreira Borges
2  // DATA: 11/07/2019
3
4  // Bibliotecas referente ao cartão SD
5  #include "FS.h"
6  #include "SD.h"
7  #include <SPI.h>
8
9  //Bibliotecas referente ao sensor MPU-6050
10 #include<Wire.h>
11 #include "MPU6050.h"
12
13
14 // Definindo portas de entrada e saída a variáveis
15
16 // Modulo SD
17 #define SD_CS 5
18
19 //Push Button
20 #define STOP 4
21
22 //MPU-6050
23 #define SDA 21
24 #define SCL 22
25
26 //Sensor Flexível
27 #define polegar   36
28 #define indicador 39
29 #define medio     34
```

```
30 #define anelar    35
31 #define mindinho  32
32
33 // Salve o número de leitura na memória RTC
34 RTC_DATA_ATTR int readingID = 0;
35
36 //Variável que carregara a informação para o "data.txt"
37 String dataMessage;
38
39 //Endereco I2C do MPU6050
40 const int MPU=0x68;
41
42 //Variaveis para armazenar valores dos sensores
43 int AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
44
45 //Definindo objeto
46 // O objeto é de suma importância para executar as
47 //funções como ler o acelerômetro, giroscópio e a temperatura
48 MPU6050 mpu;
49
50
51
52 void setup () {
53
54   // Iniciar comunicação serial para fins de depuração
55   Serial.begin (115200);
56
57   // Declarando o Push Button como Porta de Entrada
58   pinMode(STOP, INPUT);
59
60   //Recebendo os valores do sensor flex em cada dedo
61   pinMode(polegar,INPUT);
62   pinMode(indicador,INPUT);
63   pinMode(medio,INPUT);
64   pinMode(anelar,INPUT);
65   pinMode(mindinho,INPUT);
66
67   //Extraindo os dados do acelerômetro, giroscópio
68   // nos três eixos com auxilio do objeto "mpu"
69   mpu.setXAccelOffset (-196);
```

```
70    mpu.setYAccelOffset (-2711);
71    mpu.setZAccelOffset (5192);
72
73    mpu.setXGyroOffset (31);
74    mpu.setYGyroOffset (5);
75    mpu.setZGyroOffset (8);
76
77    // Definindo as portas SDA e SCL para executarem o protocolo I2C
78    Wire.begin(SDA,SCL);
79    // Iniciar o protocolo I2C
80    Wire.beginTransmission(MPU);
81    // Direcionar o endereço do escravo nessa comunicação
82    Wire.write(0x6B);
83
84    //Inicializa o MPU-6050
85    Wire.write(0);
86    Wire.endTransmission(true);
87
88    // Inicialize o cartão SD
89    SD.begin (SD_CS);
90
91    // Verificando se a conexão do SD com ESP32 foi feita
92    if (! SD.begin (SD_CS)) {
93      Serial.println ("Falha na montagem da placa");
94      return;
95    }
96    uint8_t cardType = SD.cardType ();
97
98    //Verificando se o Modulo SD tem algum cartão SD
99    if (cardType == CARD_NONE) {
100     Serial.println ("Nenhum cartão SD conectado");
101     return;
102   }
103   Serial.println ("Inicializando o cartão SD ...");
104   if (! SD.begin (SD_CS)) {
105     Serial.println ("ERRO - Falha na inicialização do cartão SD!");
106     return; // falha no init
107   }
108
109   // Se o arquivo data.txt não existir
```

```
110   // Crie um arquivo no cartão SD e escreva os rótulos de dados
111   File file = SD.open ("/ data.txt");
112   if (!file) {
113     Serial.println ("O arquivo não existe");
114     Serial.println ("Criando arquivo ...");
115     writeFile(SD, "/data.txt","ID de leitura \r \n");
116   }
117    // E se existir, ele irá subscrever
118   else {
119     Serial.println ("O arquivo já existe");
120   }
121   file.close ();
122
123   //Está acrescentando no arquivo o texto entre parentese
124   appendFile(SD,
125 "/data.txt","AcX,AcY,AcZ,GyX,GyY,GyZ,polegar,indicador,medio,anelar,mindinho,classe \n");
126
127 }
128
129 void loop () {
130
131   //PARTE DO MOVIMENTO
132
133   //Inicializando a transmissão
134   Wire.beginTransmission(MPU);
135   // Inicializando com o registrado 0x3B (ACCEL_XOUT_H)
136   Wire.write(0x3B);
137   //Verificando o funcionamento
138   Wire.endTransmission(false);
139
140   //Solicita os dados do sensor
141   Wire.requestFrom(MPU,14,true);
142
143   //Armazena o valor dos sensores nas variáveis correspondentes
144   AcX=Wire.read()<<8|Wire.read(); //0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
145   AcY=Wire.read()<<8|Wire.read(); //0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
146   AcZ=Wire.read()<<8|Wire.read(); //0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
147   Tmp=Wire.read()<<8|Wire.read(); //0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
148   GyX=Wire.read()<<8|Wire.read(); //0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
149   GyY=Wire.read()<<8|Wire.read(); //0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
```

```
150   GyZ=Wire.read()<<8|Wire.read(); //0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
151
152
153
154
155  // PARTE DO ARMAZENAMENTO
156
157
158    // Criando o documento "data.txt" no cartão SD
159      File file = SD.open ("/ data.txt");
160
161
162 // Se o botão estiver pressionado, a função "appendFile" vai acrescentar
163 //em "data.txt" os valores das varáveis transformado em
164 //string por meio da função "String.(variavel).c_str()"
165 if(digitalRead(STOP) == LOW){
166   appendFile(SD, "/data.txt",String(AcX).c_str());
167   appendFile(SD, "/data.txt",",");
168   appendFile(SD, "/data.txt",String(AcY).c_str());
169   appendFile(SD, "/data.txt",",");
170   appendFile(SD, "/data.txt",String(AcZ).c_str());
171   appendFile(SD, "/data.txt",",");
172   appendFile(SD, "/data.txt",String(GyX).c_str());
173   appendFile(SD, "/data.txt",",");
174   appendFile(SD, "/data.txt",String(GyY).c_str());
175   appendFile(SD, "/data.txt",",");
176   appendFile(SD, "/data.txt",String(GyZ).c_str());
177   appendFile(SD, "/data.txt",",");
178   appendFile(SD, "/data.txt",String(analogRead(polegar)).c_str());
179   appendFile(SD, "/data.txt",",");
180   appendFile(SD, "/data.txt",String(analogRead(indicador)).c_str());
181   appendFile(SD, "/data.txt",",");
182   appendFile(SD, "/data.txt",String(analogRead(medio)).c_str());
183   appendFile(SD, "/data.txt",",");
184   appendFile(SD, "/data.txt",String(analogRead(anelar)).c_str());
185   appendFile(SD, "/data.txt",",");
186   appendFile(SD, "/data.txt",String(analogRead(mindinho)).c_str());
187   appendFile(SD, "/data.txt",",");
188   appendFile(SD, "/data.txt","22");
189   appendFile(SD, "/data.txt","\n");
```

```
190   file.close ();
191 }
192   //Caso contrário, não preenche nada no documento "data.txt"
193   else {
194   //appendFile(SD, "/data.txt","");
195   appendFile(SD, "/data.txt","\n");
196   }
197
198   // (Debugar) Mostrar na tela se o botão está sendo pressionado
199   Serial.println(digitalRead(STOP));
200   //Espera um tempo para armazenar
201   delay(500);
202
203 }
204
205
206 // Função responsável por Gravar o texto no cartão SD e definir
207 //as se o arquivo será só de leitura, escrita ou os dois.
208 void writeFile (fs :: FS & fs, const char * caminho, const char * mensagem) {
209   Serial.printf ("Escrevendo arquivo:% s \ n", caminho);
210
211   File file = fs.open (caminho, FILE_WRITE);
212   if (!file) {
213     Serial.println ("Falha ao abrir o arquivo para gravação");
214     return;
215   }
216   if (file.print (mensagem)) {
217     Serial.println ("Arquivo gravado");
218   } else {
219     Serial.println ("Falha na gravação");
220   }
221   file.close ();
222 }
223
224 // Acrescentar dados ao cartão SD
225 void appendFile (fs :: FS & fs, const char * path, const char * message) {
226   Serial.printf ("Anexando ao arquivo:% s \n", path);
227
228   File file = fs.open (path, FILE_APPEND);
229   if (!file) {
```

```
230     Serial.println ("Falha ao abrir o arquivo para anexar");
231     return;
232   }
233   if (file.print (message)) {
234     Serial.println ("Mensagem anexada");
235   } else {
236     Serial.println ("falha ao anexar");
237   }
238   file.close ();
239 }
240
241 }
```

APÊNDICE E – Código de Treinamento da Rede Neural Artificial e extração dos pesos treinados em Python

```
# Autor:Ítalo Rodrigo Moreira Borges
# Data 02/09/2019


#IMPORTAÇÕES

#Necessária para manuseiar documentos .csv
import pandas as pd
# É o framework tem as função de RNA que o Keras utiliza
import tensorflow as tf
# Necessária para transformar uma variável em variável que possa
#Utilizar operações matriciais
import numpy as np
# É necessária para fazer o pre-processamento
from sklearn.preprocessing import MinMaxScaler
# É necessária para definir a função erro
from sklearn.metrics import mean_absolute_error
# É necessária para definir o tipo de arquitetura de RNA
from keras.models import Sequential
# É necessária para definir o tipo de camada da RNA
from keras.layers import Dense
# É necessária para transformar as classes em algo que possa ser processado
from keras.utils import np_utils

#PRÉ-PROCESSMENTO

#Importando os dataset
pre_dataset = pd.read_csv('datasetV2lutliV3.csv')
# Selecionando e armazenando as 11 entradas
```

```
pre_previsor = pre_dataset.iloc[:, 0:11]
# Selecionado e armazenando as classes
pre_classe = pre_dataset.iloc[:, 11]

#Selecionando as linhas que tem valor e descartando as outras
previsor = pre_previsor.values
#Transformando as entradas em variáveis que aceitem operações de matrizes
previsores=np.array(previsor)

#Selecionando as linhas que tem valor e descartando as outras
classe = pre_classe.values
#Transformando as entradas em variáveis que aceitem operações de matrizes
classe=np.array(classe)

#NORMALIZAÇÃO
#Transformando as variáveis de entrada em float, porque as entradas
#são inteiras
previsores = previsores.astype('float32')
# Fazendo a normalização ao dividir por 2500
previsores /= 2500


#Processando a variável que representa a classe do dataset
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
classe = labelencoder.fit_transform(classe)
classe_dummy = np_utils.to_categorical(classe)

#Separando os dados de entrada e classe entre treinamento e teste
from sklearn.model_selection import train_test_split
previsores_treinamento, previsores_teste, classe_treinamento, classe_teste =
train_test_split(previsores, classe_dummy, test_size=0.25)

# Definindo a estrutrua da rede neural artificial
classificador = Sequential()
classificador.add(Dense(units = 30, activation = 'relu', input_dim = 11))
classificador.add(Dense(units = 30, activation = 'relu'))
classificador.add(Dense(units = 22, activation = 'softmax'))
classificador.compile(optimizer = 'adam', loss = 'mean_squared_error',
                      metrics = ['categorical_accuracy'])
```

```
70 classificador.fit(previsores_treinamento, classe_treinamento, batch_size = 10,
71                   epochs = 70)
72
73 #Extraindo a Matriz de Confusão
74 resultado = classificador.evaluate(previsores_teste, classe_teste)
75 previsoes = classificador.predict(previsores_teste)
76 previsoes = (previsoes > 0.5)
77 import numpy as np
78 classe_teste2 = [np.argmax(t) for t in classe_teste]
79 previsoes2 = [np.argmax(t) for t in previsoes]
80
81 from sklearn.metrics import confusion_matrix
82 matriz = confusion_matrix(previsoes2, classe_teste2)
83
84 # Salvando o modelo
85 classificador_json = classificador.to_json()
86 with open('modeloTCCV11.json', 'w') as json_file:
87     json_file.write(classificador_json)
88 classificador.save_weights('modeloTCCV11.h5')
89
90 #Extraindo os pesos
91 pesos0=classificador.layers[0].get_weights()
92 pesos1=classificador.layers[1].get_weights()
93 pesos2=classificador.layers[2].get_weights()
94
95 #Transformando os pesos de lista para arrays
96 pesos0=np.array(pesos0[0])
97 pesos1=np.array(pesos1[0])
98 pesos2=np.array(pesos2[0])
```

APÊNDICE F – Código da Estrutura da Rede Neural Artificial com os pesos treinados em arduino

```
// Título: Estrutura da RNA treinda com sensores e com os atuadores
// Autor: Ítalo Rodrigo Moreira Borges
// V1
// Data: 23/10/2019
// V2
//Data: 01/12/2019
// V3
//Data: 08/12/2019

//Carregar as Bibliotecas da RNA

#include <stdio.h>
#include <stdlib.h>
//Necessária para fazer operação exponenciais
#include <math.h>


//Carrega a biblioteca Wire
#include<Wire.h>
#include "MPU6050.h"

//Carregar a Biblioteca do Sistema de Áudio (DFplayerMP3)
#include <Arduino.h>
#include "DFRobotDFPlayerMini.h"

//Definindo as portas do Sistema de Áudio
#define RX2 16
#define TX2 17
//Definindo as portas da ESP32 com MPU-6050
```

```
30 #define SDA 21
31 #define SCL 22
32 // Definindo o nome das portas dos sensores flexíveis
33 #define polegar   36
34 #define indicador 39
35 #define medio     34
36 #define anelar    35
37 #define mindinho  32
38
39 //Definindo objeto MPU-6050
40 MPU6050 mpu;
41 //Definindo objeto do sistema de Áudio
42 HardwareSerial mySoftwareSerial(1);  // UART1 (0), UART2 (1), UART3 (2).
43 DFRobotDFPlayerMini myDFPlayer;   // Cria a variável "myDFPlayer"
44
45
46 //Variáveis globais da RNA
47 float pesos0[11][30]=   { 0.137891516, -0.225320503, -0.237701103,
48         -0.182862639,  0.176546708,  0.0728628561,
49         -0.441293061, -0.224257246,  0.242118567,
50         -0.411614060,  0.333650619,  0.108491607,
51          0.182674438, -0.313631237, -0.449434131,
52          0.233256757, -0.0560339093, -0.000966697931,
53         -0.128564388,  0.187274426,  0.0124154370,
54         -0.186780974, -0.167195886,  0.360116154,
55          0.0687681139,  0.591279447, -0.0816421062,
56          0.291150361,  0.406638235, -0.0273020845,
57        -0.469512731,  0.0707647204,  0.269488573,
58         -0.374515057,  0.236005366,  0.402058631,
59          0.135014459, -0.336448491, -0.108403713,
60          0.349581659, -0.106416233, -0.561245859,
61          0.280817926, -0.308736593,  0.520401120,
62         -0.630731106, -0.212552577, -0.189828038,
63          0.620526671,  0.209825844,  0.0618879125,
64          0.186301649, -0.0634524450, -0.315555662,
65         -0.363250375, -0.472705990,  0.199333221,
66          0.412366062,  0.365792364,  0.235591635,
67         0.286597282,  0.125162408,  0.00196110969,
68          0.00203207135, -0.331962883, -0.318747222,
69          0.123793848, -0.269417197,  0.390299320,
```

```
70          0.262073129, -0.348711938,  0.435544938,
71         -0.427279145,  0.243501052, -0.0184687302,
72          0.128061086, -0.336559564, -0.321750611,
73         -0.531382322, -0.0408119634,  0.365059584,
74          0.161134675,  0.348655105,  0.346908689,
75          0.355170608, -0.00597212836,  0.205702931,
76          0.0149823669, -0.617150605, -0.213470370,
77        -0.0994950384, -0.744427621,  0.134790346,
78          0.119725794, -0.199352369,  0.0647558272,
79         -0.0290075727, -0.0121184587, -0.0615259781,
80         -0.0368756168, -0.348534942,  0.217402726,
81          0.105437316, -0.231049329,  0.226894721,
82         -0.0120870844,  0.145383745, -0.111824214,
83         -0.0150846550,  0.232724175, -0.352065325,
84         -0.212671131,  0.0777469724,  0.00265283789,
85         -0.609817326,  0.103473000,  0.319364935,
86         -0.0355707109, -0.0359243825,  0.0825574845,
87         0.216020867,  0.225459769, -0.141242787,
88         -0.242417201, -0.185031414,  0.153509378,
89          0.00573252328,  0.136767715, -0.0262408536,
90          0.128851891, -0.267684132,  0.121131085,
91         -0.0704214498, -0.184161916,  0.105849266,
92          0.0958587155, -0.0706535280, -0.255392551,
93          0.132208586,  0.225198284, -0.433379382,
94          0.188489705,  0.341619492,  0.0224932637,
95         -0.0538140535, -0.0620515645, -0.243158922,
96         -0.254534006, -0.0107252533,  0.161880016,
97         0.207861394, -0.537643552, -0.0860443935,
98         -0.0382187366, -0.0911735296,  0.209048212,
99          0.00225104531, -0.300058991,  0.151567876,
100         0.278904825, -0.242320359, -0.0649748966,
101         0.0134935444, -0.291234165,  0.0237297378,
102         0.206268147, -0.324634641, -0.247614667,
103         0.0325243920, -0.151230425, -0.181990445,
104         0.113436766, -0.322484642,  0.0279362015,
105         0.00741610955, -0.0401593111, -0.0967848375,
106        -0.0737005100, -0.0227541067, -0.0945305824,
107       -0.703183591,  0.315769315,  1.66468549,
108        -0.372760355,  0.657580853,  0.864120483,
109        -0.253206611,  0.359448344, -1.38068533,
```

```
110          0.895279169, -0.402648479, -0.970635474,
111          0.938301444,  0.183454469, -0.948011816,
112         -0.0711046830,  0.0965748429,  0.0177179575,
113          2.61357880, -1.19345963,  0.0368654318,
114         -1.65693998, -1.65295959,  0.0523510277,
115         -0.652377188, -0.403819442, -0.186093614,
116         -0.113671236,  0.281409323,  2.18387771,
117          1.50991106,  0.206706807, -0.0923933759,
118          0.309829324,  0.0361752436, -0.229311854,
119         -0.166942865, -0.00927948952, -0.936714232,
120         -2.06852055,  0.242337704, -0.00329423137,
121         -0.653798878, -0.114121974,  2.87948895,
122         -0.142247185,  0.0281168222, -0.117034644,
123          0.0972913876,  1.39212143, -0.637226105,
124          1.02498162, -0.438500255, -1.25139999,
125          0.584764361,  0.0768837780, -1.97536802,
126          1.95363867, -0.574188530, -1.61777115,
127          2.35480118,  0.142789125,  1.61058354,
128         -0.355032265,  0.641788423, -1.50438058,
129         -0.313575566,  0.0871969461, -2.98295307,
130          1.30396616, -0.0357800834,  1.59393418,
131          0.0648524463, -0.170193523,  0.340041757,
132         -1.24270654, -0.251525223,  0.295973152,
133          0.419677019, -2.38667107, -0.385742813,
134         -1.31898093, -0.612415373, -2.06105852,
135          0.854033053, -0.162697732, -0.947160006,
136          2.13719010,  0.227180675,  0.398255527,
137          1.02718186, -0.228367120,  1.74538088,
138         -0.337366104,  0.230433330,  0.973164022,
139         -2.75730324, -0.200028434, -0.0660054013,
140         -2.73201203,  0.191968113,  0.890253782,
141         -0.0225752294,  0.0758781135,  0.603422344,
142          0.604690075,  0.161927670,  0.288999707,
143          0.559752285, -0.770687222,  0.0969838053,
144         -0.984113395,  0.00278726849,  1.40428126,
145         -0.192706764,  0.342645973, -2.15734220,
146         -0.541645110, -0.791530609,  0.771629333,
147        -1.02890301, -0.106861033,  2.03062749,
148          0.0939872265,  0.103938073,  1.76684868,
149         -1.61158299, -0.222288102,  0.978676081,
```

```
        -1.28141749, -0.259951830, -0.845838547,
         0.849564016,  0.0112691848, -1.65759814,
        -0.214906707, -0.290492386,  0.182434529,
         1.32543802, -1.53145671,  0.0202270728,
         0.519921362, -0.260250658,  1.37841535,
        -0.222952321, -0.136719257,  0.349773228,
        -0.545830429,  0.248421744,  1.68074751};


float pesos1[30][30]={-0.0147002041,  0.0527593829, -0.322332263,
        -0.302773178, -0.0342779234, -0.137571499,
        -0.134246975,  0.191482216,  0.154840618,
        -0.195103809,  0.497096807,  0.342253476,
        -0.0854487047,  0.227117613, -0.211033732,
         0.204572201, -0.457215786,  0.0254866071,
         0.216715947, -0.0171619859,  0.0426263884,
        -0.388054878,  0.921841741,  0.287052512,
         0.403118491, -0.200487465, -0.434871793,
         0.252593279,  0.233353600,  0.348726928,
        0.227086395, -0.0718984976,  0.00417923741,
         0.315197408,  0.0498640127,  0.167285264,
         0.158565968, -0.228412539,  0.196828753,
         0.229832724,  0.142018661, -0.0332687050,
         0.00641571544, -0.0882356465, -0.429515630,
         0.323437095, -0.230106622,  0.0438450649,
        -0.146931097, -0.132332951, -0.0985457003,
         0.195151344, -0.245502964,  0.168661907,
         0.194902703,  0.0556880645,  0.131262019,
        -0.117153347, -0.0983452052, -0.115111820,
       -0.0873747021,  0.147968352,  0.908112586,
         0.217513233, -0.209576085, -0.00543540716,
         0.967071533,  0.445241243,  0.256844372,
         0.189929992,  0.235132232,  0.387434453,
        -0.397686034,  0.440805733,  0.256803930,
         0.312211454,  0.0688149109,  0.623761535,
         0.0645379499, -0.792526007, -0.389123052,
        -0.273426026,  0.412844241,  0.0325941704,
        -0.250147909, -0.309936047,  0.0207499396,
         0.0883424655, -0.225401282, -0.288145930,
        0.211211652,  0.0771203637,  0.0260510147,
```

```
190         -0.0627066493, -0.0679532737, -0.100084633,
191         -0.275654912, -0.174401820,  0.100622654,
192         -0.282871395, -0.220923603, -0.266388565,
193         -0.295724571, -0.0965403467,  0.0932044387,
194         -0.221140727, -0.185897768, -0.284170061,
195         -0.287501216,  0.0895318985,  0.0891121030,
196          0.235848635,  0.250100285, -0.280651331,
197          0.0453224778,  0.263332158,  0.0912003815,
198          0.303607255,  0.0808357298, -0.277686894,
199         0.226837248,  0.0645613745, -0.0368513465,
200          0.582376540, -0.241064057, -0.0227316022,
201          0.0944749936,  0.165781051, -0.0762444288,
202         -0.333026618, -0.0334155336, -0.550528467,
203          0.279289514, -0.168735445,  0.0835725740,
204          0.210442096,  0.140523076,  0.0503337011,
205          0.257824838, -0.157789290,  0.195312575,
206          0.103272155, -0.170599535, -0.332875282,
207         -0.318359941, -0.154515997,  0.647309065,
208          0.00425511319, -0.170348242, -0.284266353,
209        -0.183652744, -0.161308795,  0.153238639,
210          0.0409132242,  0.254841119, -0.0599049032,
211          0.107347369, -0.0690556690, -0.299018562,
212         -0.0511630923, -0.356937438,  0.333630830,
213         -0.330780536, -0.502203286,  0.335085690,
214          0.341820538,  0.528864205,  0.109793939,
215          0.0476176143,  0.0557929017,  0.0900350437,
216         -0.215066835, -0.0866009593, -0.312297523,
217         -0.322231680, -0.115634307,  0.428616107,
218         -0.178509101, -0.346324950, -0.202630550,
219        -0.0533217490, -0.00950082298, -0.710066319,
220          0.299758673, -0.0603346825, -0.229213983,
221         -1.26745045, -0.0412495844,  0.177363843,
222         -0.262485653,  0.205216929, -0.302785724,
223          0.0811437070, -0.241997361, -0.0127512803,
224         -0.273915708,  0.417487800, -0.142227769,
225         -0.0406281240, -0.288606822,  0.0776360556,
226          0.918415427, -0.892596364, -0.116637953,
227          0.689697206, -0.164608896, -0.214465588,
228          0.497360379,  0.0733343288, -0.00756989792,
229         0.282332927,  0.119788021,  0.100673556,
```

```
230         -0.182732925,  0.171637267, -0.214371502,
231         -0.295446694, -0.306127906,  0.158454001,
232          0.312308162, -0.238908514,  0.102143675,
233         -0.222555056, -0.00921359658,  0.114475042,
234         -0.0379675031, -0.0283490419,  0.0400702655,
235          0.0348899662, -0.101732984,  0.0941225886,
236         -0.124304011, -0.291293859,  0.252318293,
237         -0.00310549140,  0.297123700, -0.0879148990,
238         -0.186512902, -0.141314089,  0.0752194226,
239         0.304583162, -0.197447613, -0.0874307826,
240         -0.382068366,  0.620577216, -0.271324575,
241         -0.224997133, -0.0790816471, -0.0907106847,
242          0.321798921, -0.0234451648, -0.169546008,
243         -0.178972438,  0.199983090,  0.113871850,
244         -0.182000250,  0.700936437,  0.185954899,
245         -0.742182791,  0.366613120,  0.369150847,
246         -0.167834371, -0.580330610,  0.362926751,
247         -0.670174599, -0.145324752, -0.0137709156,
248         -0.556710601, -0.338695914, -0.0723293722,
249        -0.279609680,  0.0164770745,  0.149247646,
250          0.415811062, -0.300519645,  0.0113019347,
251          0.0607311614, -0.589506626, -0.0298319757,
252         -0.985606670,  0.445523560, -0.435654402,
253         -0.231664225, -0.478163153,  0.131433338,
254         -0.293298215,  0.141647086,  0.164143413,
255          0.233625367, -0.585369289,  0.0206025075,
256          0.665482461, -0.336393505,  0.456548780,
257          0.563853741,  0.0840193853, -0.0173224900,
258          0.385888427,  0.0347256772, -0.380864769,
259        -0.270529687, -0.252209753,  0.149664700,
260          0.140218616,  0.179604650, -0.304835200,
261         -0.160391182,  0.139411479, -0.136824191,
262          0.230176538, -0.0437779091,  0.128602609,
263         -0.168866456,  0.147847325, -0.204138428,
264          0.182765707, -0.205500394, -0.0975427926,
265          0.254923016, -0.238473669,  0.00621575117,
266         -0.277945727, -0.195058584, -0.282026291,
267          0.138953626, -0.264379442, -0.00838145614,
268          0.235701531, -0.0544763505,  0.250597179,
269        -0.151500791,  0.339951158, -0.323033929,
```

```
270         -0.269832820, -0.202408686, -0.100598752,
271         -0.143090963,  0.199018836, -0.299257725,
272          0.0919830725,  0.240539804,  0.147105813,
273          0.0249523371, -0.150087133, -0.357052684,
274          0.0643557012, -0.291031033, -0.455335617,
275          0.258808970,  0.579347134,  0.294297576,
276         -0.471072704,  0.439464241,  0.705762982,
277          0.107987948,  0.0918254554, -0.362845808,
278          0.390698522,  0.162841275,  0.0921605974,
279        -0.112641230, -0.0120124500, -0.0808508545,
280          0.351270676, -0.294604123,  0.0112416446,
281          0.113787651,  0.368255913, -0.228453323,
282          0.125900805, -0.437483966,  0.211826369,
283          0.193499357,  0.208535329,  0.114057384,
284         -0.0108327167,  0.651240289,  0.336859137,
285          0.530680418,  0.173431769, -0.270044208,
286         -0.312946260, -0.407573581, -0.619088888,
287         -0.472335488, -0.260036349,  0.280623257,
288         -0.206077874, -0.139005423, -0.300087512,
289         0.0698015094, -0.0537942089,  0.263746142,
290          0.285684139,  0.294513851,  0.0124438703,
291          0.186738536,  0.0776901171, -0.314194143,
292          0.274094015,  0.189218193,  0.300936371,
293         -0.208954841,  0.174457759, -0.0533376969,
294         -0.155664846, -0.0974734575,  0.107002765,
295         -0.277979106, -0.0331280828, -0.111905754,
296          0.105281249,  0.0818539336,  0.185283422,
297         -0.105150029, -0.131591275, -0.241748929,
298         -0.0902168006,  0.271422595, -0.0692179501,
299          0.267219454, -0.228503257,  0.0565484390,
300         -0.126639009,  0.173657537,  0.0768907070,
301          0.341326714, -0.129828021, -0.104052424,
302         -0.486744881, -0.0476464853,  0.0893920138,
303         -0.292037696, -0.628269970, -0.367475301,
304          0.0520940721, -0.261088401, -0.380994678,
305         -0.0855397955, -0.430615813,  0.525783598,
306          0.617678642,  0.563812077, -0.390739799,
307          0.0745991021,  0.0454513319, -0.128620595,
308         -0.0822684541, -0.230701759,  0.522635639,
309        -0.123722494,  0.554619253, -0.632663250,
```

```
310         -0.863205135, -0.427672446, -0.0976152420,
311         -0.387156367,  0.525839865, -0.171724945,
312          0.0935941860, -0.471161306,  0.0131277582,
313         -1.04020298, -0.0390345789, -0.432955056,
314          0.136657432,  0.144220069,  0.00231637200,
315         -0.0459638499,  0.660605729, -0.310778856,
316         -0.252469540, -0.475756735, -0.242361933,
317         -0.201364979,  0.136315733, -0.555002034,
318         -0.539667368, -0.254427820,  0.188294277,
319        -0.142855227, -0.0523813665,  0.236857027,
320         -0.0499605834,  0.113416433,  0.205025882,
321          0.0796217918, -0.231575489,  0.196757525,
322         -0.213316739, -0.185914353,  0.226702064,
323         -0.0836243480,  0.00229275227,  0.223512322,
324         -0.0213340223,  0.0131687820, -0.0868825912,
325          0.105876237,  0.170774668,  0.0991844237,
326         -0.293582618,  0.215085953, -0.129337877,
327         -0.180564269, -0.0868226588, -0.219887525,
328          0.225858897,  0.294650763, -0.303179145,
329        -0.163894683, -0.0178237259, -0.270556509,
330          0.130967379,  0.164406627,  0.0172625780,
331         -0.275703341, -0.212207675,  0.0980207026,
332          0.267705351,  0.262856036,  0.247814804,
333          0.0845529735,  0.107923120, -0.216275752,
334          0.0862820148, -0.262202740,  0.281259090,
335         -0.276082486, -0.0474604368,  0.150595248,
336          0.000668764114, -0.0314964652, -0.201344371,
337         -0.0324859321,  0.0406717658,  0.299231499,
338          0.0803391635, -0.117239311, -0.140133187,
339        -0.296622694, -0.129499614,  0.562723160,
340          0.845134139, -0.826853096, -0.232062235,
341          0.836026371, -0.0782682821,  0.0241927505,
342          0.394105017, -0.732495904,  0.457371891,
343          0.110139363, -0.0225381441,  0.0585553795,
344         -0.183409050,  0.240142837, -0.0744731948,
345          0.145700216, -0.689985394, -1.23122358,
346          0.381210595,  0.237360492, -0.961758256,
347         -0.0585413352,  0.157870889,  1.35860121,
348         -0.101651892, -0.0593722127, -0.897759199,
349        -0.274370283,  0.100802712, -0.524549663,
```

```
350          0.00213904539,  0.0625524074, -0.0650027096,
351         -0.158722281, -0.616041243, -0.223961994,
352          0.0528887287, -0.0644629076, -0.0474635363,
353          0.137488246, -0.744052947, -0.488124043,
354         -0.224958673,  0.264175713, -0.149580956,
355         -0.240009308,  0.298128814,  0.234064832,
356          0.341636330, -0.130877972, -0.282176554,
357         -0.327877104, -0.103728555, -0.0469306894,
358         -0.276949495, -0.364828497,  0.483478308,
359        -0.239167124, -0.189229354, -0.0127724214,
360          0.152656898,  0.141517356,  0.194823056,
361         -0.0254084282, -0.125816256,  0.0489594340,
362         -0.0258922316,  0.202638254, -0.572419107,
363          0.451733887,  0.174334288,  0.0336504802,
364          0.00813724659,  0.414302677, -0.148746565,
365         -0.290528268, -0.187081173, -0.0944865048,
366          0.178078935, -0.373078108,  0.220176071,
367          0.205438346, -0.311864138, -0.0572340228,
368          0.110308304, -0.163399488,  0.0469085649,
369        -0.0487752259, -0.0751434192,  0.189356342,
370         -0.107958965,  0.547334254,  0.0427899659,
371          0.0655148998, -0.0576960370,  0.00684061646,
372          0.00258211372, -0.241230607, -0.196445659,
373         -0.267293543, -0.0507498235, -0.0950309709,
374         -0.416139573,  0.0648119003,  0.303122252,
375          0.0128421308, -0.0992971659,  0.123085290,
376          0.359522045,  0.0904989615,  0.0623895824,
377         -0.282749116, -0.106746860, -0.0254473351,
378         -0.0166549310,  0.224602357,  0.311177164,
379        -0.269848645,  0.374163866,  0.130077854,
380         -0.234649196,  0.441053778, -0.206333458,
381         -0.293409497,  0.420860618,  0.00982859731,
382         -0.00502824830,  0.0461090803, -0.109238796,
383         -0.213333800,  0.143931687,  0.00842457172,
384         -0.384009391,  0.299022913,  0.231425077,
385         -0.217860833, -0.0861599892,  0.454165488,
386          0.239649549, -0.0133806793,  0.313762277,
387          0.128026903, -0.110743329, -0.273553342,
388          0.214879677, -0.144800648, -0.103683010,
389        -0.0293576717, -0.121533841,  0.0413068719,
```

```
390         -0.250568926, -0.106697932, -0.292615771,
391         -0.125035465,  0.389022261,  0.271028668,
392          0.322730511,  0.0877427980,  0.119882092,
393          0.0193562862,  0.0897189528,  0.0553621575,
394         -0.136963233,  0.769835413,  0.165531039,
395         -0.468940139,  0.532243907,  0.0316210315,
396         -0.672640979, -0.309009343,  0.483948678,
397         -0.718974054, -0.163154691,  0.0560416467,
398         -0.378287971, -0.0202501956,  0.102845438,
399        -0.116758823,  0.333407432, -0.326286942,
400         -0.298276871, -0.265357852, -0.291530013,
401         -0.0545706823, -0.139038101, -0.0849839300,
402         -0.136839926,  0.284863204, -0.0601268783,
403          0.212323919,  0.118619889, -0.467744619,
404          0.187889621,  0.0960171819, -0.274786055,
405          0.0491847917,  0.248664424, -0.112274297,
406          0.208710149,  0.321335822,  0.427963376,
407         -0.116209172,  0.200526744, -0.107630007,
408          0.167456612, -0.223468915,  0.210141644,
409        -0.0492959023, -0.0916918740, -1.05251539,
410          0.122334145, -0.112369090,  0.0344708562,
411         -0.476014227,  0.0393678062, -0.289736509,
412         -0.321285307,  0.159524575, -0.122347608,
413          0.315438747,  0.331916809, -0.418459892,
414          0.133053809,  0.371791273,  0.202320129,
415          0.104625039,  0.697545350, -0.718468010,
416         -1.01826453,  0.169168308, -0.00139994745,
417          0.250735670, -0.203595579, -0.572906315,
418          0.133968621,  0.147972569,  0.176488191,
419        -0.110667020,  0.297968030,  0.187498018,
420          0.241570741,  0.503271401, -0.149697885,
421         -0.218484968,  0.177954897, -0.0612788200,
422         -0.0562011302,  0.102980942, -0.824717462,
423         -0.305707395,  0.113495573, -0.0142401401,
424          0.0364527516,  0.412690878,  0.134613708,
425         -0.189708129, -0.196268007, -0.0792878568,
426          0.745337725, -0.663768828,  0.666644275,
427          0.359014362, -0.167081222,  0.336243749,
428          0.151140153, -0.154609695, -0.152835295,
429        -0.170285136, -0.102274463, -0.00616447302,
```

```
430         -0.490923226,  0.0974962413, -0.228806168,
431         -0.0741197690, -0.787400007, -0.262198120,
432         -0.122844495, -0.110070065, -0.210314006,
433          0.184123635, -0.278937936,  0.211624786,
434          0.392525494, -0.269278288, -0.313578695,
435          0.521205127, -0.262114018,  0.104414076,
436          0.200612128,  0.918891191, -0.307897717,
437          0.0431581140,  0.265394986,  0.352968186,
438          0.314736128, -0.251375973,  0.196009502,
439         0.302877098, -0.105359823, -0.614104629,
440          1.00303459, -0.327588677,  0.106139511,
441         -0.0203447621, -0.171524063, -0.0676171035,
442         -0.0394094363, -0.173390716, -0.0799034089,
443          0.119138606, -0.155059099, -0.0800225809,
444         -0.0257932600, -0.0741345212,  0.398476183,
445          1.05520451,  0.440288872, -0.264466137,
446         -0.0307008140, -0.618715882, -0.133247942,
447         -0.946738183, -0.236024544,  0.659539998,
448          0.328422040, -0.147317678,  0.478137851,
449         0.277739435, -0.267260522,  0.525883853,
450          0.311923623, -0.516963005, -0.200540066,
451          0.481274694,  0.428421617, -0.0233886838,
452          0.387141883,  0.253997445,  0.514194429,
453          0.129308805,  0.458666891,  0.0742752254,
454         -0.497586280,  0.429364443,  0.0554989278,
455          0.443339258, -0.540331602, -0.799214900,
456         -0.422660559, -0.508995891,  0.280602664,
457          0.188810334,  0.0930831209,  0.974216282,
458         -0.290050179,  0.0538487695, -0.949275732};
459
460
461 float pesos2[30][22]= { 0.156856477,  0.0134109259,  0.313806653,
462         -0.300077915,  0.123380661, -0.241035491,
463          0.167494476,  0.193274736,  0.283395112,
464          0.167672157, -0.0800651312, -0.0737966895,
465         -0.203785628,  0.184476256,  0.152219743,
466          0.261191726,  0.131185770,  0.121684790,
467          0.325622201, -0.0549691319, -0.0755813122,
468         -0.282753885,
469        -0.119054548,  0.00817816518,  0.179514036,
```

```
470          -0.205788076,  0.0963532105,  0.266023785,
471          -0.0534114949, -0.0658484325, -0.0406206734,
472           0.142152041, -0.202189997, -0.181229681,
473           0.158364758, -0.344452739, -0.0332988910,
474          -0.445869058, -0.170688897,  0.223440796,
475          -0.889137983, -0.931044638,  0.235289529,
476           0.266187042,
477         -0.360791475, -0.0321586356,  0.0937680379,
478           0.0838299319,  0.588401854,  0.928224683,
479           0.342942655, -0.212655500, -0.482138097,
480          -0.916176915, -0.248029932, -1.53074801,
481           0.589338899,  0.786488950, -0.904850364,
482           0.713364959,  0.0248732883, -1.08648765,
483          -0.0841439813, -0.0527592562,  0.0347303748,
484           0.0589474961,
485         -0.360159993, -0.0907842442, -0.469152302,
486          -0.754851520,  0.0778510571, -1.38657939,
487          -0.100698195, -0.434463352, -1.01319301,
488           0.205451831, -1.97374201, -1.80359435,
489          -1.23123121,  0.487982064, -2.26637721,
490          -0.144753516, -2.79040837, -1.06477559,
491           0.430303007,  0.544200003, -2.44851065,
492           0.0670012608,
493         -0.309847206, -0.439135492, -0.637680352,
494           0.270081848,  0.554883659, -0.308765918,
495          -0.197622895,  0.646503031,  0.473167419,
496           0.242724150,  0.117234252,  0.0764697120,
497          -0.417938262, -1.57022810, -0.375598252,
498           0.254102647, -0.0384471789,  0.506240547,
499          -0.350751311, -0.0605403073, -0.571280360,
500           0.0533770062,
501          0.259203434, -0.0882267356,  0.325523496,
502           0.0751249492,  0.150799632,  0.0473681092,
503           0.213325441,  0.180813491, -0.277052164,
504           0.159343958,  0.153225094,  0.174530625,
505          -0.0598050952,  0.0170993805,  0.335029125,
506           0.0169588625,  0.232101321, -0.230367988,
507          -0.223499656, -0.148220763,  0.107036412,
508          -0.0494724810,
509         -2.04846382, -0.287019193, -0.278582484,
```

```
510          0.111767009, -0.298566908,  0.0949019790,
511          0.248479933, -0.533497393, -2.06501436,
512          0.0288336612, -0.173011556, -0.563310862,
513          0.471263617,  0.265219390,  0.0213218071,
514          0.321773022,  0.384359062, -0.666494489,
515         -0.103333250, -0.0745385736,  0.741522133,
516          0.128788933,
517        -1.32867754, -0.149702549, -1.00396848,
518         -1.13628900, -0.739848077,  0.819538653,
519         -0.0859504119, -1.44986045,  0.188521028,
520         -0.0285090413, -0.202212289, -0.239370868,
521          0.217271298,  0.198350474,  0.103627160,
522         -1.61131811, -0.0817205310, -0.183692172,
523         -0.191518366, -0.861683846,  0.926538110,
524          0.0980190113,
525        -0.185035482, -0.183590040,  0.264804304,
526          0.295679033, -0.183336541, -0.184853747,
527         -0.104414374,  0.253885269,  0.334476948,
528         -0.221682072, -0.321753770,  0.0229237080,
529         -0.325739712, -0.0390822589, -0.186909437,
530          0.112101465, -0.218726635, -0.282057941,
531          0.0250279009,  0.117737919, -0.142459035,
532          0.0392785072,
533         0.174101412, -0.226342514, -0.565795004,
534         -0.0678735301,  0.130899623, -0.347104102,
535         -0.0360725857, -0.880612969, -0.0596495047,
536          0.0270414036,  0.162268788,  0.126531169,
537          0.231122971, -0.162322059,  0.183321893,
538          0.436032802, -0.365735203, -0.502993286,
539         -0.994059741, -0.879374027, -0.0355955660,
540         -0.657600522,
541         0.263170838,  0.460327923,  0.119020283,
542          0.0111001898, -0.0334875621, -0.487391144,
543          0.114327401,  0.142314255, -0.205354750,
544          0.206336781, -1.18081391, -2.16507459,
545         -0.534034908, -1.49382424,  0.104358032,
546         -0.787072718, -2.02784204, -1.94517529,
547         -0.00786090363, -0.0390309617,  0.0234329477,
548          0.477958769,
549         0.167657867, -0.661721051, -0.231993645,
```

```
550         -0.0808910206, -0.650638282, -0.701779664,
551          0.486726582, -0.824228585, -0.629496455,
552         -0.379346430,  0.349078536,  0.426926970,
553          0.167036086,  0.00139091013,  0.106785275,
554          0.140809700,  0.00688993325, -0.418503404,
555         -0.160079271, -0.270978630,  0.298599482,
556          0.0757959038,
557        -0.238073468, -0.168616235, -0.448751688,
558         -0.174507350, -0.00637670001, -0.546452940,
559         -1.27735972, -0.786334932, -0.404807389,
560         -0.321322113, -1.63830066,  0.363643587,
561         -0.369137883, -1.36427116, -0.699641168,
562         -1.20276284, -1.36980021, -0.0762889311,
563          0.334887117,  0.547550917, -0.845480919,
564         -0.528903842,
565        -0.862924874,  0.0466885567, -0.802209675,
566         -1.37080419, -0.525331974,  0.692635775,
567         -0.110072531, -0.406341106, -0.376287013,
568         -0.186153114,  0.00608463772,  0.156244233,
569          0.281175613, -0.170418590,  0.162260070,
570         -0.301170677, -0.0290512256,  0.0174913667,
571          0.0500049293, -1.05462837,  0.0203077346,
572          0.0110524129,
573        -0.0535513014, -0.0209056810, -0.451663882,
574         -0.821024358, -0.357449442,  0.304531872,
575          0.160665318, -0.452233732, -0.517028987,
576         -0.281183720,  0.227577716,  0.0109452149,
577          0.199999183,  0.0499633923, -0.138951778,
578         -0.219516695, -0.188305840, -0.275364161,
579         -0.386826694, -1.11896944,  0.124387659,
580         -0.323153883,
581         0.317891300, -0.339134187,  0.233313411,
582          0.0622276478, -0.720278502, -0.912401676,
583          0.190697044, -0.0414014682, -0.453695625,
584         -0.301528931,  0.158928558,  0.153256074,
585         -0.160028428, -0.220772818,  0.223618805,
586         -0.106205389,  0.288845122,  0.0464417264,
587          0.282215834, -0.168293193,  0.0981160849,
588         -0.0148630841,
589         0.167673826, -0.711074948, -0.241479784,
```

```
590         -0.487407476,  0.306848943,  0.234255522,
591         -0.335994601, -0.149979472,  0.125441253,
592          0.241979837,  0.531775355,  0.636248827,
593          0.333849996, -0.611099899,  0.154305205,
594          0.140817091,  0.0547897965,  0.266704351,
595         -0.241988987,  0.382309049, -0.364754468,
596          0.116650946,
597         0.151526034, -0.495454311, -0.0517557785,
598         -0.246881559,  0.680514872, -0.0698960647,
599          0.151328117, -0.453758895,  0.315659732,
600          0.343013316,  0.141154721,  0.424230576,
601          0.0982683450, -0.772600293, -0.0118307639,
602          0.147279888, -0.314908773, -0.262060970,
603         -0.484094083, -1.05448639, -0.384713709,
604         -0.0293389633,
605        -0.470469594,  0.696611583, -0.863559961,
606          0.0577461198, -0.871411979, -0.0421082191,
607          0.533488154, -0.194329277, -0.763382316,
608          0.170130938, -1.36837280, -0.111898750,
609         -0.0501095355,  0.170984402,  0.324526846,
610         -0.667212963, -0.412693441, -0.731994867,
611          0.231372610,  0.109730259,  0.375045985,
612          0.406311244,
613         0.0764690042, -0.726523817,  0.131091937,
614         -0.324576050,  0.704911828,  0.526001215,
615         -0.745682538,  0.120757177, -2.77956486,
616         -0.778810024, -2.68450046, -1.50597930,
617         -0.403703213, -0.452161580, -0.618003845,
618          0.472341239, -0.941112697, -2.34110785,
619          0.155358002,  0.485461593,  0.386780322,
620         -2.93996763,
621         0.0336980522,  0.286566108, -0.410683334,
622          0.791713655, -0.0489855334, -0.814078331,
623         -0.700204253,  1.12896585,  0.697202742,
624          0.539316773, -0.449429452, -1.13306272,
625         -1.47565997, -1.09291613, -0.983191788,
626          0.181879699, -0.0204234105,  0.769889176,
627         -0.258361757,  0.368237853, -1.16195810,
628          0.329126894,
629        -0.606043041,  0.208513245, -0.503186464,
```

```
        -0.0356369615,  0.235496357,  0.657189071,
        -0.359805763,  0.515336037,  0.355253577,
        -2.36007428, -0.812481821, -1.51404524,
        -0.900796056,  0.347225577, -1.71777368,
         0.351540148,  0.513000369,  0.293114424,
         0.309292912,  0.315509260,  0.277475029,
         0.171469569,
        0.261547118,  0.375163585,  0.609782100,
         1.11215413, -2.72792125, -0.878865898,
         0.694771409, -0.230491444, -0.941118419,
        -1.03218102, -0.727091908, -0.266593248,
        -0.732137561,  0.144030035,  0.173861429,
         0.0309504792,  0.454551607,  0.355852336,
         0.0324732848, -0.994953156,  0.185454100,
        -0.295670420,
        0.233250767,  0.239816114,  0.0655916110,
        -0.198020667,  0.320367664, -0.226230189,
        -0.118390441, -0.0250766110,  0.656280816,
         0.619720995,  0.189526260, -1.18087661,
        -3.29405355, -1.69295561,  0.410485417,
        -0.673619032, -1.48636484, -0.763676584,
        -0.804265440,  0.219354659, -0.0931643695,
         0.562295854,
        0.148334667,  0.813470185,  0.158511475,
        -0.205254331, -0.623724341, -3.41700053,
        -0.364005655, -0.251824051, -0.526885867,
         0.264143795, -0.140463799, -0.342512548,
        -0.316706032, -1.11210549,  0.221237734,
        -1.40474176, -3.40544605, -1.50540853,
         0.266001612,  0.571652472, -2.63893747,
        -1.72270393,
        0.251765788,  0.105123214, -0.324404985,
         0.0593581945,  0.0700410679, -0.293653846,
         0.0844007879,  0.0588349216,  0.0123908892,
         0.163215712, -0.161383286, -0.133786678,
         0.186106309, -0.278943151, -0.0714680403,
        -0.266053021, -0.0820417255,  0.0944759026,
         0.0378546417, -0.130209535,  0.158543855,
        -0.122348472,
       -0.579232216, -1.36585140, -2.25135899,
```

```
670         -1.81380332, -0.455926538,  0.556649029,
671         -0.0267609730, -1.53022885, -0.251107603,
672          0.327946126,  0.567028761,  0.779912651,
673          0.650483251,  0.0932623744,  0.520383477,
674         -0.555876553,  0.108500168, -0.238227740,
675          0.215466440, -0.756538928,  0.697811365,
676          0.0841674358,
677        -0.0327422209,  0.583919764,  0.0220430586,
678          0.221243292, -0.250638902, -1.72836328,
679          0.182287425,  0.396549642, -0.0442214087,
680         -0.544503570, -0.987643361, -0.390023172,
681         -0.581785560, -0.477073759, -0.884607792,
682         -0.515535593, -1.00041974,  0.0739071071,
683          0.125386283, -0.189593479, -1.74961030,
684         -3.53517222,
685        -0.0455291681, -0.304109782, -0.0232152361,
686          0.132260829, -0.375222653, -0.248933837,
687         -0.289254576, -0.314214885, -0.0365596004,
688          0.108011998, -0.313036531,  0.0354209431,
689         -0.109305777,  0.0247605555, -0.299110502,
690         -0.0303202048, -0.0246428307,  0.0499036498,
691         -0.305287957, -0.0419930555, -0.0958675668,
692         -0.192023858,
693        -0.543451488, -0.300668538,  0.436871648,
694          0.354986489, -0.806060612,  0.219163775,
695          0.0139113432,  0.365272820,  0.282707632,
696         -0.878507376, -0.946661353, -0.547891080,
697         -0.836115360, -0.0264956560, -2.25859451,
698          0.695635319, -0.422194809,  0.702962041,
699          0.256164938,  0.152642384, -0.442035705,
700         -2.52015066};
701
702
703 //********  feedforward    *********
704 // Entradas dos 22 sinais coletados no momento de aquisição do dataset
705 //ENTRADAS[1][11]
706 //{AcX,AcY,AcZ,GyX,GyY,GyZ,polegar,indicador,medio,anelar,mindinho}
707 //obrigado    (1)
708 //float camadaEntrada[1][11]={64732,49552,62484,65457,65400,
709 //65468,2331,2830,2869,2383,2032};
```

```
710 //u           (2)
711 //float camadaEntrada[1][11]={588,49320,62412,32,65340,65455,
712 //2023,3055,3309,890,1314};
713 //n           (3)
714 //float camadaEntrada[1][11]={61812,12076,55920,65414,65124,
715 /65510,1754,3805,2659,877,1264};
716 // b          (4)
717 //float camadaEntrada[1][11]={65240,49788,61600,65523,65413,
718 //65448,928,3767,2907,1901,2439};
719 // oi         (5)
720 //float camadaEntrada[1][11]={65000,49644,61788,30,65393,
721 //65481,993,938,1104,712,2875};
722 //tudo bem?   (6)
723 //float camadaEntrada[1][11]={51264,63896,59052,38,35,
724 //65407,1842,1456,1340,1072,1097};
725 // tchau      (7)
726 //float camadaEntrada[1][11]={1344,49588,62496,65341,
727 //567,65534,1968,3287,3035,2229,2419};
728 //Deus        (8)
729 //float camadaEntrada[1][11]={1528,49312,63484,65383,
730 //65533,13,1260,3735,1192,808,1061};
731 //Abraço      (9)
732 //float camadaEntrada[1][11]={53224,55788,62288,65507,
733 //65421,65533,992,976,1193,904,1113};
734 //Desculpa    (10)
735 //float camadaEntrada[1][11]={59536,50352,65492,65460,
736 //65403,65473,1968,1511,1438,1021,2935};
737 //Brincar     (11)
738 //float camadaEntrada[1][11]={55260,54928,59396,65508,
739 //65343,33,1973,1449,1286,1105,2269};
740 //Barato      (12)
741 //float camadaEntrada[1][11]= {60336,56592,53140,65445,
742 //65302,65491,2034,1921,1979,1518,1895};
743 //Café        (13)
744 //float camadaEntrada[1][11]={57504,61000,52280,65451,
745 //65435,65528,1744,1623,2903,2335,2871};
746 // Dormir     (14)
747 //float camadaEntrada[1][11]={6860,51144,5224,65529,
748 //65330,65464,1791,4095,3823,1186,1122};
749 //Futuro      (15)
```

```
750 //float camadaEntrada[1][11]={63644,49688,62008,21,
751 //65389,65509,2803,2307,3053,2138,2743};
752 //Fácil       (16)
753 //float camadaEntrada[1][11]={756,50020,60688,65520,
754 //223,75,1965,4095,1535,2079,2675};
755 //Dois        (17)
756 //float camadaEntrada[1][11]={51932,62300,58460,65379,
757 //65288,65517,1802,3381,1344,1140,1152};
758 // Musica     (18)
759 //float camadaEntrada[1][11]={52968,58692,58504,65532,
760 65466,65396,1069,3303,1058,744,971};
761 // Igual      (19)
762 //float camadaEntrada[1][11]={236,50296,59196,64935,
763 //229,65126,2545,3411,2651,1138,1185};
764 //Inteligente(20)
765 //float camadaEntrada[1][11]={64956,49456,62592,65525,
766 //65472,65430,1722,1367,1285,1083,1044};
767 // Luva       (21)
768 //float camadaEntrada[1][11]={60064,10260,54200,65399,
769 //1663,62443,2192,3583,2941,1967,2599};
770 //Sinal      (22)
771 //float camadaEntrada[1][11]={51892,56932,63860,65472,
772 //65458,65476,1985,1388,1292,1200,1353};
773
774
775
776 // Variáveis
777 // Saídas da Função de Ativação nas Camadas
778 float Sij[1][30],Sjk[1][30],Ssaida[1][22];
779 //Resultado do somatório em forma de Vetor
780 float somaSinapse0[1][30],somaSinapse1[1][30],somaSinapse2[1][22];
781 // Variáveis auxiliar
782 //Do tipo Int
783 int i,j,k;
784 //Do tipo Float
785 float sum,denominador;
786 int L=1,M=11,C=30;
787 // Camada de Entrada
788 float camadaEntrada[1][11]={0,0,0,0,0,0,0,0,0,0,0};
789 // Para Normalizar
```

```
790 float camadaEntrada_Norm[1][11]={0,0,0,0,0,0,0,0,0,0,0};
791
792
793 // Variáveis Globais do MPU-6050
794 //Endereco I2C do MPU6050
795 const int MPU=0x68;
796 //Variaveis para armazenar valores dos sensores
797 int AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
798 // Variáveis Globais dos sensores Flexíveis
799 float dedoPol,dedoInd,dedoMed,dedoAnel,dedoMind;
800
801
802
803
804 void setup() {
805   // Taxa de transmissão tipo, RX, TX
806   mySoftwareSerial.begin(9600, SERIAL_8N1,RX2,TX2);
807
808   // Iniciar comunicação serial para fins de depuração
809   Serial.begin (115200);
810
811   // Se o módulo não estiver conectado, informara que houve um problema
812   if (!myDFPlayer.begin(mySoftwareSerial)) {
813   // Verifica o funcionamento do módulo
814     Serial.println(myDFPlayer.readType(), HEX);
815     Serial.println(F("Erro ao iniciar, verifique:"));
816     Serial.println(F("1.A conexao do modulo."));
817     Serial.println(F("2.Se o SD card foi inserido corretamente."));
818     while (true);
819   }
820   //Caso contrário, continua a execução do código
821   Serial.println(F("DFPlayer Mini online."));
822   //----Tempo de comunicação----
823   myDFPlayer.setTimeOut(500);
824
825
826   //----Define o dispositivo----
827   myDFPlayer.outputDevice(DFPLAYER_DEVICE_SD);
828
829   //----Quantidade de Arquivos----
```

```
830   Serial.print(F("Arquivos encontrados no cartao SD: "));
831   Serial.println(myDFPlayer.readFileCounts(DFPLAYER_DEVICE_SD));
832
833   //----Ajusta o Equalizador e inicia com a primeira música----
834    // 0 - Normal, 1 - Pop, 2 - Rock, 3 - Jazz, 4 - Classic e 5 - Bass
835   myDFPlayer.EQ(2);
836     //----Ajuste do Volume----
837   myDFPlayer.volume(30);  //Volume de 0 á 30.
838
839   //Definindo os sensores flexíveis como entrada
840   pinMode(polegar,INPUT);
841   pinMode(indicador,INPUT);
842   pinMode(medio,INPUT);
843   pinMode(anelar,INPUT);
844   pinMode(mindinho,INPUT);
845
846   // Ajustando MPU-6050 de acordo com a calibração
847   mpu.setXAccelOffset (-196);
848   mpu.setYAccelOffset (-2711);
849   mpu.setZAccelOffset (5192);
850
851   mpu.setXGyroOffset (31);
852   mpu.setYGyroOffset (5);
853   mpu.setZGyroOffset (8);
854   Wire.begin(SDA,SCL);
855   Wire.beginTransmission(MPU);
856   Wire.write(0x6B);
857
858   //Inicializa o MPU-6050
859   Wire.write(0);
860   Wire.endTransmission(true);
861 }
862
863 void loop() {
864
865 // Adquirindo os dados do sensor MPU-6050 (Variáveis de Entrada)
866   //PARTE DO MOVIMENTO
867
868   //Inicializando a transmissão
869   Wire.beginTransmission(MPU);
```

```
870   // Inicializando com o registrado 0x3B (ACCEL_XOUT_H)
871   Wire.write(0x3B);
872   //Verificando o funcionamento
873   Wire.endTransmission(false);
874
875   //Solicita os dados do sensor
876   Wire.requestFrom(MPU,14,true);
877
878   //Armazena o valor dos sensores nas variaveis correspondentes
879   AcX=Wire.read()<<8|Wire.read(); //0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
880   AcY=Wire.read()<<8|Wire.read(); //0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
881   AcZ=Wire.read()<<8|Wire.read(); //0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
882   //Tmp=Wire.read()<<8|Wire.read(); //0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
883   GyX=Wire.read()<<8|Wire.read(); //0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
884   GyY=Wire.read()<<8|Wire.read(); //0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
885   GyZ=Wire.read()<<8|Wire.read(); //0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
886
887 //Adquirindo os dados do sensor MPU-6050 (Variáveis de Entrada)
888   dedoPol=analogRead(polegar);
889   dedoInd=analogRead(indicador);
890   dedoMed=analogRead(medio);
891   dedoAnel=analogRead(anelar);
892   dedoMind=analogRead(mindinho);
893
894 //***** REDE NEURAL PROCESSANDO OS DADOS (CLASSIFICANDO)  ******
895
896   // Atribuindo os valores a um vetor que representa a entrada da RNA
897   camadaEntrada[0][0]=AcX;
898   camadaEntrada[0][1]=AcY;
899   camadaEntrada[0][2]=AcZ;
900   camadaEntrada[0][3]=GyX;
901   camadaEntrada[0][4]=GyY;
902   camadaEntrada[0][5]=GyZ;
903   camadaEntrada[0][6]=dedoPol;
904   camadaEntrada[0][7]=dedoInd;
905   camadaEntrada[0][8]=dedoMed;
906   camadaEntrada[0][9]=dedoAnel;
907   camadaEntrada[0][10]=dedoMind;
908
909
```

```
910 // Para Debugar
911   Serial.print("Camada de Entrada: \n");
912   for(i = 0; i < 1; i++){
913       for(j = 0; j < 11; j++){
914           Serial.print(camadaEntrada[i][j]);
915           Serial.print("\t");
916       }
917       Serial.print("\n");
918   }
919 
920 //********  REDE NEURAL PROCESSANDO OS DADOS (CLASSIFICANDO)  **********
921   //Pré-processando o dado com a normalização
922   for(i = 0; i < 1; i++)
923   {
924       for(j = 0; j < 11; j++)
925       {
926           camadaEntrada_Norm[i][j] = camadaEntrada[i][j]/2500.0;
927       }
928   }
929 
930   //Entrada da Rede Neural
931   //Multiplicar Matriz A com Matriz B
932   for(i = 0; i < 1; i++) {
933      for(j = 0; j < 30; j++) {
934         sum=0;
935         for( k = 0; k < 11; k++) {
936           sum=sum+(camadaEntrada_Norm[i][k]*pesos0[k][j]);
937           somaSinapse0[i][j]=sum;
938         }
939      }
940   }
941 
942   //Passando pela a função de ativação RELU
943   ///RELU
944   for(i = 0; i < L; i++)
945   {
946     for(j = 0; j < C; j++)
947     {
948       if(somaSinapse0[i][j] > 0){
949         Sij[i][j] = somaSinapse0[i][j];
```

```
950       }
951       else
952       {
953         Sij[i][j] = 0;
954       }
955     }
956   }
957   //RELU
958
959   L=1;M=30;C=30;
960
961   //Entrando na 1º camada oculta
962   for(i = 0 ; i < L; i++) {
963      for(j = 0; j < C; j++) {
964         sum=0;
965         for(k = 0; k < M; k++) {
966           sum=sum+(Sij[i][k]*pesos1[k][j]);
967           somaSinapse1[i][j]=sum;
968         }
969      }
970   }
971
972   //Passando pela função de ativação RELU
973   ///RELU
974   for(i = 0; i < L; i++)
975   {
976     for(j = 0; j < C; j++)
977     {
978       if(somaSinapse1[i][j] > 0){
979         Sjk[i][j] = somaSinapse1[i][j];
980       }
981       else
982       {
983         Sjk[i][j] = 0;
984       }
985     }
986   }
987   //RELU
988
989   L=1;M=30;C=22;
```

```
990
991   //Entrando na 2º camada oculta
992   for(i=0;i<L;i++) {
993      for(j=0;j<C;j++) {
994         sum=0;
995         for(k=0;k<M;k++) {
996           sum=sum+(Sjk[i][k]*pesos2[k][j]);
997           somaSinapse2[i][j]=sum;
998         }
999      }
1000  }
1001
1002  //Passando pela função de ativação softmax
1003  // SOFTMAX
1004  //Calculando o denominador
1005  denominador=0;
1006  for(i = 0; i < L; i++)
1007  {
1008    for(j = 0; j < C; j++)
1009    {
1010      denominador +=pow(M_E,somaSinapse2[i][j]);
1011    }
1012  }
1013
1014  for(i = 0; i < L; i++)
1015  {
1016    for(j = 0; j < C; j++)
1017    {
1018      //np.exp(soma)/denominador
1019      Ssaida[i][j] = pow(M_E,somaSinapse2[i][j])/denominador;
1020    }
1021  }
1022  //SOFTMAX
1023
1024
1025  Serial.print("Saida da Funcao da camada de saida: \n");
1026  for(i = 0; i < L; i++){
1027      for(j = 0; j < C; j++){
1028          Serial.print(Ssaida[i][j]);
1029          Serial.print("\t");
```

```
1030        }
1031        Serial.print("\n");
1032    }
1033
1034   //Atuação com Sistema de Áudio
1035     if (Ssaida[0][0] >= 0.7)
1036    {
1037      Serial.println("---OBRIGADO---");
1038      myDFPlayer.play(22);  //Le o primeiro arquivo no cartão
1039      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1040    }
1041    if (Ssaida[0][1] >= 0.7)
1042    {
1043      Serial.println("---U---");
1044      myDFPlayer.play(21);  //Le o segundo arquivo no cartão
1045      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1046    }
1047
1048
1049    if (Ssaida[0][2] >= 0.7)
1050    {
1051      Serial.println("---N---");
1052      myDFPlayer.play(20);  //Le o terceiro arquivo no cartão
1053      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1054    }
1055
1056    if (Ssaida[0][3] >= 0.7)
1057    {
1058      Serial.println("---B---");
1059      myDFPlayer.play(19);  //Le o quarto arquivo no cartão
1060      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1061    }
1062
1063    if (Ssaida[0][4] >= 0.7)
1064    {
1065      Serial.println("---oi---");
1066      myDFPlayer.play(18);  //Le o quinto arquivo no cartão
1067      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1068    }
1069
```

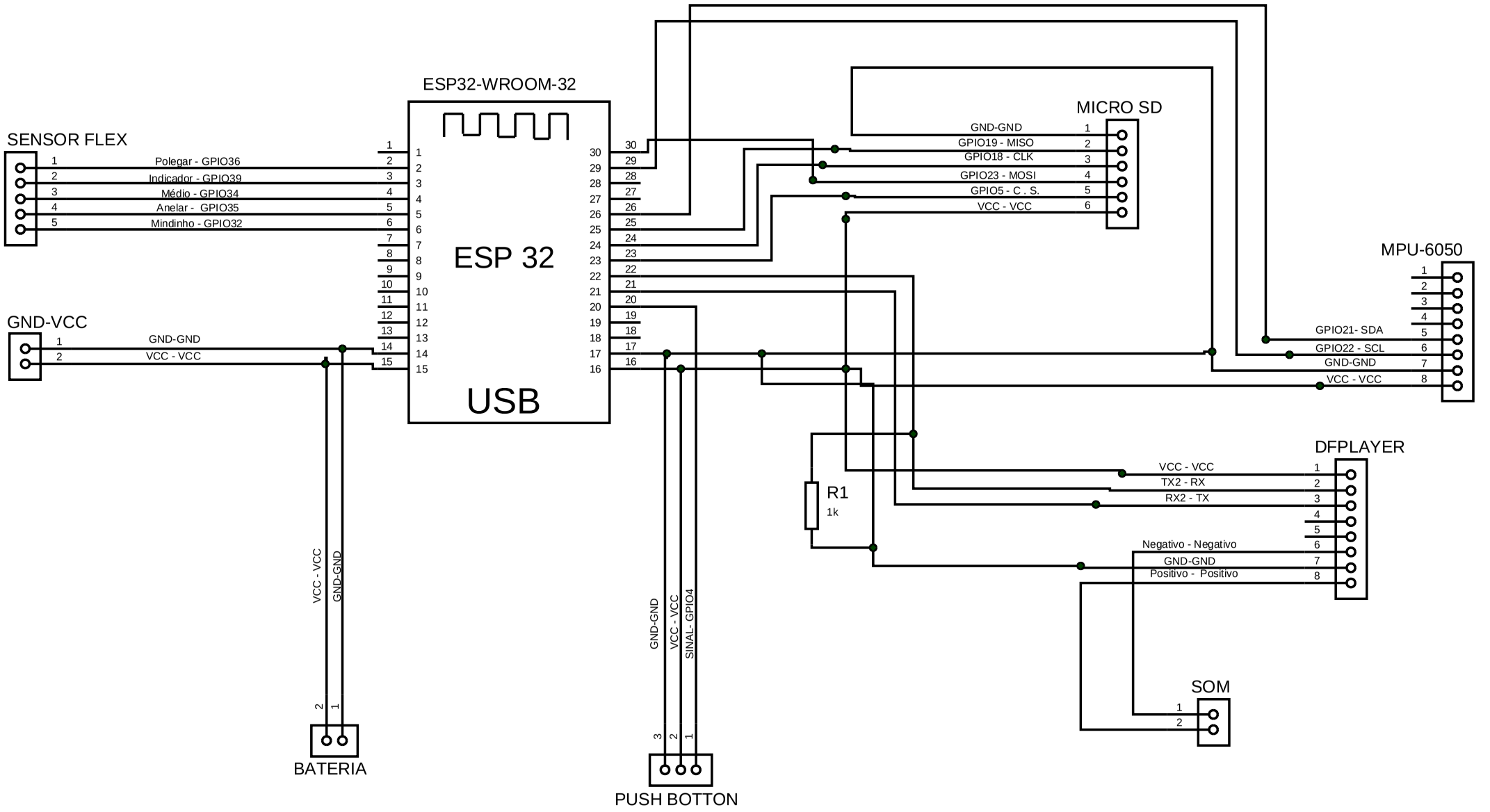
```
1070      if (Ssaida[0][5] >= 0.7)
1071    {
1072      Serial.println("---Tudo bem?---");
1073      myDFPlayer.play(17);  //Le o sexto arquivo no cartão
1074      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1075    }
1076
1077      if (Ssaida[0][6] >= 0.7)
1078    {
1079      Serial.println("---Tchau!---");
1080      myDFPlayer.play(16);  //Le o sétimo arquivo no cartão
1081      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1082    }
1083
1084      if (Ssaida[0][7] >= 0.7)
1085    {
1086      Serial.println("---Deus---");
1087      myDFPlayer.play(15);  //Le o oitavo arquivo no cartão
1088      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1089    }
1090
1091
1092    if (Ssaida[0][8] >= 0.7)
1093    {
1094      Serial.println("---Abraço---");
1095      myDFPlayer.play(14);  //Le o nono arquivo no cartão
1096      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1097    }
1098
1099
1100    if (Ssaida[0][9] >= 0.7)
1101    {
1102      Serial.println("---Desculpa---");
1103      myDFPlayer.play(13);  //Le o décimo arquivo no cartão
1104      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1105    }
1106
1107    if (Ssaida[0][10] >= 0.7)
1108    {
1109      Serial.println("---Brincar---");
```

```
1110      myDFPlayer.play(12);  //Le o décimo primeiro arquivo no cartão
1111      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1112    }
1113
1114    if (Ssaida[0][11] >= 0.7)
1115    {
1116      Serial.println("---Barato---");
1117      myDFPlayer.play(11);  //Le o décimo segundo arquivo no cartão
1118      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1119    }
1120
1121      if (Ssaida[0][12] >= 0.7)
1122    {
1123      Serial.println("---Café---");
1124      myDFPlayer.play(10);  //Le o décimo terceiro arquivo no cartão
1125      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1126    }
1127
1128      if (Ssaida[0][13] >= 0.7)
1129    {
1130      Serial.println("---Dormir---");
1131      myDFPlayer.play(9);  //Le o décimo quarto arquivo no cartão
1132      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1133    }
1134
1135     if (Ssaida[0][14] >= 0.7)
1136    {
1137      Serial.println("---Futuro---");
1138      myDFPlayer.play(8);  //Le o décimo quinto arquivo no cartão
1139      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1140    }
1141    if (Ssaida[0][15] >= 0.7)
1142    {
1143      Serial.println("---Facil---");
1144      myDFPlayer.play(7);  //Le o decimo sexto arquivo no cartão
1145      delay(3000);          // Delay de 3 segundos para exibir o sinal de 3 s
1146    }
1147
1148
1149    if (Ssaida[0][16] >= 0.7)
```

```
1150    {
1151      Serial.println("---Dois---");
1152      myDFPlayer.play(6);  //Le o décimo sétimo arquivo no cartão
1153      delay(3000);         // Delay de 3 segundos para exibir o sinal de 3 s
1154    }
1155
1156    if (Ssaida[0][17] >= 0.7)
1157    {
1158      Serial.println("---Musica---");
1159      myDFPlayer.play(5);  //Le o décimo oitavo arquivo no cartão
1160      delay(3000);         // Delay de 3 segundos para exibir o sinal de 3 s
1161    }
1162
1163    if (Ssaida[0][18] >= 0.7)
1164    {
1165      Serial.println("---Igual---");
1166      myDFPlayer.play(4);  //Le décimo nono arquivo no cartão
1167      delay(3000);         // Delay de 3 segundos para exibir o sinal de 3 s
1168    }
1169
1170      if (Ssaida[0][19] >= 0.7)
1171    {
1172      Serial.println("---Inteligente---");
1173      myDFPlayer.play(3);  //Le o vigésimo arquivo no cartão
1174      delay(3000);         // Delay de 3 segundos para exibir o sinal de 3 s
1175    }
1176
1177      if (Ssaida[0][20] >= 0.7)
1178    {
1179      Serial.println("---Luva---");
1180      myDFPlayer.play(2);  //Le o vigésimo primeiro arquivo no cartão
1181      delay(3000);         // Delay de 3 segundos para exibir o sinal de 3 s
1182    }
1183
1184      if (Ssaida[0][21] >= 0.7)
1185    {
1186      Serial.println("---Sinal---");
1187      myDFPlayer.play(1);  //Le o vigémio segundo arquivo no cartão
1188      delay(3000);         // Delay de 3 segundos para exibir o sinal de 3 s
1189    }
```

```
1190
1191 }
1192
1193 }
```

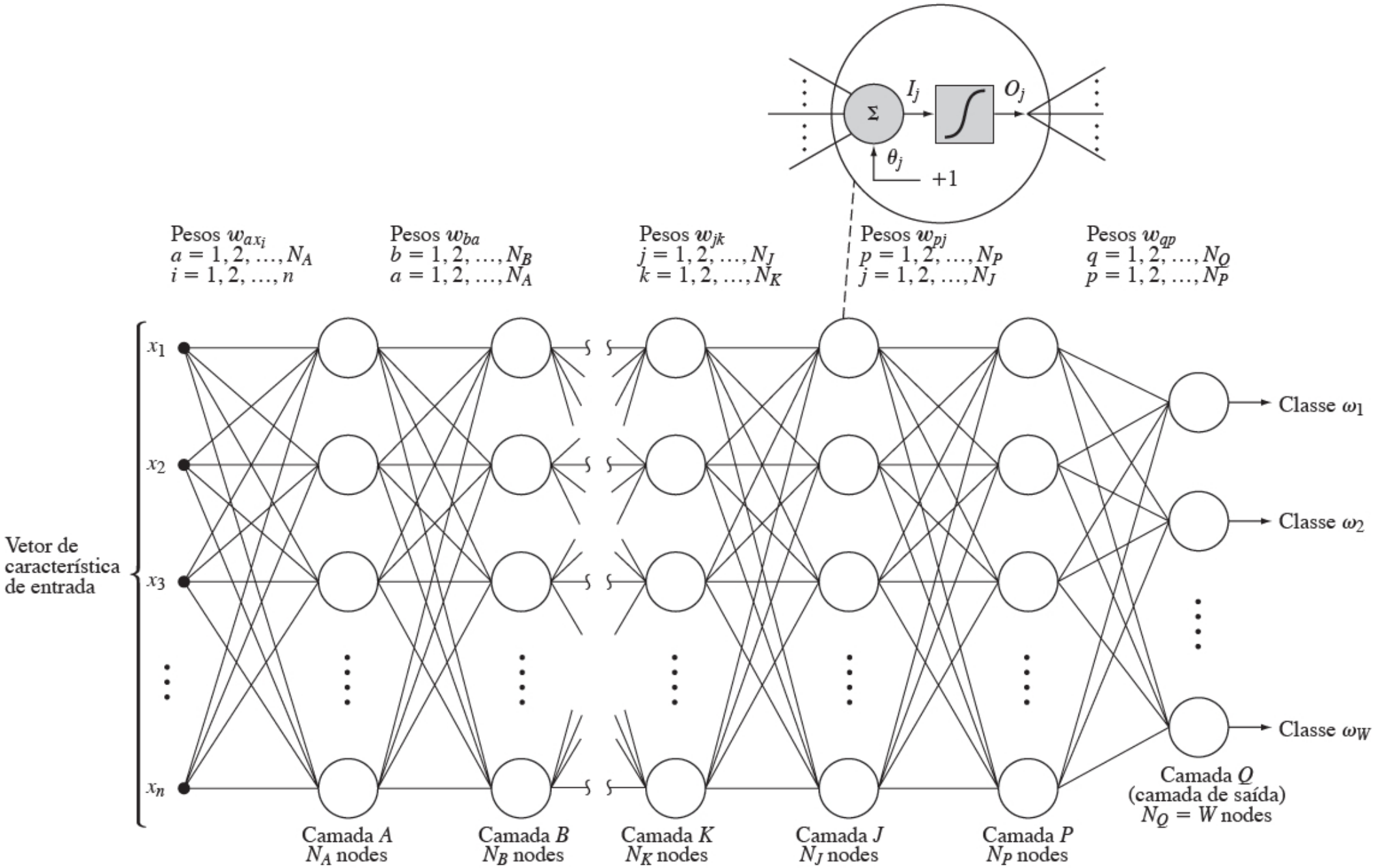
APÊNDICE G – Esquemático do Circuito Eletrônico da LuTLI



NOME DO PROJETO:	Esquemático da LuTLI
INSTITUIÇÃO:	Universidade de Brasília
AUTOR:	Ítalo Rodrigo Moreira Borges
DATA:	07/10/2019
PÁGINA:	1 de 1
TEMPO:	21:02:39

Anexos

ANEXO A – Arquitetura genérica Rede Neural Artificial Multicamada perceptron com retropropagação. Fonte: (GONZALES; WOODS, 2011).


I_j
O_j
Σ
θ_j
+1
Pesos w_{ax_i}
$a = 1, 2, \ldots, N_A$
$i = 1, 2, \ldots, n$
Pesos w_{ba}
$b = 1, 2, \ldots, N_B$
$a = 1, 2, \ldots, N_A$
Pesos w_{jk}
$j = 1, 2, \ldots, N_J$
$k = 1, 2, \ldots, N_K$
Pesos w_{pj}
$p = 1, 2, \ldots, N_P$
$j = 1, 2, \ldots, N_J$
Pesos w_{qp}
$q = 1, 2, \ldots, N_Q$
$p = 1, 2, \ldots, N_P$
Vetor de característica de entrada
x_1
x_2
x_3
x_n
Classe ω_1
Classe ω_2
Classe ω_W
Camada A
N_A nodes
Camada B
N_B nodes
Camada K
N_K nodes
Camada J
N_J nodes
Camada P
N_P nodes
Camada Q
(camada de saída)
$N_Q = W$ nodes

ANEXO B – Materiais Utilizados para a Calibração do sensor MPU6050

- Microcontrolador *Arduino Uno*.
- Microcontrolador *ESP32*.
- *Jumpers*.
- Cabo USB.
- Sensor MPU-6050.
- *Protoboard*.

ANEXO C – Materiais Utilizados para a confecção artesanal do sensor flexível

- Esquadro 60° x 21 cm.
- Tesoura.
- Fita Isolante.
- Papel A4 branco *chamex* multi 210 mm x 297 mm.
- Lápis *Faber Castell* com graduação 9b.
- Fita de cobre para blindagem de guitarras.
- *Jumpers*.
- Estanho.
- Ferro de solda.
- Papel cartão, fita elástica e látex.
- Multímetro.
- *Protoboard*.

ANEXO D – Materiais Utilizados para a integração da luva com os sensores

- Resistores de 1k Ω e 100 Ω.
- Super cola.
- Cabo *Flat*.
- Luva de lã.
- Multímetro.
- *Protoboard*.
- Estanho.
- Ferro de Solda.
- Multímetro.

ANEXO E – Materiais e ferramentas utilizadas para a Montagem do circuito e testes dos algoritmos

- Super cola.
- Cabo *Flat*.
- Luva de lã.
- Multímetro.
- *Protoboard*.
- Módulo *SD card*.
- MPU-6050.
- *ESP32*.
- USB.
- Notebook.
- Software *Arduino*.
- Software *Spyder 3*.
- Pacotes *Keras*,*tensorflow*, *pandas* e *seaborn*.
- *Jumpers*.
- *Push Button*.
- Multímetro.

ANEXO F – Materiais Utilizados para confecção da PCB

- Multímetro.
- Estanho.
- Ferro de solda.
- Módulo *SD card*.
- MPU-6050.
- *ESP32*.
- USB.
- Notebook.
- Software Proteus.
- Alto falante.
- Módulo MP3.
- MPU-6050.
- Placa de circuito impresso.
- Luva com sensores flexíveis.
- Borracha de poliamida.
- Linha de costura.
- Velcro
- *Push Button*.

- Multímetro.
- *Protoboard*.

ANEXO G – Custo Final da LuTLI

Tabela 5 – Custo da LuTLI. Fonte: Autor.

Quantidade	**Produto**	**Preço (R$)**
1	Módulo *SD card*	14,99
1	Módulo MP3	6,99
5	Sensor flexível	6,86
1	MPU-6050	19,99
1	Luva de lã	4,99
1	*ESP32-WROOM-32*	10,66
1	*Push Button*	0,20
1	Placa de Circuito Impresso	40,00
1	Cabo *Flat* 50 *cm*	10,20
1	Bateria lítio	16,25
1	Auto falante	4,69
1	Borracha Poliamida	10,00
1	Preço final	145,82